Active Integrity Constraints and Revision Programming

Luciano Caroprese[1] and Mirosław Truszczyński[2]
[1] *Università della Calabria*
87030 Rende, Italy
E mail: caroprese@deis.unical.it and
[2] *Department of Computer Science, University of Kentucky*
Lexington, KY 40506, USA
E mail: mirek@cs.uky.edu

submitted 25 April 2009; revised 18 November 2009, 17 May 2010; accepted 3 August 2010

Abstract

We study *active integrity constraints* and *revision programming*, two formalisms designed to describe integrity constraints on databases and to specify policies on *preferred* ways to enforce them. Unlike other more commonly accepted approaches, these two formalisms attempt to provide a declarative solution to the problem. However, the original semantics of *founded repairs* for active integrity constraints and *justified revisions* for revision programs differ. Our main goal is to establish a *comprehensive framework* of semantics for active integrity constraints, to find a parallel framework for revision programs, and to relate the two. By doing so, we demonstrate that the two formalisms proposed independently of each other and based on different intuitions when viewed within a broader semantic framework turn out to be notational variants of each other. That lends support to the adequacy of the semantics we develop for each of the formalisms as the foundation for a declarative approach to the problem of database update and repair. In the paper we also study computational properties of the semantics we consider and establish results concerned with the concept of the minimality of change and the invariance under the shifting transformation.

KEYWORDS: inconsistent databases, active integrity constraints, revision programming

1 Introduction

Integrity constraints are conditions on databases. If a database violates integrity constraints, it needs to be *repaired* — updated so that the integrity constraints hold again. Often there are several ways to enforce integrity constraints. The paper is concerned with the problem to *specify* policies for preferred ways to repair databases in a *declarative* way as part of the description of integrity constraints.

A database can be viewed as a finite set of ground atoms in the language of first-order logic determined by the database schema and an infinite countable set of constants. An integrity constraint can be modeled by a formula in this language. A database *satisfies* an integrity constraint if it is its *Herbrand* model. Since databases and sets of integrity constraints are *finite*, without loss of generality, we will limit our attention to the case when databases are subsets of some finite set At of *propositional* atoms, and integrity constraints are clauses in the propositional language generated by At. The notions we propose and the results we obtain in that restricted setting lift to the first-order one (including aggregate operations and built-in predicates) via the standard concept of *grounding*. We do not discuss this matter here in more detail, as our main objective is to develop a semantic framework for declarative specifications of repair policies rather than to study practical issues of possible implementations.

To illustrate the problem of database repair with respect to integrity constraints, let us consider the database $\mathcal{I} = \{a, b\}$ and the integrity constraint $\neg a \vee \neg b$. Clearly, $\mathcal{I}$ does not satisfy $\neg a \vee \neg b$ and needs to be "repaired" — replaced by a database that satisfies the constraint. Assuming $At = \{a, b, c, d\}$, the databases $\emptyset$, $\{a\}$, $\{b\}$, $\{a, c\}$ are examples of databases that could be considered as replacements for $\mathcal{I}$. Since the class of replacements of $\mathcal{I}$ is quite large, the question arises whether there is a principled way to narrow it down. One of the most intuitive and commonly accepted postulates is that the change between the initial database $\mathcal{I}$ and the revised database $\mathcal{R}$, given by $\mathcal{I} \div \mathcal{R}$, be minimal (an example of an early work exploiting that idea is the paper by Winslett (1990); for a more detailed discussion of the role of minimality in studies of database updates we refer to the paper by Chomicki (2007)). In our case, the minimality of change narrows down the class of possible revisions to $\{a\}$ and $\{b\}$.

In some cases, the minimality of change is not specific enough and may leave too many candidate revisions. The problem can be addressed by formalisms that allow the database designer to formulate integrity constraints and, in addition, to state preferred ways for enforcing them. In this paper, which represents an extended version of two conference papers (Caroprese and Truszczyński 2008b; Caroprese and Truszczyński 2008a), we study two such formalisms: active integrity constraints introduced by Caroprese, Greco, Sirangelo and Zumpano (2006), and revision programming introduced by Marek and Truszczyński (1998).

Active integrity constraints and revision programs are languages for specifying integrity constraints. However, unlike in the standard case, when integrity constraints are just first-order formulas that make no distinctions among its models, both sets of active integrity constraints and revision programs are meant to represent *policies* for preferring some models over others. In other words, they give database system designers means to express policies for narrowing down the space of models that need to be considered when repairing inconsistencies or when querying an inconsistent database. In a sense, the two formalisms arise from the need to provide declarative counterparts to procedural attempts to accomplish the same objective (Widom and Ceri 1996; Jagadish et al. 1999). An in-depth understanding of the semantics and, in general, properties of these two formalisms is then essential. Developing that understanding is the main goal of our paper.

To recall, active integrity constraints *explicitly* encode both integrity constraints and preferred basic actions to repair them, in the case when the constraints are violated. To specify a precise meaning of sets of active integrity Caroprese et al.

(2006) proposed the semantics of *founded repairs*. Founded repairs are change-minimal and satisfy a certain groundedness condition.

Revision programs consist of *revision rules*. Each revision rule represents an integrity constraint, and *implicitly* encodes preferred ways to enforce it by means of a certain syntactic convention. Following intuitions from logic programming, Marek and Truszczyński (1998) proposed two semantics for revision programs: the semantics of *justified* revisions and the semantics of *supported* revisions. Each semantics reflects some form of preferences on ways to repair a database given a revision program.

The original semantics of active integrity constraints and revision programming seemingly cannot be related in any direct way. They have different computational properties. For instance, the problem of the existence of a founded repair for a set of normal active integrity constraints is Σ_P^2-complete, while the same problem for justified revisions of normal revision programs is NP-complete. Furthermore, while the semantics for revision programming do not have the minimality of change property, founded repairs with respect to active integrity constraints do.

In this paper, we demonstrate that despite the differences in the syntax, and the lack of a simple correspondence between justified revisions and founded repairs, the formalisms of revision programs and active integrity constraints are closely related. There are two keys to the relationship. First, we need a certain syntactic restriction on revision programs. Specifically, we introduce the class of *proper* revision programs and show that restricting to proper programs does not affect the expressive power.

Second, we need to broaden the families of the semantics for each formalism so that the two sides could be aligned. To this end for active integrity constraints we introduce new semantics by dropping the minimality of change condition, which results in the semantics of *weak repairs* and *founded weak repairs*. We also adapt to the case of active integrity constraints the semantics of justified revisions (justified weak revisions), which leads us to the semantics of *justified weak repairs* and *justified repairs*. For revision programs, we modify the semantics of revisions and justified revisions by imposing on them the minimality condition. Moreover we introduce the semantics of *founded revisions* (*founded weak revisions*) that corresponds to the semantics of founded repairs (founded weak repairs). We show that under a simple bijection between proper revision programs and active integrity constraints, founded (weak) revisions correspond to founded (weak) repairs and justified (weak) revisions correspond to justified (weak) repairs. This result demonstrates that both formalisms, even though rooted in different intuitions, can be "completed" so that to become notational variants of each other.

Both in the case of active integrity constraints and revision programs, the concepts of "groundedness" we consider do not imply, in general, the property of the minimality of change. However, there are broad classes of sets of active integrity constraints, as well as classes of revision programs when it is so. In the paper, we present one class of sets of active integrity constraints, for which, independently of what database they are considered with, groundedness based on the notion of being justified does imply minimality, that is, for which justified weak repairs are minimal and so, are justified repairs (cf. Theorem 4). We also show that for every set of active integrity constraints there is a class of databases such that the minimality of justified weak repairs is guaranteed (cf. Theorem 3). Because of the correspondence between active integrity constraints and revision programs, one can derive analogous results for revisions programs.

A fundamental property of semantics describing database updates is the invariance under a certain transformation of repair instances that consists of (1) removing some elements from a database and adding to it some other elements (thus, "shifting" the database into a different one), and then (2) rewriting active integrity constraints by replacing literals to reflect the changed status of some atoms in the database (cf. Section 13 for a detailed definition). Intuitively such a transformation, we call it *shifting*, when applied to a database and a set of integrity constraints should result in a new database repair instance, "isomporphic" to the original one under any reasonable database repair semantics. We show that it indeed is so for all the semantics we consider in the paper. Thanks to the correspondence between the setting of active integrity constraints and revision programs, the same holds true in that latter setting, too. Shifting is an important property. It allows us to reduce the general database repair (revision) problem, which is specified by two parameters, a database and a set of active integrity constraints (or a revision program), to a special case, when the database to be repaired is empty. The resulting setting is simpler as it involves one parameter only (a set of active integrity constraints or a revision program, respectively). An important consequence of this is the existence of a direct way, in which database repair problem can be related to standard logic programming with the semantics of supported and stable models (Marek and Truszczyński 1998; Pivkina 2001). This paves the way to computational techniques for finding database repairs and revisions.

The paper is organized as follows. In the following section, we situate our paper in the context of some related work. In Section 3, we give a formal introduction to the database update problem. In Section 4, we recall basic concepts related to active integrity constraints, including the semantics of repairs and founded repairs (Caropresse et al. 2006). Next, for a set of active integrity constraints we define weak repairs, founded weak repairs, justified weak repairs and justified repairs. We then discuss the *normalization* of active integrity constraints in Section 6. We prove that justified repairs of a database with respect to the "normalization" of a set of arbitrary active integrity constraints are justified repairs of this database with respect to the original ("non-normalized") active integrity constraints (cf. Theorem 5). This class of justified repairs is the most restrictive semantics for the database repair problem among those we consider. Thus, it offers repairs that can be regarded as most strongly grounded in a database repair instance (a database and a set of active integrity constraints).

Section 7 contains complexity results concerning the existence of repairs of the types we consider in the paper, and Section 8 gives a brief summary of our knowledge concerning the semantics of active integrity constraints. In particular, we discuss there the relationships among the semantics as well as how one could take advantage of the multitude of the semantics considered to handle inconsistency (non-existence of repairs of the most restrictive types).

Next, we recall basic concepts of revision programming. We then introduce some new semantics for revision programs. In Section 11 we establish a precise connection between active integrity constraints and revision programs. We also obtain some complexity results.

Section 13 is concerned with the shifting transformation (Marek and Truszczyński 1998; Pivkina 2001). We show that all semantics discussed in the paper (for either formalism) are invariant under the shifting transformation (the proofs of those results are quite technical and we provide them in the appendix). The last section of the paper offers additional discussion of the contributions of the paper and lists some open problems.

We close the introduction by stressing that our goal is not to single out any of the semantics as the "right" one. For instance, while the semantics of justified repairs (revisions) seems to be best motivated by the principles of groundedness and minimality, the semantics given by the justified repairs (revisions) of the normalization of active integrity constraints (revision programs), being even more restrictive, certainly deserves attention. And, in those cases when justified semantics do not offer any repairs (revisions) relaxing the minimality requirement or the groundedness requirement offers justified weak repairs (revisions) or founded repairs (revisions) that one could use to enforce constraints. We discuss this matter, as well as computational trade-offs, in Section 8 and at the end of Section 12.

2 Related Work

Integrity constraints may render a database inconsistent. Addressing database inconsistency is a problem that has been studied extensively in the literature, and several approaches to database maintenance under integrity constraints have been proposed.

Our work is closely related to studies of *event-condition-event* (ECA) rules in *active* databases (Widom and Ceri 1996). The main difference is that while the formalisms of active integrity constraints and revision programs are declarative, ECA rules have only been given a procedural interpretation.

To recall, an ECA rule consists of three parts:

1. Event: It specifies situations that trigger the rule (e.g. the insertion, deletion or update of a tuple, the execution of a query, the login by a user)
2. Condition: It usually models an integrity constraint. Being true in a triggered ECA rule means the constraint is violated and causes the execution of the action
3. Action: Typically, it is a set of update actions (*insert*, *delete*, *update*). It is executed when the condition of a triggered rule is true.

ECA rules without the event part are called *condition-action* (CA) rules. The structure of CA rules is similar to *normal* active integrity constraints, as we consider them here. In this sense, the formalisms of ECA rules and active integrity constraints are similar. However, there are significant differences, too. Most importantly, the work on ECA rules focused so far only on *procedural* semantics and particular rule processing algorithms. These algorithms determine which ECA rules are invoked and in what order. They use different methods for conflict resolution (needed when several rules are triggered at the same time), and for ensuring termination (executing an action of a rule may make another triggered rule applicable, whose action in turn may make the first rule applicable again).

Another approach to specify the policy for selecting a rule from among those that were activated was proposed by Jagadish et al. (1999). It is based on the specification of a set of *meta-rules* of four types:

1. *Positive requirement meta-rules*: A meta-rule of this type specifies that if a rule A executes, than a rule B must execute as well.
2. *Disabling Rules*: A meta-rule of this type specifies if a rule A is executed then a rule B will not be executed and vice versa.
3. *Preference meta-rules*: A preference meta-rule specifies a preference between two rules. If A is preferable over B and both are fireable then A will be fired.
4. *Scheduling meta-rules*: A meta-rule of this type specifies the order of execution of two fireable rules.

Again, so far only procedural approaches to interpret meta-rules have been developed and studied.

In the two cases discussed, the lack of declarative semantics means there are no grounds for a principled evaluation of rule processing algorithms. In contrast, in our work we focus on declarative semantics for sets of active integrity constraints and revision programs. In particular, we propose several new semantics and study their properties. Our results apply to CA rules and, in fact, they are more general, as active integrity constraints allow several possible actions to choose from. On the other hand, at present our formalisms do not allow us to specify triggering events.

Our work is also related to studies of *consistent query answering* (Arenas et al. 1999; Arenas et al. 2003).[1] That research established a logical characterization of the notion of a consistent answer in a relational database that may violate integrity constraints, developed properties of consistent answers, and methods to compute them.

The notion of a consistent answer is based on the notion of *repair*. A repair of a database is a database that is consistent with respect to a given set of integrity constraints and differs minimally from the original one. A consistent answer to a query Q over a (possibly inconsistent) database $\mathcal{I}$ with respect to a set of integrity constraints is a tuple that belongs to the answers to the same query over all repairs of $\mathcal{I}$. Computing consistent answers exploits the notion of a *residue* (Chakravarthy et al. 1990). Given a query and a set of integrity constraints over a database $\mathcal{I}$, instead of computing all the repairs of $\mathcal{I}$ and querying them, the consistent answers are obtained by computing a new query and submitting it to $\mathcal{I}$. The answers to

[1] Chomicki (2007) gives an in-depth overview of this line of research.

the new query are exactly the consistent answers to the original one. The soundness, completeness and termination of this technique is proved for several classes of constraints and queries. However, the completeness is lost in the case of disjunctive or existential queries. Arenas, Bertossi and Chomicki (2003) present a more general approach that allows us to compute consistent answers to any first-order query. It is based on the notion of a logic program with exceptions. Bravo and Bertossi (2006) study the problem of consistent query answering for databases with *null* values. They propose a semantics for integrity constraint satisfaction for that setting. Marileo and Bertossi (2007) developed a system for computing consistent query answers based on that semantics.

In research on consistent query answering, the semantics of choice is that of minimal change — queries are answered with respect to all databases that differ minimally from the present one and that satisfy all integrity constraints. Thus, no distinction is made among different ways inconsistencies could be removed and no formalisms for specifying policies for removing inconsistencies are discussed. The objectives of the research on active integrity constraints and revision programs have been, in a sense, orthogonal. Up to now (including this paper), the main focus was on embedding within integrity constraints declarative policies for removing inconsistencies, and on establishing possible semantics identifying candidate databases to consider as repairs. It has not yet addressed the problem of consistent query answering with respect to these semantics, an intriguing and important problem to address in the future.

A closely related framework to ours was proposed and studied by Greco et al. (2003). It was designed for computing repairs and consistent answers over inconsistent databases. Greco et al. (2003) defined a repair as an inclusion-minimal set of update actions (insertions and deletions) that makes the database consistent with respect to a set of integrity constraints. The framework relies on *repair constraints*, rules that specify a set of insertions and deletions which are disallowed, and *prioritized constraints*, rules that define priorities among repairs. In that framework, to compute repairs or the consistent answers, one rewrites the constraints into a prioritized extended disjunctive logic programs with two different forms of negation (negation as failure and classical negation). As shown by Caroprese et al. (2006), the framework can be cast as a special case of the formalism of active integrity constraints. A different notion of minimality, based on the cardinality of sets of insert and delete actions, is studied in (Lopatenko and Bertossi 2006). This work presents a set of detailed complexity results of the problem of consistent query answering in the case only cardinality-based repairs are considered.

Katsuno and Mendelzon (1991), consider the problem of knowledge base updates. They analyze some knowledge base update operators and propose a set of postulates knowledge base update operators should satisfy, but do not advocate any particular update operator. For Katsuno and Mendelzon a knowledge base is a propositional formula. Our setting is much more concrete as we consider databases, knowledge bases that are conjunctions of atoms and integrity constraints and, importantly, where updates are restricted to insertion or deletions of atoms. Moreover, our focus is not in update operators but on defining types of databases that can result from a given database when integrity constraints are enforced according to policies they encode. However, the semantics we propose and study in the paper give rise to knowledge base operators that could be considered from the standpoint of Katsuno-Mendelzon postulates. We provide additional comments on that mater in the last section of the paper.

3 Integrity Constraints and Database Repairs — Basic Concepts

Databases and entailment. We consider a finite set At of propositional atoms. We represent databases as subsets of At. A database $\mathcal{I}$ entails a literal $L = a$ (respectively, $L = not\, a$), denoted by $\mathcal{I} \models L$, if $a \in \mathcal{I}$ (respectively, $a \notin \mathcal{I}$). Moreover, $\mathcal{I}$ entails a set of literals S, denoted by $\mathcal{I} \models S$, if it entails each literal in S.

Update actions, consistency. Databases are *updated* by inserting and deleting atoms. An *update action* is an expression of the form $+a$ or $-a$, where $a \in At$. Update action $+a$ states that a is to be inserted. Similarly, update action $-a$ states that a is to be deleted. We say that a set $\mathcal{U}$ of update actions is *consistent* if it does not contain update actions $+a$ and $-a$, for any $a \in At$.

Sets of update actions determine database updates. Let $\mathcal{I}$ be a database and $\mathcal{U}$ a consistent set of update actions. We define the result of *updating* $\mathcal{I}$ *by means of* $\mathcal{U}$ as the database

$$\mathcal{I} \circ \mathcal{U} = (\mathcal{I} \,\cup\, \{a \mid +a \in \mathcal{U}\}) \,\setminus\, \{a \mid -a \in \mathcal{U}\}\cdot$$

We have the following straightforward property of the operator $\circ$, which asserts that if a set of update actions is consistent, the order in which they are executed is immaterial.

Proposition 1

If $\mathcal{U}_1$ and $\mathcal{U}_2$ are sets of update actions such that $\mathcal{U}_1 \cup \mathcal{U}_2$ is consistent, then for every database $\mathcal{I}$, $\mathcal{I} \circ (\mathcal{U}_1 \cup \mathcal{U}_2) = (\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2$

Integrity constraints, entailment (satisfaction). It is common to impose on databases conditions, called *integrity constraints*, that must always be satisfied. In the propositional setting, an *integrity constraint* is a formula

$$r = L_1,\ \ldots,\ L_m \supset \bot, \tag{1}$$

where L_i, $1 \leq i \leq m$, are literals and ',' stands for the conjunction. Any subset of At (and so, also any database) can be regarded as a propositional interpretation. We say that a database $\mathcal{I}$ *satisfies* an integrity constraint r, denoted by $\mathcal{I} \models r$, if $\mathcal{I}$ satisfies the propositional formula represented by r. Moreover, $\mathcal{I}$ *satisfies* a set R of integrity constraints, denoted by $\mathcal{I} \models R$, if $\mathcal{I}$ satisfies each integrity constraint in R. In this way, an integrity constraint encodes a condition on databases: the conjunction of its literals must not hold (or equivalently, the disjunction of the corresponding dual literals must hold).

Any language of (propositional) logic could be used to describe integrity constraints (in the introduction we used the language with the connectives $\vee$ and $\neg$).

Our present choice is reminiscent of the syntax used in logic programming. It is not coincidental. While for integrity constraints we adopt a classical meaning of the logical connectives, for active integrity constraints the meaning depends on and is given by the particular semantics considered. We discuss later several possible semantics for active integrity constraints and discuss their properties. In most of them, the way we interpret boolean connectives, in particular, the negation and the disjunction, has some similarities to the default negation operator in logic programming and so, as it is common in the logic programming literature, we denote them with *not* and $|$ rather than $\neg$ and $\vee$.

Given a set η of integrity constraints and a database $\mathcal{I}$, the problem of *database repair* is to update $\mathcal{I}$ so that integrity constraints in η hold.

Definition 1 (Weak Repairs and Repairs)
Let $\mathcal{I}$ be a database and η a set of integrity constraints. A *weak repair* for $\langle \mathcal{I}, \eta \rangle$ is a consistent set $\mathcal{U}$ of update actions such that $(\{+a \mid a \in \mathcal{I}\} \cup \{-a \mid a \in At \setminus \mathcal{I}\}) \cap \mathcal{U} = \emptyset$ ($\mathcal{U}$ consists of "essential" update actions only), and $\mathcal{I} \circ \mathcal{U} \models \eta$ (constraint enforcement).

A consistent set $\mathcal{U}$ of update actions is a *repair* for $\langle \mathcal{I}, \eta \rangle$ if it is a weak repair for $\langle \mathcal{I}, \eta \rangle$ and for every $\mathcal{U}' \subseteq \mathcal{U}$ such that $\mathcal{I} \circ \mathcal{U}' \models \eta$, $\mathcal{U}' = \mathcal{U}$ (minimality of change). $\square$

If an original database satisfies integrity constraints (formally, if $\mathcal{I} \models \eta$), then no change is needed to enforce the constraints and so $\mathcal{U} = \emptyset$ is the *only* repair for $\langle \mathcal{I}, \eta \rangle$. However, there may be other *weak* repairs for $\langle \mathcal{I}, \eta \rangle$. This points to the problem with weak repairs. They allow for the possibility of updating $\mathcal{I}$ by means of a weak repair $\mathcal{U}$ for $\langle \mathcal{I}, \eta \rangle$ even when $\mathcal{I}$ does not violate η. Thus, the minimality of change is a natural and useful property and, for the most part, we are interested in properties of repairs and their refinements. However, considering weak repairs explicitly is useful as it offers a broader perspective.

If a set η of integrity constraints is inconsistent, there is no database satisfying it (constraints cannot be enforced). In such case, the database repair problem is trivial and not interesting. For that reason, it is common in the database research to restrict investigations to the case when integrity constraints are consistent. However, assuming consistency of integrity constraints does not yield any significant simplifications in our setting. Moreover, as we point out in the next section, a different notion of inconsistency arises in formalisms we study here that is more relevant and interesting. Therefore, in this paper, we do not adopt the assumption that integrity constraints are consistent.

Finally, we note that the problem of the existence of a weak repair is NP-complete (it is just a simple reformulation of the SAT problem). Indeed, given a database $\mathcal{I}$ and a set of integrity constraints $\eta = \{L_{1,1}, \ldots, L_{1,m_1} \supset \bot, \ldots, L_{n,1}, \ldots, L_{n,m_n} \supset \bot\}$, a weak repair for $\langle \mathcal{I}, \eta \rangle$ exists if and only if η is satisfiable (we point out that the class of propositional integrity constraints is, modulo a standard syntactic transformation, the same as the class of all propositional CNF theories). Since repairs exist if and only if weak repairs do, the problem of the existence of a repair is NP-complete, too.

4 Active Integrity Constraints - an Overview

Given no other information but a set of integrity constraints, we have no reason to prefer one repair over another. If several repairs are possible, guidance on how to select a repair to execute could be useful. The formalism of *active integrity constraints* (Caroprese et al. 2006) was designed to address this problem. We will now review it and offer a first extension by introducing the semantics of founded weak repairs.

Dual literals, dual update actions, mappings $ua(\cdot)$ and $lit(\cdot)$. For a propositional literal L, we write L^D for the dual literal to L. Further, if $L = a$, we define $ua(L) = +a$. If $L = not\, a$, we define $ua(L) = -a$. Conversely, for an update action $\alpha = +a$, we set $lit(\alpha) = a$ and for $\alpha = -a$, $lit(\alpha) = not\, a$. We call $+a$ and $-a$ the *duals* of each other, and write α^D to denote the update action dual to an update action α. Finally, we extend the notation introduced here to sets of literals and sets of update actions, as appropriate.

Active integrity constraints, the body and head. An *active integrity constraint* (*aic*, for short) is an expression of the form

$$r = L_1,\ \ldots,\ L_m \supset \alpha_1 | \ldots | \alpha_k \tag{2}$$

where L_i are literals, α_j are update actions, and

$$\{lit(\alpha_1)^D, \ldots, lit(\alpha_k)^D\} \subseteq \{L_1, \ldots, L_m\}. \tag{3}$$

The set $\{L_1, \ldots, L_m\}$ is the *body* of r; we denote it by $body(r)$. Similarly, the set $\{\alpha_1, \ldots, \alpha_k\}$ is the *head* of r; we denote it by $head(r)$.

Active integrity constraints as integrity constraints; entailment (satisfaction). An active integrity constraint with the empty head can be regarded as an integrity constraint (and so, we write the empty head as $\bot$, for consistency with the notation of integrity constraints). An active integrity constraint with a non-empty body can be viewed as an integrity constraint that *explicitly* provides support for some update actions to apply. Namely, the body of an active integrity constraint r of the form (2) represents a *condition* that must be *false* and so, it represents the integrity constraint $L_1,\ \ldots,\ L_m \supset \bot$. Thus, we say that a database $\mathcal{I}$ *satisfies* an active integrity constraint r if it satisfies the corresponding integrity constraint $L_1,\ \ldots,\ L_m \supset \bot$. We write $\mathcal{I} \models r$ to denote that. This concept extends to sets of active integrity constraints in the standard way. However, an active integrity constraint is more than just an integrity constraint. It also provides support for use of update actions that are listed in its head.

Updatable and non-updatable literals. The role of the condition (3) is to ensure that an active integrity constraint supports only those update actions that can "fix" it (executing them ensures that the resulting database satisfies the constraint). The condition can be stated concisely as follows: $[lit(head(r))]^D \subseteq body(r)$. We call literals in $[lit(head(r))]^D$ *updatable* by r. They are precisely those literals that can be affected by an update action in $head(r)$. We call every literal in $body(r) \setminus [lit(head(r))]^D$ *non-updatable* by r. We denote the set of literals updatable by r as $up(r)$ and the set of literals non-updatable by r as $nup(r)$.

With the notation we introduced, we can discuss the intended meaning of an active integrity constraint r of the form (2) in more detail. First, r functions as an integrity constraint $L_1, \ldots, L_m \supset \bot$. Second, it provides support for one of the update actions α_i, *assuming all non-updatable literals in* r *hold in the repaired database*. In particular, the constraint $a, b \supset -a| - b$, given $\mathcal{I} = \{a, b\}$, provides the support for $-a$ or $-b$, independently of the repaired database, as it has no non-updatable literal. In the same context of $\mathcal{I} = \{a, b\}$, the constraint $a, b \supset -a$ provides support for $-a$ but only if b is present in the repaired database.

It is now straightforward to adapt the concept of a (weak) repair to the case of active integrity constraints. Specifically, a set $\mathcal{U}$ of update actions is a *(weak) repair* for a database $\mathcal{I}$ with respect to a set η of active integrity constraints if it is a repair for $\mathcal{I}$ with respect to the set of integrity constraints represented by η.

Let us consider the active integrity constraint $r = a, b \supset -b$, and let $\mathcal{I} = \{a, b\}$ be a database. Clearly, $\mathcal{I}$ violates r as the condition expressed in the body of r is *true*. There are two possible repairs of $\mathcal{I}$ with respect to r or, more precisely, with respect to the integrity constraint encoded by r: performing the update action $-a$ (deleting a), and performing the update action $-b$ (deleting b). Since r provides support for the update action $-b$, we select the latter.

Repairs do not need to obey preferences expressed by the heads of active integrity constraints. To formalize the notion of "support" and translate it into a policy to select "preferred" repairs, Caroprese et al. (2006) proposed the concept of a *founded repair* — a repair that is *grounded* (in some sense, *implied*) by a set of active integrity constraints. The following definition, in addition to founded repairs, introduces a new semantics of founded *weak* repairs.

Definition 2 (Founded (weak) repairs)
Let $\mathcal{I}$ be a database, η a set of active integrity constraints, and $\mathcal{U}$ a consistent set of update actions.

1. An update action α is *founded* with respect to $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$ if there is $r \in \eta$ such that $\alpha \in head(r)$, $\mathcal{I} \circ \mathcal{U} \models nup(r)$, and $\mathcal{I} \circ \mathcal{U} \models lit(\beta)^D$, for every $\beta \in head(r) \setminus \{\alpha\}$.
2. The set $\mathcal{U}$ is *founded* with respect to $\langle \mathcal{I}, \eta \rangle$ if every element of $\mathcal{U}$ is founded with respect to $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$.
3. $\mathcal{U}$ is a *founded (weak) repair* for $\langle \mathcal{I}, \eta \rangle$ if $\mathcal{U}$ is a (weak) repair for $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$ is *founded* with respect to $\langle \mathcal{I}, \eta \rangle$. □

The notion of foundedness of update actions is not restricted to update actions in $\mathcal{U}$. In other words, any update action whether in $\mathcal{U}$ or not may be founded with respect to $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$. However, if an update action, say α, is founded with respect to $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$, and $\mathcal{U}$ enforces constraints, that is, $\mathcal{I} \circ \mathcal{U} \models \eta$, then $\mathcal{U}$ must contain α. Indeed, let us assume that α is founded with respect to $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$ by means of an active integrity constraint $r \in \eta$. Let us also assume that $\mathcal{I} \not\models r$, that is, $\mathcal{I} \models body(r)$. By the foundedness, all literals in $body(r)$, except possibly for $lit(\alpha)^D$, are satisfied in $\mathcal{I} \circ \mathcal{U}$. Thus, since $\mathcal{U}$ enforces r, it must contain α. In other words, foundedness of α "grounds" α in $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$

In the same time, it is important to note that just foundedness of a set $\mathcal{U}$ of update actions does not imply the constraint enforcement nor the minimality of change. We show that in the example below. Therefore, in the definition of founded (weak) repairs, the property of being a (weak) repair must be imposed explicitly.

Example 1
Let $\mathcal{I} = \emptyset$ and η consist of the following active integrity constraints:

$$r_1 = \quad not\ a \quad \supset \quad +a$$
$$r_2 = \quad not\ b, c \quad \supset \quad +b$$
$$r_3 = \quad b, not\ c \quad \supset \quad +c\cdot$$

The unique founded repair for $\langle \mathcal{I}, \eta \rangle$ is $\{+a\}$. The set $\{+a, +b, +c\}$ is founded, guarantees constraint enforcement (and so, it is a founded weak repair), but it is *not* change-minimal. The set $\{+b, +c\}$ is founded but does not guarantee constraint enforcement. We also note that foundedness properly narrows down the class of repairs. If $\eta = \{a, b \supset -b\}$, and $\mathcal{I} = \{a, b\}$ (an example we considered earlier), $\mathcal{U} = \{-a\}$ is a repair for $\langle \mathcal{I}, \eta \rangle$ but not a founded repair. □

We emphasize that founded repairs are not minimal founded weak repairs but founded weak repairs that happen to be repairs (are minimal among all repairs). In particular, it is possible that founded *weak* repairs exist but founded repairs do not.

Example 2
Let $\mathcal{I} = \emptyset$ and η consist of the following active integrity constraints:

$$not\ a, b, c \supset +a \qquad not\ b, a, c \supset +b$$
$$not\ c, a, b \supset +c \qquad not\ a \supset \bot$$

We recall that the integrity constraint $not\ a \supset \bot$ is a special active integrity constraint (with an empty head).One can check that the only founded sets of update actions are $\mathcal{U}_1 = \emptyset$ ($\emptyset$ is always vacuously founded) and $\mathcal{U}_2 = \{+a, +b, +c\}$. Moreover, $\mathcal{U}_3 = \{+a\}$ is a repair and $\mathcal{U}_2$ is a weak repair. Thus, $\mathcal{U}_2$ is a founded weak repair but, as it is not minimal, not a founded repair. In fact, there are no founded repairs in this example. □

This example demonstrates that when we encode into integrity constraints a policy for selecting preferred repairs, that policy may be "non-executable" for some databases under the semantics of founded repairs, as founded repairs may simply not exist. Moreover, it may be so even if the set of integrity constraints underlying the active integrity constraints involved is consistent, that is, if weak repairs exist (or, equivalently, if repairs exist, as repairs exist if and only if weak repairs do). The same is possible under the semantics of founded weak repairs and under all other semantics we consider later in the paper. In other words, the assumption of consistency of integrity constraints does not buy us much and so, we decided not to adopt it.

Finally, we discuss the key issue arising in the context of founded repairs that points out to the need of considering other semantics for active integrity constraints. In some cases, founded repairs, despite combining foundedness with change-minimality, are still not grounded strongly enough. The problem is the circularity of support.

Example 3
Let $\mathcal{I} = \{a, b\}$ and let η_1 consist of the following aic's:

$$r_1 = \quad a, b \quad \supset \quad -a$$
$$r_2 = \quad a, not\, b \quad \supset \quad -a$$
$$r_3 = \quad not\, a, b \quad \supset \quad -b\cdot$$

One can check that $\mathcal{U} = \{-a, -b\}$ is a repair for $\langle \mathcal{I}, \eta_1 \rangle$. Moreover, it is a founded repair: $-a$ is founded with respect to $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}$, with r_2 providing the support necessary for foundedness of $-a$ (i.e. Item 1 of Definition 2 is satisfied by $-a$, η_1, $\mathcal{I}$, $\mathcal{U}$ and r_2),while $-b$ is founded with respect to $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}$ because of r_3 (i.e. Item 1 of Definition 2 is satisfied by $-b$, η_1, $\mathcal{I}$, $\mathcal{U}$ and r_3).

The problem is that, arguably, $\mathcal{U} = \{-a, -b\}$ supports itself through *circular* dependencies. The constraint r_1 is the only one violated by $\mathcal{I}$ and is the one forcing the need for a repair. However, r_1 does not support foundedness of $-a$ with respect to $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}$, as $\mathcal{I} \circ \mathcal{U}$ does not satisfy the literal $b \in nup(r_1)$ (required by Item 1 of Definition 2). Similarly, r_1 does not support foundedness of $-b$ with respect to $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}$ (in fact, $-b$ is not even mentioned in the head of r_1). Thus, the support for the foundedness of $-a$ and $-b$ in $\mathcal{U}$ must come from r_2 and r_3 only. In fact, r_2 provides the support needed for $-a$ to be founded with respect to $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}$. However, that requires that b be absent from $\mathcal{I} \circ \mathcal{U}$ and so, $\mathcal{U}$ must contain the update action $-b$. Similarly, the support for foundedness of $-b$ is given by r_3, which requires that a be absent from $\mathcal{I} \circ \mathcal{U}$, that is, that $-a$ be in $\mathcal{U}$. Thus, in order for $-b$ to be founded, $\mathcal{U}$ must contain $-a$, and for $-a$ to be founded, $\mathcal{U}$ must contain $-b$. In other words, the foundedness of $\{-a, -b\}$ is "circular": $-a$ is founded (and so included in $\mathcal{U}$) due to the fact that $-b$ has been included in $\mathcal{U}$, and $-b$ is founded (and so included in $\mathcal{U}$) due to the fact that $-a$ has been included in $\mathcal{U}$, and there is no independent justification for having any of these two actions included — as we noted, r_1 does not "found" any of $-a$ nor $-b$. □

The problem of circular justifications cannot be discarded by simply hoping they will not occur in practice. If there are several independent sources of integrity constraints, such circular dependencies may arise, if only inadvertently.

To summarize this section, the semantics of repairs for active integrity constraints enforces constraints and satisfies the minimality of change property. It has no groundedness properties beyond what is implied by the two requirements. The semantics of founded repairs gives preference to some ways of repairing constraints over others. It only considers repairs whose all elements are founded. However, foundedness may be circular and some founded (weak) repairs may be "self-grounded" as in the example above. In the next section, we address the issue of self-groundedness of founded (weak) repairs.

On the computational side, the complexity of the semantics of repairs is lower than that of founded repairs. From the result stated in the previous section, it follows that the problem of the existence of a repair is NP-complete, while the problem of the existence of a founded repair is Σ_P^2-complete (Caroprese et al. 2006). As we observed earlier, founded repairs are not minimal founded weak repairs and, in general, the existence of founded weak repairs is not equivalent to the existence of founded repairs. Thus, the complexity of the problem to decide whether founded weak repairs exist need not be the same as that of deciding the existence of founded repairs. Indeed, the complexities of the two problems are different (assuming no collapse of the polynomial hierarchy). Namely, the problem of the existence of founded weak repairs is "only" NP-complete (the proof is simple and we omit it).

5 Justified repairs

In this section, we will introduce another semantics for active integrity constraints that captures a stronger concept of groundedness than the one behind founded repairs. The goal is to disallow circular dependencies like the one we discussed in Example 3.

We start by defining when a set of update actions is *closed* under active integrity constraints. Let η be a set of active integrity constraints and let $\mathcal{U}$ be a set of update actions. If $r \in \eta$, and for every *non-updatable* literal $L \in body(r)$ there is an update action $\alpha \in \mathcal{U}$ such that $lit(\alpha) = L$ then, after applying $\mathcal{U}$ or any of its consistent supersets to the initial database, the result of the update, say $\mathcal{R}$, satisfies all non-updatable literals in $body(r)$. To guarantee that $\mathcal{R}$ satisfies r, $\mathcal{R}$ must *falsify* at least one literal in $body(r)$. To this end $\mathcal{U}$ must contain at least one update action from $head(r)$.

Closed sets of update actions. A set $\mathcal{U}$ of update actions is *closed* under an aic r if $nup(r) \subseteq lit(\mathcal{U})$ implies $head(r) \cap \mathcal{U} \neq \emptyset$. A set $\mathcal{U}$ of update actions is *closed* under a set η of active integrity constraints if it is closed under every $r \in \eta$.

If a set of update actions is not closed under a set η of active integrity constraints, executing its elements does not guarantee to enforce constraints represented by η. Therefore closed sets of update actions are important. We regard closed sets of update actions that are also minimal as "forced" by η, as all elements in a minimal set of update actions closed under η are necessary (no nonempty subset can be dropped).

Example 4
Let us consider the database and active integrity constraints from Example 3. The set $\mathcal{U} = \{-a, -b\}$ is closed under η_1. We observe that the empty set is also closed under η_1. Therefore $\mathcal{U}$ is not minimal. □

No-effect actions. Another key notion in our considerations is that of *no-effect actions*. Let $\mathcal{I}$ be a database and $\mathcal{R}$ a result of updating $\mathcal{I}$. An update action $+a$ (respectively, $-a$) is a *no-effect* action with respect to $(\mathcal{I}, \mathcal{R})$ if $a \in \mathcal{I} \cap \mathcal{R}$ (respectively, $a \notin \mathcal{I} \cup \mathcal{R}$). Informally, a no-effect action does not change the status of its underlying atom. We denote by $ne(\mathcal{I}, \mathcal{R})$ the set of all no-effect actions with respect to $(\mathcal{I}, \mathcal{R})$. We note the following two simple properties reflecting the nature of no-effect actions — their redundancy.

Proposition 2
Let $\mathcal{I}$ be a database. Then

1. For every databases $\mathcal{R}, \mathcal{R}'$, if $ne(\mathcal{I}, \mathcal{R}) \subseteq ne(\mathcal{I}, \mathcal{R}')$, then $\mathcal{R}' \circ ne(\mathcal{I}, \mathcal{R}) = \mathcal{R}'$

2. For every set $\mathcal{E}$ of update actions such that $\mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ is consistent, if $\mathcal{E}' \subseteq \mathcal{E}$, then $\mathcal{I} \circ \mathcal{E}' = \mathcal{I} \circ (\mathcal{E}' \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}))$.

Proof:

1. Since $ne(\mathcal{I}, \mathcal{R}) = \{+a \mid a \in \mathcal{I} \cap \mathcal{R}\} \cup \{-a \mid a \notin \mathcal{I} \cup \mathcal{R}\}$ and $ne(\mathcal{I}, \mathcal{R}') = \{+a \mid a \in \mathcal{I} \cap \mathcal{R}'\} \cup \{-a \mid a \notin \mathcal{I} \cup \mathcal{R}'\}$, we have $\mathcal{I} \cap \mathcal{R} \subseteq \mathcal{I} \cap \mathcal{R}'$ and $\mathcal{I} \cup \mathcal{R}' \subseteq \mathcal{I} \cup \mathcal{R}$. It follows that $\mathcal{I} \cap \mathcal{R} \subseteq \mathcal{R}'$ and $\mathcal{R}' \subseteq \mathcal{I} \cup \mathcal{R}$. Thus, $\mathcal{R}' \circ ne(\mathcal{I}, \mathcal{R}) = (\mathcal{R}' \cup (\mathcal{I} \cap \mathcal{R})) \cap (\mathcal{I} \cup \mathcal{R}) = \mathcal{R}'$.
2. As $\mathcal{E}' \subseteq \mathcal{E}$, then $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) \subseteq ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}')$. Since $\mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ is consistent, Propositions 1 and 2(1) imply that $\mathcal{I} \circ (\mathcal{E}' \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})) = (\mathcal{I} \circ \mathcal{E}') \circ ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) = \mathcal{I} \circ \mathcal{E}'$. □

Our semantics of justified repairs is based on the knowledge-representation principle, a form of the frame axiom (McCarthy and Hayes 1969), that remaining in the previous state requires no reason (persistence by inertia). Thus, when justifying update actions necessary to transform $\mathcal{I}$ into $\mathcal{R}$ based on η we assume the set $ne(\mathcal{I}, \mathcal{R})$ as given. This brings us to the notion of a justified weak repair.

Definition 3 (Justified weak repairs)
Let $\mathcal{I}$ be a database and η a set of active integrity constraints. A consistent set $\mathcal{U}$ of update actions is a justified action set for $\langle \mathcal{I}, \eta \rangle$ if $\mathcal{U}$ is a minimal set of update actions containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ and closed under η.

If $\mathcal{U}$ is a justified action set for $\langle \mathcal{I}, \eta \rangle$, then $\mathcal{E} = \mathcal{U} \setminus ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$. □

Intuitively, a set $\mathcal{U}$ of update actions is a justified action set, if it is precisely the set of update actions forced or *justified* by η and the no-effect actions with respect to $\mathcal{I}$ and $\mathcal{I} \circ \mathcal{U}$. This "fixpoint" aspect of the definition is reminiscent of the definitions of semantics of several non-monotonic logics, including (disjunctive) logic programming with the answer set semantics. The connection can be made more formal and we take advantage of it in the section on the complexity and computation.

Before we proceed, we will illustrate the notion of justified weak repairs.

Example 5
Let us consider again Example 3. The set $\mathcal{U} = \{-a, -b\}$ is not a justified weak repair for $\langle \mathcal{I}, \eta_1 \rangle$. One can check that $\mathcal{U} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ $(= \{-a, -b\})$ contains $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ $(= \emptyset)$, and is closed under η_1. But, as we observed in Example 4, it is not a minimal set of update actions containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ and closed under η_1. Indeed, $\emptyset$ has these two properties, too. In fact, one can check that $\langle \mathcal{I}, \eta_1 \rangle$ has no justified weak repairs.

Next, let us consider a new set, η_2, of aic's, where r_1 is replaced with $r_1' = a, b \supset -a | -b$. The constraint r_1' provides support for $-a$ or $-b$ independently of the repaired database (as there are no non-updatable literals in r_1'). If $-a$ is selected (with support from r_1'), r_3 supports $-b$. If $-b$ is selected (with support from r_1'), r_2 supports $-a$, Thus the cyclic support given by r_2 and r_3 in the presence of r_1 is broken. Indeed, one can check that $\{-a, -b\}$ is a justified weak repair, in fact, the only one. □

We note that the set $ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})$ can be quite large. In particular, the cardinality of the set of update actions $-a$, where $a\notin\mathcal{I}\cup\mathcal{R}$, cannot be bounded by the size of the database repair problem, which is given by the size of $\mathcal{I}$ and η. However, only those update actions $-a$ of that type are important from the perspective of justified weak revisions, whose literals $not\,(a)$ occur in the bodies of active integrity constraints in η (as no other update action of that type can play a role in determining minimal sets of update actions closed under integrity constraints).

We will now study justified action sets and justified weak repairs. We start with an alternative characterization of justified weak repairs.

Theorem 1
Let $\mathcal{I}$ be a database, η a set of active integrity constraints and $\mathcal{E}$ a consistent set of update actions. Then $\mathcal{E}$ is a justified weak repair for $\langle\mathcal{I},\eta\rangle$ if and only if $\mathcal{E}\cap ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})=\emptyset$ and $\mathcal{E}\cup ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$ is a justified action set for $\langle\mathcal{I},\eta\rangle$.

Proof: ($\Rightarrow$) Since $\mathcal{E}$ is a justified weak repair for $\langle\mathcal{I},\eta\rangle$, $\mathcal{E}=\mathcal{U}\setminus ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})$ for some consistent set $\mathcal{U}$ of update actions such that $\mathcal{U}$ is minimal containing $ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})$ and closed under η. By Proposition 2(2), $\mathcal{I}\circ\mathcal{U}=\mathcal{I}\circ\mathcal{E}$. Thus, $\mathcal{E}\cap ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})=\emptyset$. Moreover, since $ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})\subseteq\mathcal{U}$, $\mathcal{U}=\mathcal{E}\cup ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$. Hence, $\mathcal{E}\cup ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$ is a justified action set for $\langle\mathcal{I},\eta\rangle$.

($\Leftarrow$) Let $\mathcal{U}=\mathcal{E}\cup ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$. We will show that $ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})=ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$. To this end, let $+a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})$. Then, $a\in\mathcal{I}$ and $-a\notin\mathcal{U}$ (the latter property follows by the consistency of $\mathcal{U}$). It follows that $-a\notin\mathcal{E}$ and, consequently, $+a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$. Similarly, we show that if $-a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})$, then $-a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$. Thus, we obtain that $ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})\subseteq ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$.

Conversely, let $+a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$. Then $a\in\mathcal{I}$ and $+a\in\mathcal{U}$. Since $\mathcal{U}$ is consistent (it is a justified action set for $\langle\mathcal{I},\eta\rangle$), $\mathcal{I}\circ\mathcal{U}$ is well defined and $+a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})$. The case $-a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$ is similar. Thus, $ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})\subseteq ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})$ and the claim follows.

Since $\mathcal{E}\cap ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})=\emptyset$, we obtain that $\mathcal{E}=\mathcal{U}\setminus ne(\mathcal{I},\mathcal{I}\circ\mathcal{U})$. Since $\mathcal{U}$ is a justified action set for $\langle\mathcal{I},\eta\rangle$, $\mathcal{E}$ is a justified weak repair for $\langle\mathcal{I},\eta\rangle$. □

Justified weak repairs have two key properties for the problem of database update: constraint enforcement (hence the term “weak repair”) and foundedness.

Theorem 2
Let $\mathcal{I}$ be a database, η a set of active integrity constraints, and $\mathcal{E}$ a justified weak repair for $\langle\mathcal{I},\eta\rangle$. Then

1. For every atom a, exactly one of $+a$ and $-a$ is in $\mathcal{E}\cup ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$
2. $\mathcal{I}\circ\mathcal{E}\models\eta$
3. $\mathcal{E}$ is founded for $\langle\mathcal{I},\eta\rangle$.

Proof: Throughout the proof, use the notation $\mathcal{U}=\mathcal{E}\cup ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$.

1. Since $\mathcal{U}$ is consistent (cf. Theorem 1), for every atom a, at most one of $+a$, $-a$ is in $\mathcal{U}$. If $+a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$ or $-a\in ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$ then the claim follows. Otherwise, the status of a changes as we move from $\mathcal{I}$ to $\mathcal{I}\circ\mathcal{E}$. That is, either $+a$ or $-a$ belongs to $\mathcal{E}$ and, consequently, to $\mathcal{U}$, as well.

2. Let us consider $r \in \eta$. Since $\mathcal{U}$ is closed under η (cf. Theorem 1), we have $nup(r) \not\subseteq lit(\mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}))$ or $head(r) \cap (\mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})) \neq \emptyset$. Let us assume the first possibility, and let L be a literal such that $L \in nup(r)$ and $ua(L) \notin \mathcal{U}$. By (1), $ua(L^D) \in \mathcal{U}$. Consequently, $\mathcal{I} \circ \mathcal{U} \not\models L$. By Proposition 2(2), $\mathcal{I} \circ \mathcal{E} \not\models L$. Since $L \in body(r)$, $\mathcal{I} \circ \mathcal{E} \models r$.
   Thus, let us assume that $head(r) \cap \mathcal{U} \neq \emptyset$ and let $\alpha \in head(r) \cap \mathcal{U}$. Then $\alpha \in head(r)$ and so, $lit(\alpha)^D \in body(r)$. Furthermore, $\alpha \in \mathcal{U}$ and so, $\mathcal{I} \circ \mathcal{U} \models lit(\alpha)$. By Proposition 2(2), $\mathcal{I} \circ \mathcal{E} \models lit(\alpha)$. Thus, $\mathcal{I} \circ \mathcal{E} \models r$ in this case, too.
3. Let $\alpha \in \mathcal{E}$. By Theorem 1, $\alpha \notin ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$. Thus, $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) \subseteq \mathcal{U} \setminus \{\alpha\}$. Since $\mathcal{U}$ is a minimal set closed under η and containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$, $\mathcal{U} \setminus \{\alpha\}$ is not closed under η. That is, there is $r \in \eta$ such that $nup(r) \subseteq lit(\mathcal{U} \setminus \{\alpha\})$ and $head(r) \cap (\mathcal{U} \setminus \{\alpha\}) = \emptyset$. We have
   $$\begin{aligned} \mathcal{I} \circ (\mathcal{U} \setminus \{\alpha\}) &= \mathcal{I} \circ (ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) \cup (\mathcal{E} \setminus \{\alpha\})) \\ &= (\mathcal{I} \circ ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})) \circ (\mathcal{E} \setminus \{\alpha\}) \cdot \end{aligned}$$
   By Proposition 2 (and the fact that $ne(\mathcal{I}, \mathcal{R}) = ne(\mathcal{R}, \mathcal{I})$, for every databases $\mathcal{I}$ and $\mathcal{R}$),
   $$\mathcal{I} \circ (\mathcal{U} \setminus \{\alpha\}) = \mathcal{I} \circ (\mathcal{E} \setminus \{\alpha\}) \cdot \tag{4}$$
   From $nup(r) \subseteq lit(\mathcal{U} \setminus \{\alpha\})$, it follows that $\mathcal{I} \circ (\mathcal{U} \setminus \{\alpha\}) \models nup(r)$. By (4), $\mathcal{I} \circ (\mathcal{E} \setminus \{\alpha\}) \models nup(r)$. Since $\alpha \in head(r)$, $lit(\alpha^D) \notin nup(r)$. Thus, $I \circ \mathcal{E} \models nup(r)$.
   The inclusion $nup(r) \subseteq lit(\mathcal{U} \setminus \{\alpha\})$ also implies $nup(r) \subseteq lit(\mathcal{U})$. Since $\mathcal{U}$ is closed under η, $head(r) \cap \mathcal{U} \neq \emptyset$ and so, $head(r) \cap \mathcal{U} = \{\alpha\}$.
   Let us consider $\beta \in head(r)$ such that $\beta \neq \alpha$. It follows that $\beta \notin \mathcal{U}$. By (1), $\beta^D \in \mathcal{U}$ and, consequently, $I \circ \mathcal{U} \models \beta^D$. Since $\mathcal{I} \circ \mathcal{U} = \mathcal{I} \circ \mathcal{E}$ (Proposition 2), it follows that α is founded with respect to $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{E}$. □

Theorem 2 directly implies that justified weak repairs are founded weak repairs.

Corollary 1
Let $\mathcal{I}$ be a database, η a set of active integrity constraints, and $\mathcal{E}$ a justified weak repair for $\langle \mathcal{I}, \eta \rangle$. Then, $\mathcal{E}$ is a founded weak repair for $\langle \mathcal{I}, \eta \rangle$.

Examples 3 and 5 show that the converse to Corollary 1 does not hold. That is, there are founded weak repairs that are not justified weak repairs.

While a stronger property than foundedness, being a justified weak repair still does not guarantee change-minimality (and so, the term *weak* cannot be dropped).

Example 6
Let $\mathcal{I}' = \emptyset$, and η_3 be a set of aic's consisting of

$$\begin{aligned} r_1 = \quad & not\, a, b \quad \supset \quad +a \mid -b \\ r_2 = \quad & a, not\, b \quad \supset \quad -a \mid +b \cdot \end{aligned}$$

Clearly, $\mathcal{I}'$ is consistent with respect to η_3.Let us consider the set of update actions $\mathcal{E} = \{+a, +b\}$. It is easy to verify that $\mathcal{E}$ is a justified weak repair for $\langle \mathcal{I}', \eta_3 \rangle$. Therefore, it ensures constraint enforcement and it is founded. However, $\mathcal{E}$ is not minimal and the empty set of update actions is its only repair. □

Thus, to have change-minimality, it needs to be enforced directly as in the case of founded repairs. By doing so, we obtain the notion of *justified repairs*.

Definition 4 (Justified repair)
Let $\mathcal{I}$ be a database and η a set of active integrity constraints. A set $\mathcal{E}$ of update actions is a *justified repair* for $\langle \mathcal{I}, \eta \rangle$ if $\mathcal{E}$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$, and for every $\mathcal{E}' \subseteq \mathcal{E}$ such that $\mathcal{I} \circ \mathcal{E}' \models \eta$, $\mathcal{E}' = \mathcal{E}$. □

Theorem 2 has yet another corollary, this time concerning justified and founded repairs.

Corollary 2
Let $\mathcal{I}$ be a database, η a set of active integrity constraints, and $\mathcal{E}$ a justified repair for $\langle \mathcal{I}, \eta \rangle$. Then, $\mathcal{E}$ is a founded repair for $\langle \mathcal{I}, \eta \rangle$.

Proof: Let $\mathcal{E}$ be a justified repair for $\langle \mathcal{I}, \eta \rangle$. It follows by Theorem 2 that $\mathcal{I} \circ \mathcal{E} \models \eta$. Moreover, by the definition of justified repairs, $\mathcal{E}$ is change minimal. Thus, $\mathcal{E}$ is a repair. Again by Theorem 2, $\mathcal{E}$ is founded. Thus, $\mathcal{E}$ is a founded repair for $\langle \mathcal{I}, \eta \rangle$. □

Examples 3 and 5 show that the inclusion asserted by Corollary 2 is proper. Indeed, we argued in Example 3 that $\{-a, -b\}$ is a founded repair. Then, in Example 5 we showed that it is not a justified weak repair. Thus, $\{-a, -b\}$ is not a justified repair, either.

As illustrated by Example 6, in general, justified repairs form a proper subclass of justified weak repairs. However, in some cases the two concepts coincide — the minimality is a consequence of the groundedness underlying the notion of a justified weak repair. One such case is identified in the next theorem. The other important case is discussed in the next section.

Theorem 3
Let $\mathcal{I}$ be a database and η a set of active integrity constraints such that for each update action $\alpha \in \bigcup_{r \in \eta} head(r)$, $\mathcal{I} \models lit(\alpha^D)$. If $\mathcal{E}$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$, then $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$.

Proof: Let $\mathcal{E}$ be a justified weak repair for $\langle \mathcal{I}, \eta \rangle$ and let $\mathcal{E}' \subseteq \mathcal{E}$ be such that $\mathcal{I} \circ \mathcal{E}' \models \eta$.

We define $\mathcal{U} = \mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$. By Theorem 1 and Proposition 2(2), $\mathcal{U}$ is a minimal set of update actions containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ and closed under η. Let $\mathcal{U}' = \mathcal{E}' \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ and let $r \in \eta$ be such that $ua(nup(r)) \subseteq \mathcal{U}'$. Since $\mathcal{I} \circ \mathcal{E}' \models \eta$, $\mathcal{I} \circ \mathcal{E}' \not\models body(r)$. Thus, it follows that there is $L \in up(r)$ such that $\mathcal{I} \circ \mathcal{E}' \not\models L$. Since $L \in up(r)$, there is $\alpha \in head(r)$ such that $L = lit(\alpha^D)$. By the assumption, $\mathcal{I} \models L$, that is, $\mathcal{I} \models lit(\alpha^D)$. Since $\mathcal{I} \circ \mathcal{E}' \not\models L$, $\mathcal{I} \circ \mathcal{E}' \models lit(\alpha)$. Thus, $\alpha \in \mathcal{E}'$ and, consequently, $\alpha \in \mathcal{U}'$. It follows that $\mathcal{U}'$ is closed under r and, since r was an arbitrary element of η, under η. too. Thus, $\mathcal{U}' = \mathcal{U}$, that is, $\mathcal{E}' \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) = \mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$. Since $E' \subseteq \mathcal{E}$ and $\mathcal{E} \cap ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) = \emptyset$, $\mathcal{E}' = \mathcal{E}$. It follows that $\mathcal{E}$ is a minimal set of update actions such that $\mathcal{I} \circ \mathcal{E} \models \eta$. □

The theorem above states that whenever each update action occurring in η is essential with respect to $\mathcal{I}$ (it is able to perform a real change over $\mathcal{I}$), the minimality of each justified weak repair is guaranteed (that is, it is a justified repair).

6 Normal active integrity constraints and normalization

An active integrity constraint r is *normal* if $|head(r)| \leq 1$. We will now study properties of normal active integrity constraints. First, we will show that for that class of constraints, updating by justified weak repairs guarantees the minimality of change property and so, the explicit reference to the latter can be omitted from the definition of justified repairs.

Theorem 4
Let $\mathcal{I}$ be a database and η a set of normal active integrity constraints. If $\mathcal{E}$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$ then $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$.

Proof: Let $\mathcal{E}$ be a justified weak repair for $\langle \mathcal{I}, \eta \rangle$. We have to prove that $\mathcal{E}$ is minimal with respect to constraint enforcement. To this end, let us consider $\mathcal{E}' \subseteq \mathcal{E}$ such that $\mathcal{I} \circ \mathcal{E}' \models \eta$.

We define $\mathcal{U} = \mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ and $\mathcal{U}' = \mathcal{E}' \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$. We will show that $\mathcal{U}'$ is closed under η. Let $r \in \eta$ be such that $ua(nup(r)) \subseteq \mathcal{U}'$.

Since $\mathcal{I} \circ \mathcal{E}' \models r$, $\mathcal{I} \circ \mathcal{E}' \not\models body(r)$. By our assumption, $ua(nup(r)) \subseteq \mathcal{U}'$. Thus, $\mathcal{I} \circ \mathcal{U}' \models nup(r)$. Since $\mathcal{U}'$ is consistent, Proposition 2(2) implies that $\mathcal{I} \circ \mathcal{E}' = \mathcal{I} \circ \mathcal{U}'$. Thus, $\mathcal{I} \circ \mathcal{E}' \models nup(r)$. If $head(r) = \emptyset$, $\mathcal{I} \circ \mathcal{E}' \models body(r)$ and so, $\mathcal{I} \circ \mathcal{E}' \not\models r$, a contradiction. Thus, $head(r) = \{\alpha\}$, for some update action α. Moreover, as $\mathcal{I} \circ \mathcal{E}' \models r$, $\mathcal{I} \circ \mathcal{E}' \not\models lit(\alpha^D)$. Consequently, $\mathcal{I} \circ \mathcal{E}' \models lit(\alpha)$.

Since $\mathcal{U}' \subseteq \mathcal{U}$, $ua(nup(r)) \subseteq \mathcal{U}$. By Theorem 1, $\mathcal{U}$ is closed under η. Thus, $\alpha \in \mathcal{U}$. Since $\mathcal{I} \circ \mathcal{U} = \mathcal{I} \circ \mathcal{E}$ (Proposition 2(2)), $\mathcal{I} \circ \mathcal{E} \models lit(\alpha)$.

If $\mathcal{I} \models lit(\alpha)$ then, as $\mathcal{I} \circ \mathcal{E} \models lit(\alpha)$, we have $\alpha \in ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) \subseteq \mathcal{U}'$. If $\mathcal{I} \not\models lit(\alpha)$ then, as $\mathcal{I} \circ \mathcal{E}' \models lit(\alpha)$, we have that $\alpha \in \mathcal{E}' \subseteq \mathcal{U}'$. Thus, $\mathcal{U}'$ is closed under r and so, also under η. Consequently, $\mathcal{U}' = \mathcal{U}$. Since $\mathcal{E} \cap ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) = \emptyset$, it follows that $\mathcal{E}' = \mathcal{E}$. Thus, $\mathcal{E}$ is a minimal set of update actions such that $\mathcal{I} \circ \mathcal{E} \models \eta$. □

Normalization. Next, we introduce the operation of *normalization* of active integrity constraints, which consists of eliminating disjunctions from the heads of rules. For an active integrity constraint $r = \phi \supset \alpha_1 | \dots | \alpha_n$, by r^n we denote the set of *normal* active integrity constraints $\{\phi \supset \alpha_1, \dots, \phi \supset \alpha_n\}$. For a set η of active integrity constraints, we set $\eta^n = \bigcup_{r \in \eta} r^n$. It is shown by Caroprese et al. (2006) that $\mathcal{E}$ is founded for $\langle \mathcal{I}, \eta \rangle$ if and only if $\mathcal{E}$ is a founded for $\langle \mathcal{I}, \eta^n \rangle$. Thus, $\mathcal{E}$ is a founded (weak) repair for $\langle \mathcal{I}, \eta \rangle$ if and only if $\mathcal{E}$ is a founded (weak) repair for $\langle \mathcal{I}, \eta^n \rangle$. For justified repairs, we have a weaker result. Normalization may eliminate some justified repairs. That leads to an even more narrow class of repairs than justified ones, an issue we discuss later in Section 8.

Theorem 5
Let $\mathcal{I}$ be a database and η a set of active integrity constraints.

1. If a set $\mathcal{E}$ of update actions is a justified repair for $\langle \mathcal{I}, \eta^n \rangle$, then $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$
2. If a set $\mathcal{E}$ of update action is a justified weak repair for $\langle \mathcal{I}, \eta^n \rangle$, then $\mathcal{E}$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$.

Proof: Let $\mathcal{E}$ be a justified repair for $\langle \mathcal{I}, \eta^n \rangle$. We define $\mathcal{U} = \mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$. By Corollary 2, $\mathcal{E}$ is a founded repair for $\langle \mathcal{I}, \eta^n \rangle$. By a result obtained by Caroprese et al. (2006), $\mathcal{E}$ is a founded repair for $\langle \mathcal{I}, \eta \rangle$ and, consequently, a repair for $\langle \mathcal{I}, \eta \rangle$.

Since $\mathcal{E}$ is, in particular, a justified weak repair for $\langle \mathcal{I}, \eta^n \rangle$, $\mathcal{U}$ is a justified action set for $\langle \mathcal{I}, \eta^n \rangle$ (Theorem 1). Thus, $\mathcal{U}$ is a minimal set of update actions containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ and closed under η^n. To prove that $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$, it suffices to show that $\mathcal{U}$ is a minimal set of update actions containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ and closed under η.

Let us consider an active integrity constraint

$$r = lit(\alpha_1^D), \ldots, lit(\alpha_n^D), \phi \supset \alpha_1 | \ldots | \alpha_n$$

in η such that $ua(nup(r)) \subseteq \mathcal{U}$ (we note that $nup(r)$ consists precisely of the literals that appear in ϕ). It follows that $I \circ \mathcal{U} \models nup(r)$. Since $\mathcal{E}$ is a repair, $\mathcal{I} \circ \mathcal{E} \not\models body(r)$. By Proposition 2(2), $\mathcal{I} \circ \mathcal{E} = \mathcal{I} \circ \mathcal{U}$. Thus, $\mathcal{I} \circ \mathcal{U} \not\models body(r)$. It follows that there is i, $1 \leq i \leq n$, such that $\mathcal{I} \circ \mathcal{U} \not\models lit(\alpha_i^D)$. Thus, $\alpha_i^D \notin \mathcal{U}$. By Theorem 2(1), $\alpha_i \in \mathcal{U}$. Thus, $\mathcal{U}$ is closed under r and, consequently, under η, as well.

We will now show that $\mathcal{U}$ is minimal in the class of sets of update actions containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ and closed under η. Let $\mathcal{U}'$ be a set of update actions such that $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) \subseteq \mathcal{U}' \subseteq \mathcal{U}$ and $\mathcal{U}'$ is closed under η. Let us consider an active integrity constraint in $s \in \eta^n$ such that $ua(nup(s)) \subseteq \mathcal{U}'$.

By the definition of η^n, there is an active integrity constraint $r \in \eta$ such that

$$r = lit(\alpha_1^D), \ldots, lit(\alpha_i^D), \ldots, lit(\alpha_n^D), \phi \supset \alpha_1 | \ldots | \alpha_i | \ldots | \alpha_n$$

and

$$s = lit(\alpha_1^D), \ldots, lit(\alpha_i^D), \ldots, lit(\alpha_n^D), \phi \supset \alpha_i.$$

Since $ua(nup(s) \subseteq \mathcal{U}'$, $ua(nup(r) \subseteq \mathcal{U}'$. As $\mathcal{U}'$ is closed under η, there is j, $1 \leq j \leq n$, such that $\alpha_j \in \mathcal{U}'$. For every k such that $1 \leq k \leq n$ and $k \neq i$, $\alpha_k^D \in \mathcal{U}'$. By the consistency of $\mathcal{U}'$, we conclude that $\alpha_i \in \mathcal{U}'$. Thus, $\mathcal{U}'$ is closed under s and, consequently, under η^n. Since $\mathcal{U}' \subseteq \mathcal{U}$ and $\mathcal{U}$ is minimal containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ and closed under η^n it follows that $\mathcal{U}' = \mathcal{U}$. Thus, $\mathcal{U}$ is minimal containing $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ and closed under η. Consequently, $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$.

(2) If $\mathcal{E}$ is a justified weak repair for $\langle \mathcal{I}, \eta^n \rangle$ then, by Theorem 4, $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta^n \rangle$. By (1), $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$ and so, a justified weak repair for $\langle \mathcal{I}, \eta \rangle$. □

The following example shows that the inclusions in the previous theorem are, in general, proper.

Example 7

Let us consider an empty database $\mathcal{I}' = \emptyset$, the set η_4 of aic's

$$\begin{aligned} r_1 &= \; not\ a, not\ b \supset +a | +b \\ r_2 &= \; a, not\ b \supset +b \\ r_3 &= \; not\ a, b \supset +a, \end{aligned}$$

its normalized version η_4^n

$$\begin{aligned} r_{1,1} &= \; not\ a, not\ b \supset +a & \qquad r_{2,1} &= \; a, not\ b \supset +b \\ r_{1,2} &= \; not\ a, not\ b \supset +b & \qquad r_{3,1} &= \; not\ a, b \supset +a, \end{aligned}$$

and the set of update actions $\mathcal{E} = \{+a, +b\}$. It is easy to verify that $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}', \eta_4 \rangle$. However, $\mathcal{E}$ is not a justified weak repair for $\langle \mathcal{I}', \eta_4^n \rangle$ (and so, not a justified repair for $\langle \mathcal{I}', \eta_4^n \rangle$). Indeed, it is not a minimal set containing $ne(\mathcal{I}', \mathcal{I}' \circ \mathcal{E}) = \emptyset$ and closed under η_4^n, as $\emptyset$ is also closed under η_4^n. □

7 Complexity and Computation

We noted earlier that the problem of the existence of a (weak) repair is NP-complete, and the same is true for the problem of the existence of founded weak repairs. On the other hand, the problem of the existence of a founded repair is Σ_P^2-complete (Caroprese et al. 2006). In this section, we study the problem of the existence of justified (weak) repairs.

For our hardness results, we will use problems in logic programming. We will consider disjunctive and normal logic programs that satisfy some additional syntactic constraints. Namely, we will consider only programs without rules which contain multiple occurrences of the same atom (that is, in the head and in the body, negated or not; or in the body — both positively and negatively). We call such programs *simple*. It is well known that the problem of the existence of a stable model of a normal logic program is NP-complete (Marek and Truszczyński 1991), and of the disjunctive logic program — Σ_2^P-complete (Eiter and Gottlob 1995). The proofs provided by Marek and Truszczyński (1991) and Eiter and Gottlob (1995) imply that the results hold also under the restriction to simple normal and simple disjunctive programs, respectively (in the case of disjunctive logic programs, a minor modification of the construction is required).

Let ρ be a logic programming rule, say

$$\rho = a_1 | \ldots | a_k \leftarrow \beta \cdot$$

We define

$$aic(\rho) = not\ a_1, \ldots, not\ a_k, \beta \supset +a_1 | \ldots | + a_k \cdot$$

We extend the operator $aic(\cdot)$ to logic programs in a standard way. We note that if a rule ρ is simple, then $body(aic(\rho))$ is consistent and $nup(aic(\rho)) = body(\rho)$.

We recall that a set M of atoms is an answer set of a disjunctive logic program P if M is a minimal set closed under the reduct P^M, where P^M consists of the rules obtained by dropping all negative literals from those rules in P that do not contain a literal *not* a in the body, for any $a \in M$ (we refer to the paper by Gelfond and Lifschitz (1991) for details). Our first two lemmas establish a result needed for hardness arguments.

Lemma 1
Let P be a simple disjunctive logic program and M', M sets of atoms such that $M' \subseteq M$. Then M' is a model of P^M if and only if $\{+a \mid a \in M'\} \cup \{-a \mid a \notin M\}$ is closed under $aic(P)$.

Proof: Let us define $\mathcal{U} = \{+a \mid a \in M'\} \cup \{-a \mid a \notin M\}$. We note that $\mathcal{U}$ is consistent.

(⇒) Let $r \in aic(P)$, $\rho \in P$ be a rule such that $r = aic(\rho)$, and ρ' be the rule obtained by eliminating from ρ all negative literals.

Since P is simple, $nup(r) = body(\rho)$. Let us assume that $nup(r) \subseteq \mathcal{U}$. It follows that $\rho' \in P^M$ and that $M' \models body(\rho')$. Thus, $head(\rho') \cap M' \neq \emptyset$. Since $head(\rho) = head(\rho')$ and $head(r) = head(aic(\rho)) = ua(head(\rho))$, $head(r) \cap \mathcal{U} \neq \emptyset$. That is, $\mathcal{U}$ is closed under r and, since r was chosen arbitrarily, under $aic(P)$, too.

(⇐) Let us consider $\rho' \in P^M$. There is $\rho \in P$ such that for every negative literal $not\, a \in body(\rho)$, $a \notin M$, and dropping all negative literals from ρ results in ρ'. If $body(\rho') \subseteq M'$, then $body(\rho) \subseteq lit(\mathcal{U})$. Thus, $nup(aic(\rho)) \subseteq \mathcal{U}$. It follows that $head(aic(\rho)) \cap \mathcal{U} \neq \emptyset$. Thus, $head(\rho) \cap lit(\mathcal{U}) \neq \emptyset$. Since $head(\rho)$ consists of atoms and $head(\rho') = head(\rho)$, $head(\rho') \cap M' \neq \emptyset$. That is, $M' \models \rho'$ and, consequently, $M' \models P^M$. □

Theorem 6

Let P be a simple disjunctive logic program. A set M of atoms is an answer set of P if and only if $ua(M)$ is a justified weak repair for $\langle \emptyset, aic(P) \rangle$.

Proof: (⇒) Let M be an answer set of P. That is, M is a minimal set closed under the rules in the reduct P^M. By Lemma 1, $\{+a \mid a \in M\} \cup \{-a \mid a \notin M\}$ is closed under $aic(P)$. Let $\mathcal{U}'$ be a set of update actions such that $\{-a \mid a \notin M\} \subseteq \mathcal{U}' \subseteq \{+a \mid a \in M\} \cup \{-a \mid a \notin M\}$. We define $M' = \{a \mid +a \in \mathcal{U}'\}$. Then $M' \subseteq M$. By Lemma 1, $M' \models P^M$. Since M is an answer set of P, $M' = M$ and $\mathcal{U}' = \mathcal{U}$. It follows that $\{+a \mid a \in M\} \cup \{-a \mid a \notin M\}$ is a minimal set closed under $aic(P)$ and containing $\{-a \mid a \notin M\}$. Since $ua(M) = \{+a \mid a \in M\}$ and $ne(\emptyset, \emptyset \circ ua(M)) = \{-a \mid a \notin M\}$, Theorem 1 implies that $ua(M)$ is justified weak repair for $\langle \emptyset, aic(P) \rangle$.

(⇐) By Theorem 1, $\{+a \mid a \in M\} \cup \{-a \mid a \notin M\}$ is a minimal set containing $\{-a \mid a \notin M\}$ and closed under $aic(P)$. By Lemma 1, M is a model of P^M. Let $M' \subseteq M$ be a model of P^M. Again by Lemma 1, $\{+a \mid a \in M'\} \cup \{-a \mid a \notin M\}$ is closed under $aic(P)$. It follows that $\{+a \mid a \in M'\} \cup \{-a \mid a \notin M\} = \{+a \mid a \in M\} \cup \{-a \mid a \notin M\}$. Thus, $M' = M$ and so, M is a minimal model of P^M, that is, an answer set of P. □

We now move on to results concerning upper bounds (membership) and derive the main results of this section.

Lemma 2

Let η be a finite set of normal active integrity constraints and let $\mathcal{U}$ be a finite set of update actions. There is the least set of update actions $\mathcal{W}$ such that $\mathcal{U} \subseteq \mathcal{W}$ and $\mathcal{W}$ is closed under η. Moreover, this least set $\mathcal{W}$ can be computed in polynomial time in the size of η and $\mathcal{U}$.

Proof: We prove the result by demonstrating a bottom-up process computing $\mathcal{W}$. The process is similar to that applied when computing a least model of a Horn program. We start with $\mathcal{W}_0 = \mathcal{U}$, Assuming that $\mathcal{W}_i$ has been computed, we identify in η every active integrity constraint r such that $nup(r) \subseteq lit(\mathcal{W}_i)$, and add the head of each such rule r to $\mathcal{W}_i$. We call the result $\mathcal{W}_{i+1}$. If $\mathcal{W}_{i+1} = \mathcal{W}_i$, we stop. It is straightforward to prove that the last set constructed in the process is closed under η, contains $\mathcal{U}$, and is contained in every set that is closed under η and contains $\mathcal{U}$. Moreover, the construction can be implemented to run in polynomial time. □

Theorem 7
Let $\mathcal{I}$ be a database and η a set of normal active integrity constraints. Then checking if there exists a justified repair (justified weak repair, respectively) for $\langle\mathcal{I},\eta\rangle$ is an NP-complete problem.

Proof: By Theorem 4, it is enough to prove the result for justified weak repairs.

(Membership) The following algorithm decides the problem: (1) Nondeterministically guess a consistent set of update actions $\mathcal{E}$. (2) Compute $ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$. (3) If $\mathcal{E}\cap ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})\neq\emptyset$ return NO. Otherwise, compute the least set $\mathcal{W}$ of update actions that is closed under η and contains $ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$. (4) If $\mathcal{W}=\mathcal{E}\cup ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$, then return YES. Otherwise, return NO. From Lemma 2, it follows that the algorithm runs in polynomial time. From Theorem 1, it follows that the algorithm is correct.

(Hardness) The problem of the existence of an answer set of a simple normal logic program P is NP-complete. By Theorem 4 and Theorem 6, P has an answer set if and only if there exists a justified weak repair for $\langle\emptyset, aic(P)\rangle$. Since $aic(P)$ can be constructed in polynomial time in the size of P, the result follows. □

Lemma 3
Let η be a finite set of active integrity constraints and let $\mathcal{U}'$ and $\mathcal{U}''$ be sets of update actions. The problem whether there is a set $\mathcal{U}$ of update actions such that $\mathcal{U}$ is closed under η and $\mathcal{U}'\subseteq\mathcal{U}\subset\mathcal{U}''$ is in NP.

Proof: Once we nondeterministically guess $\mathcal{U}$, checking all the required conditions can be implemented in polynomial time. □

Lemma 4
Let η be a finite set of active integrity constraints, $\mathcal{I}$ a database, and $\mathcal{E}$ be a set of update actions. The problem whether there is a set $\mathcal{E}'\subset\mathcal{E}$ of update actions such that $\mathcal{I}\circ\mathcal{E}'\models\eta$ is in NP.

Proof: Once we nondeterministically guess $\mathcal{E}$, checking all the required conditions can be implemented in polynomial time. □

Theorem 8
Let $\mathcal{I}$ be a database and η a set of active integrity constraints. The problem of the existence of a justified weak repair for $\langle\mathcal{I},\eta\rangle$ is a Σ_2^P-complete problem.

Proof: (Membership) The problem can be decided by a nondeterministic polynomial-time Turing Machine with an NP-oracle. Indeed, in the first step, one needs to guess (nondeterministically) a consistent set $\mathcal{E}$ of update actions. Setting $\mathcal{U}=\mathcal{E}\cup ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})$, one needs to verify that

1. $\mathcal{E}\cap ne(\mathcal{I},\mathcal{I}\circ\mathcal{E})=\emptyset$
2. $\mathcal{U}$ is closed under η

3. for each $\mathcal{U}'$ such that $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) \subseteq \mathcal{U}' \subseteq \mathcal{U}$ and $\mathcal{U}'$ closed under η, $\mathcal{U}' = \mathcal{U}$ (by Lemma 3, one call to an NP-oracle suffices).

(Hardness) The problem of the existence of an answer set of a simple disjunctive logic program P is Σ_2^P-complete. By Theorem 6, P has an answer set if and only if there exists a justified weak repair for $\langle \emptyset, aic(P) \rangle$. Thus, the result follows. □

Theorem 9
Let $\mathcal{I}$ be a database and η a set of active integrity constraints. The problem of the existence of a justified repair for $\langle \mathcal{I}, \eta \rangle$ is a Σ_2^P-complete problem.

Proof: (Membership) The problem can be decided by a nondeterministic polynomial-time Turing Machine with an NP-oracle. Indeed, in the first step, one needs to guess (nondeterministically) a consistent set $\mathcal{E}$ of update actions. Setting $\mathcal{U} = \mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$, one needs to verify that

1. $\mathcal{E} \cap ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) = \emptyset$
2. $\mathcal{U}$ is closed under η
3. for each $\mathcal{U}'$ such that $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) \subseteq \mathcal{U}' \subseteq \mathcal{U}$ and $\mathcal{U}'$ closed under η, $\mathcal{U}' = \mathcal{U}$ (by Lemma 3, one call to an NP-oracle suffices)
4. for each $\mathcal{E}'$ such that $\mathcal{E}' \subset \mathcal{E}$, $\mathcal{I} \circ \mathcal{E}' \not\models \eta$ (By Lemma 4, one call to an NP-oracle suffices).

(Hardness) Since for the class of instances $\langle \emptyset, aic(P) \rangle$ justified weak repairs coincide with justified repairs (Theorem 3), the result follows. □

8 Some implications of the results obtained so far

We recall that given a database $\mathcal{I}$ and a set η of aic's, the goal is to replace $\mathcal{I}$ with $\mathcal{I}'$ so that $\mathcal{I}'$ satisfies η. The set of update actions needed to transform $\mathcal{I}$ into $\mathcal{I}'$ must at least be a repair for $\langle \mathcal{I}, \eta \rangle$ (assuming we insist on change-minimality, which normally is the case). However, it should also obey preferences captured by the heads of constraints in η. Let us denote by $\mathbf{R}(\mathcal{I}, \eta)$, $\mathbf{WR}(\mathcal{I}, \eta)$, $\mathbf{FR}(\mathcal{I}, \eta)$, $\mathbf{FWR}(\mathcal{I}, \eta)$, $\mathbf{JR}(\mathcal{I}, \eta)$, and $\mathbf{JWR}(\mathcal{I}, \eta)$ the classes of repairs, weak repairs, founded repairs, founded weak repairs, justified repairs and justified weak repairs for $\langle \mathcal{I}, \eta \rangle$, respectively. Figure 1 shows the relationships among these classes, with all inclusions being in general proper. Under each class we also give the complexity of deciding whether a repair from that class exists.

Thus, given an instance $\langle \mathcal{I}, \eta \rangle$ of the database repair problem, one might first attempt to select a repair for $\langle \mathcal{I}, \eta \rangle$ from the most restricted set of repairs, $\mathbf{JR}(\mathcal{I}, \eta^n)$. Not only these repairs are strongly tied to preferences expressed by η — the related computational problems are relatively easy. The problem to decide whether $\mathbf{JR}(\mathcal{I}, \eta^n)$ is empty is NP-complete. However, the class $\mathbf{JR}(\mathcal{I}, \eta^n)$ is narrow and it may be that $\mathbf{JR}(\mathcal{I}, \eta^n) = \emptyset$. If it is so, the next step might be to try to repair $\mathcal{I}$ by selecting a repair from $\mathbf{JR}(\mathcal{I}, \eta)$. This class of repairs for $\langle \mathcal{I}, \eta \rangle$ reflects the preferences captured by η. Since it is broader than the previous one, there is a better possibility it will be non-empty. However, the computational complexity grows

				$\mathbf{FR}(\mathcal{I}, \eta^n)$ [Σ^2_P-c]				
				$\parallel$				
$\mathbf{JR}(\mathcal{I}, \eta^n)$ [NP-c]	$\subseteq$	$\mathbf{JR}(\mathcal{I}, \eta)$ [Σ^2_P-c]	$\subseteq$	$\mathbf{FR}(\mathcal{I}, \eta)$ [Σ^2_P-c]	$\subseteq$	$\mathbf{R}(\mathcal{I}, \eta)$ [NP-c]	$=$	$\mathbf{R}(\mathcal{I}, \eta^n)$ [NP-c]
$\parallel$		$\cap\!\!\mid$		$\cap\!\!\mid$		$\cap\!\!\mid$		$\cap\!\!\mid$
$\mathbf{JWR}(\mathcal{I}, \eta^n)$ [NP-c]	$\subseteq$	$\mathbf{JWR}(\mathcal{I}, \eta)$ [Σ^2_P-c]	$\subseteq$	$\mathbf{FWR}(\mathcal{I}, \eta)$ [NP-c]	$\subseteq$	$\mathbf{WR}(\mathcal{I}, \eta)$ [NP-c]	$=$	$\mathbf{WR}(\mathcal{I}, \eta^n)$ [NP-c]
				$\parallel$				
				$\mathbf{FWR}(\mathcal{I}, \eta^n)$ [NP-c]				

Fig. 1. Relationships among classes of repairs

— the existence problem for $\mathbf{JR}(\mathcal{I}, \eta)$ is Σ^2_P-complete. If also $\mathbf{JR}(\mathcal{I}, \eta) = \emptyset$, it still may be that founded repairs exist. Moreover, deciding whether a founded repair exists is not harder than the previous step. Finally, if there are no founded repairs, one still may consider just a repair. This is not quite satisfactory as it ignores the preferences encoded by η and concentrates only on the constraint enforcement. However, deciding whether a repair exists is "only" NP-complete. Moreover, this class subsumes all other classes of repairs and offers the best chance of success.

We note that if we fail to find a justified or founded repair in the process described above, we may decide that respecting preferences encoded in aic's is more important than the minimality of change postulate. In such case, rather to proceed to seek a repair, as discussed above, we also have an option to consider justified weak repairs of $\langle \mathcal{I}, \eta \rangle$, where the existence problem is Σ^P_2-complete and, then founded weak repairs for $\langle \mathcal{I}, \eta \rangle$, where the existence problem is NP-complete.

Finally, we point out that when we choose a smaller class $\mathcal{A}$ of repairs (e.g., $\mathbf{JR}(\mathcal{I}, \eta^n)$) instead of a more general one $\mathcal{A}'$ (e.g., $\mathbf{JR}(\mathcal{I}, \eta)$) we observe an important (and desirable) effect on *consistent query answering*. In consistent query answering (*conservative reasoning*) an atom a is *true* if it belongs to every repaired database, *false* if it does not belong to any repaired database, and *unknown* otherwise, that is, when it belongs to a *proper* subset of the set of repaired databases. It is clear that if $\mathcal{A} \subseteq \mathcal{A}'$, the set of the true atoms under $\mathcal{A}'$ is a subset of the set of the true atoms obtained by applying $\mathcal{A}$, and the same holds for atoms that are false. In other words, the stronger the semantics, the larger the set of atoms that receive a definite truth value (are true or false).

9 Revision Programming — an Overview

We review the basic terminology of revision programming, and recall the two semantics introduced by Marek, Truszczyński and Pivkina (1994; 1998; 2001): the semantics of supported revisions, and the semantics of justified weak revisions (originally referred to as justified revisions and renamed here for consistency with the general naming schema we use).

Revision literals. A *revision literal* is an expression $\mathbf{in}(a)$ or $\mathbf{out}(a)$, where $a \in At$.

Revision literals $\mathbf{in}(a)$ and $\mathbf{out}(a)$ are *duals* of each other. If α is a revision literal, we denote its dual by α^D. We extend this notation to sets of revision literals. We say that a set of revision literals is *consistent* if it does not contain a pair of dual literals. Revision literals represent elementary updates one can apply to a database. We define the result of applying a *consistent* set $\mathcal{U}$ of revision literals to a database $\mathcal{I}$ as follows:

$$\mathcal{I} \oplus \mathcal{U} = (\mathcal{I} \cup \{a \mid \mathbf{in}(a) \in \mathcal{U}\}) \setminus \{a \mid \mathbf{out}(a) \in \mathcal{U}\}\cdot$$

Revision rules, normal rules and constraints. A *revision rule* is an expression of the form

$$r = \alpha_1 | \dots | \alpha_k \leftarrow \beta_1, \dots, \beta_m, \tag{5}$$

where $k, m \geq 0$, $k + m \geq 1$, and α_i and β_j are revision literals. The set $\{\alpha_1, \dots, \alpha_k\}$ is the *head* of the rule (5); we denote it by $head(r)$. Similarly, the set $\{\beta_1, \dots, \beta_m\}$ is the *body* of the rule (5); we denote it by $body(r)$. A revision rule is *normal* if $|head(r)| \leq 1$. As in the case of active integrity constraints, we denote the empty head as $\bot$. We call rules with the empty head *constraints*. If $|body(r)| = 0$ we omit the implication symbol. Examples of revision rules are: (1) $\mathbf{in}(a)|\mathbf{out}(b) \leftarrow \mathbf{in}(c)$, (2) $\mathbf{in}(a)|\mathbf{in}(c)$, (3) $\mathbf{in}(a) \leftarrow \mathbf{out}(b)$, and (4) $\bot \leftarrow \mathbf{in}(a), \mathbf{out}(b)$. The second rule is an example of a rule with the empty body, the third one is an example of a normal rule and the last one is an example of a constraint. The informal reading of a revision rule, say the first rule given above, $\mathbf{in}(a)|\mathbf{out}(b) \leftarrow \mathbf{in}(c)$, is: *insert a or delete b, if c is present.*

Revision programs. A *revision program* is a collection of revision rules. A revision program is *normal* if all its rules are normal.

Entailment (satisfaction). A database $\mathcal{I}$ *satisfies* a revision literal $\mathbf{in}(a)$ ($\mathbf{out}(b)$, respectively), if $a \in \mathcal{I}$ ($b \notin \mathcal{I}$, respectively). A database $\mathcal{I}$ *satisfies* a revision rule (5) if it satisfies at least one literal α_i, $1 \leq i \leq k$, whenever it satisfies every literal β_j, $1 \leq j \leq m$. Finally, a database $\mathcal{I}$ satisfies a revision program P, if $\mathcal{I}$ satisfies every rule in P. We use the symbol $\models$ to denote the satisfaction relation.

For revision literals $\alpha = \mathbf{in}(a)$ and $\beta = \mathbf{out}(b)$, we set $lit(\alpha) = a$ and $lit(\beta) = not\, b$. We extend this notation to sets of revision literals. We note that every database interprets revision literals and the corresponding propositional literals in the same way. That is, for every database $\mathcal{I}$ and for every set of revision literals L, $\mathcal{I} \models L$ if and only if $\mathcal{I} \models lit(L)$.

It follows that a revision rule (5) specifies an integrity constraint equivalent to the propositional formula: $lit(\beta_1), \dots, lit(\beta_m) \supset lit(\alpha_1), \dots, lit(\alpha_k)$. However, a revision rule is not only an integrity constraint. Through its syntax, it also encodes a preference on how to "fix" a database, when it violates the constraint. Not satisfying a revision rule r means satisfying all revision literals in the body of r *and* not satisfying any of the revision literals in the head of r. Thus, enforcing the constraint means constructing a database that (1) does not satisfy some revision literal in the body of r, *or* (2) satisfies at least one revision literal in the head of r. The underlying idea of revision programming is to prefer those revisions that result in databases with the property (2).

As an example, let us consider the revision rule $r = \mathbf{in}(a) \leftarrow \mathbf{out}(b)$, and the empty database $\mathcal{I}$. Clearly, $\mathcal{I}$ does not satisfy r. Although $\mathcal{I}$ can be fixed either by inserting a, so that $head(r)$ becomes *true*, or by inserting b, so that $body(r)$ becomes *false*, the syntax of r makes the former preferred.

Normal revision programs were introduced and studied by Marek and Truszczyński (1994; 1998), who proposed the syntax and the semantics of supported and justified weak revisions. The formalism was extended by Pivkina (2001) to allow disjunctions of revision literals in the heads of rules, and the semantics of justified weak revisions was generalized to that case. We will now recall these definitions.

First, we define the notion of the *inertia set*. Let $\mathcal{I}$ and $\mathcal{R}$ be databases. We define the *inertia set* wrt $\mathcal{I}$ and $\mathcal{R}$, denoted $I(\mathcal{I}, \mathcal{R})$, by setting

$$I(\mathcal{I}, \mathcal{R}) = \{\mathbf{in}(a) \mid a \in \mathcal{I} \cap \mathcal{R}\} \cup \{\mathbf{out}(a) \mid a \notin \mathcal{I} \cup \mathcal{R}\}\cdot$$

In other words, $I(\mathcal{I}, \mathcal{R})$ is the set of all *no-effect* revision literals for $\mathcal{I}$ and $\mathcal{R}$, that is, revision literals that have no effect when revising $\mathcal{I}$ into $\mathcal{R}$.

Now, let P be a *normal* revision program and $\mathcal{R}$ be a database. By $P_{\mathcal{R}}$ we denote the program obtained from P by removing each rule $r \in P$ such that $\mathcal{R} \not\models body(r)$.

Definition 5 (Supported updates and supported revisions)
Let P be a normal revision program and $\mathcal{I}$ a database. A set $\mathcal{U}$ of revision literals is a *supported update* of $\mathcal{I}$ wrt P if $\mathcal{U}$ is consistent and $\mathcal{U} = head(P_{\mathcal{I} \oplus \mathcal{U}})$. A set $\mathcal{E}$ is a *supported revision* of $\mathcal{I}$ wrt P if $\mathcal{E} = \mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$, where $\mathcal{U}$ is a supported update. □

Intuitively, a consistent set $\mathcal{U}$ of revision literals is a supported update if it is precisely the set of literals "supported" by P and the database resulting from updating $\mathcal{I}$ with $\mathcal{U}$. Eliminating from a supported revision all no-effect literals yields a supported revision.

While not evident explicitly from the definition, supported updates and revisions guarantee constraint enforcement, as proved by Marek and Truszczyński (1998).

Proposition 3
Let P be a normal revision program and $\mathcal{I}$ a database. If $\mathcal{E}$ is a supported revision of P, then $\mathcal{I} \oplus \mathcal{E} \models P$. □

Supported updates do not take into account the inertia set. Supported revisions do, but only superficially: simply removing no-effect literals from the corresponding supported update. It is then not surprising that supported updates and revisions may be self-grounded and non-minimal, as we show in the following example.

Example 8
Let P be a revision program containing the rules $\{\mathbf{in}(a) \leftarrow \mathbf{in}(b),\ \mathbf{in}(b) \leftarrow \mathbf{in}(a),$ $\mathbf{in}(c) \leftarrow \mathbf{out}(d)\}$, and let $\mathcal{I}$ the empty database. $\mathcal{I}$ does not satisfy P as it violates the rule $\mathbf{in}(c) \leftarrow \mathbf{out}(d)$. One can check that set $\mathcal{U} = \{\mathbf{in}(a), \mathbf{in}(b), \mathbf{in}(c)\}$ modeling the insertions of a, b and c, is a supported update and a supported revision. However it is not minimal as its subset $\{\mathbf{in}(c)\}$ is sufficient to guarantee the satisfaction of P. □

The problem in the previous example is self-groundedness or the circularity of support between $\mathbf{in}(a)$ and $\mathbf{in}(b)$. Each of them supports the other one but the set containing both is superfluous. To address the problem, Marek and Truszczyński (1994; 1998) proposed for normal revision programs the semantics of justified weak revisions, later extended to the disjunctive case by Pivkina (2001). The idea was to "ground" justified weak revisions in the program and the inertia set by means of a *minimal closure*.

Definition 6 (Minimal closed sets of revision literals)
A set $\mathcal{U}$ of revision literals is *closed* under a revision program P (not necessarily normal) if for every rule $r \in P$, whenever $body(r) \subseteq \mathcal{U}$, then $head(r) \cap \mathcal{U} \neq \emptyset$. If $\mathcal{U}$ is closed under P and for every set $\mathcal{U}' \subseteq \mathcal{U}$ closed under P, we have $\mathcal{U}' = \mathcal{U}$, then $\mathcal{U}$ is a *minimal closed* set for P. □

With this definition in hand, we can define the concepts of justified updates and justified weak revisions.

Definition 7 (Justified updates and justified weak revisions)
Let P be a revision program and let $\mathcal{I}$ be a database. A consistent set $\mathcal{U}$ of revision literals is a *P-justified update* for $\mathcal{I}$ if it is a minimal set closed under $P \cup I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$.

If $\mathcal{U}$ is a P-justified update for $\mathcal{I}$, then $\mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$ is a *P-justified weak revision* for $\mathcal{I}$. □

We note that $P \cup I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$ is well defined as revision literals (and so, in particular, the revision literals in $I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$) are special revision rules (normal and with empty bodies).

The inertia set plays an essential role in the definition, as it is used directly in the definition of a P-justified update. Again, it is not self-evident from the definition that justified updates and justified weak revisions, when applied to an initial database yield a database satisfying the program. However, the definition does indeed imply so (Marek and Truszczyński 1998; Pivkina 2001).

Proposition 4
Let P be a revision program and $\mathcal{I}$ a database. If $\mathcal{U}$ is a justified update or justified weak revision of P, then $\mathcal{I} \oplus \mathcal{U} \models P$. □

We point out that the original term for the *justified weak revisions* was *justified revisions* (Marek and Truszczyński 1998). We changed the name for consistency with the naming schema we used for active integrity constraints.

10 A Family of Declarative Semantics for Revision Programming

The two semantics in the previous section were defined based on how revisions are "grounded" in a program, an initial database, and the inertia set. The fundamental postulates of constraint enforcement and minimality of change played no explicit role in those considerations. The first one is no problem as it is a side effect of each of the two types of groundedness considered (cf. Propositions 3 and 4). The second one does not hold for supported revisions. And while Marek and Truszczyński

(1998) proved that justified weak revisions are change-minimal in the case of *normal* revision programs, it is not so in the general case.

Example 9
Let P be a revision program consisting of the rules $\mathbf{in}(a)|\mathbf{out}(b)$, $\mathbf{out}(a)|\mathbf{in}(b)$, and let $\mathcal{I}$ be the empty database. It is easy to verify that set $\{\mathbf{in}(a), \mathbf{in}(b)\}$ is a justified weak revision. However, it is not minimal as $\mathcal{I}$ is already consistent and no update is needed (or, in other words, the empty update fixes the consistency). □

We will now develop a range of semantics for revision programs by taking the postulates of constraint enforcement and minimality of change explicitly into consideration.

Definition 8 (Weak Revisions and Revisions)
A consistent set $\mathcal{U}$ of revision literals is a *weak revision* of $\mathcal{I}$ wrt a revision program P if (1) $\mathcal{U} \cap I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U}) = \emptyset$ (relevance — all revision literals in $\mathcal{U}$ actually change $\mathcal{I}$ or, in other words, none of them is a no-effect literal wrt $\mathcal{I}$ and $\mathcal{I} \oplus \mathcal{U}$); and (2) $\mathcal{I} \oplus \mathcal{U} \models P$ (constraint enforcement). Further, $\mathcal{U}$ is a *revision* of $\mathcal{I}$ with respect to a revision program P if it is a weak revision and for every $\mathcal{U}' \subseteq \mathcal{U}$, $\mathcal{I} \oplus \mathcal{U}' \models P$ implies that $\mathcal{U}' = \mathcal{U}$ (minimality of change). □

Example 10
Let P be the program consisting of the two rules from Example 9 and the rule $\mathbf{in}(c) \leftarrow \mathbf{out}(d)$. As before, let $\mathcal{I} = \emptyset$. There are several weak revisions of $\mathcal{I}$ with respect to P, for instance, $U_1 = \{\mathbf{in}(d)\}$, $\mathcal{U}_2 = \{\mathbf{in}(d), \mathbf{in}(a), \mathbf{in}(b)\}$, $\mathcal{U}_3 = \{\mathbf{in}(c)\}$, and $\mathcal{U}_4 = \{\mathbf{in}(c), \mathbf{in}(a), \mathbf{in}(b)\}$. The weak revisions $\mathcal{U}_1$ and $\mathcal{U}_3$ are minimal and so, they are revisions. □

(Weak) revisions do not reflect the preferences on how to revise a database encoded in the syntax of revision rules. Justified weak revisions and supported revisions, which we discussed in the previous section, do.

Example 10 (continued)
Both the semantics of supported revisions and justified weak revisions exclude the weak revisions $\mathcal{U}_1 = \{\mathbf{in}(d)\}$ and $\mathcal{U}_2 = \{\mathbf{in}(d), \mathbf{in}(a), \mathbf{in}(b)\}$, in favor of $\mathcal{U}_3 = \{\mathbf{in}(c)\}$ and $\mathcal{U}_4 = \{\mathbf{in}(c), \mathbf{in}(a), \mathbf{in}(b)\}$ ($\mathcal{U}_3$ and $\mathcal{U}_4$ indeed are supported and justified weak revisions), thus *preferring* to satisfy the head of the rule $\mathbf{in}(c) \leftarrow \mathbf{out}(d)$ rather than to violate its the body. Indeed, one can check that $\mathcal{U}_3$ and $\mathcal{U}_4$ are indeed both supported and justified weak revisions, while $\mathcal{U}_1$ and $\mathcal{U}_2$ are neither. □

We will now introduce several additional semantics that aim to capture this preference. First, we define a new semantics for revision programs by strengthening the semantics of justified weak revisions. We do so simply by imposing change-minimality explicitly.

Definition 9 (Justified Revisions)
Let P be a revision program and let $\mathcal{I}$ be a database. A P-justified weak revision $\mathcal{E}$ for $\mathcal{I}$ is a *P-justified revision* for $\mathcal{I}$ if $\mathcal{E}$ is a revision of $\mathcal{I}$ wrt P (that is, for every set $\mathcal{E}' \subseteq \mathcal{E}$ such that $\mathcal{I} \oplus \mathcal{E}' \models P$, $\mathcal{E}' = \mathcal{E}$). □

Example 10 (continued)

Let us consider again Example 10. The set $\mathcal{U}_3$ is a P-justified revision for $\mathcal{I}$, while $\mathcal{U}_4$ is not, reflecting the fact that we require that P-justified revisions be revisions (that is, satisfy change minimality). □

Justified revisions have several useful properties. They are change-minimal and are grounded in the program *and* the inertia set. However, as stable models of logic programs, to which they are closely related, in some settings they may be too restrictive.

Example 11

Let $P = \{\mathbf{in}(a) \leftarrow \mathbf{in}(a),\ \mathbf{in}(a) \leftarrow \mathbf{out}(a)\}$ and let $\mathcal{I} = \emptyset$. Clearly, $\mathcal{I}$ is inconsistent with respect to P. The set $\mathcal{U} = \{\mathbf{in}(a)\}$ is a revision of $\mathcal{I}$ and one might argue that P provides it a justification: the two rules together "force" a into $\mathcal{I}$, as in any particular situation one of them applies and provides a justification for $\mathbf{in}(a)$. This type of an argument is known as "reasoning by cases." However, one can check that $\mathcal{U}$ is not a P-justified revision of $\mathcal{I}$ and not a P-justified weak revision, either. Thus, justified (weak) revisions in general exclude such reasonings as valid. □

To provide a semantics capturing such justifications, we introduce now the concept of foundedness and the semantics of founded (weak) revisions. We follow closely intuitions behind founded (weak) repairs.

Definition 10 (Founded (weak) revisions)

Let $\mathcal{I}$ be a database, P a revision program and, and $\mathcal{E}$ a consistent set of revision literals.

1. A revision literal α is *P-founded* wrt $\mathcal{I}$ and $\mathcal{E}$ if there is $r \in P$ such that $\alpha \in head(r)$, $\mathcal{I} \oplus \mathcal{E} \models body(r)$, and $\mathcal{I} \oplus \mathcal{E} \models \beta^D$, for every $\beta \in head(r) \setminus \{\alpha\}$.
2. The set $\mathcal{E}$ is *P-founded* wrt $\mathcal{I}$ if every element of $\mathcal{E}$ is P-founded wrt $\mathcal{I}$ and $\mathcal{E}$.
3. $\mathcal{E}$ is a *P-founded (weak) revision* for $\mathcal{I}$ if $\mathcal{E}$ is a (weak) revision of $\mathcal{I}$ wrt P and $\mathcal{E}$ is *P-founded* wrt $\mathcal{I}$. □

It is clear from the definition that P-foundedness of a revision literal α with respect to a consistent set of revision literals $\mathcal{E}$ can be established by considering rules in P independently of each other, which supports reasoning by cases such as the one used in Example 11 (in this specific case, $\mathbf{in}(a)$ is founded either because of the first rule or becaue of the second rule). Indeed, one can verify that the revision $\mathcal{U}$ in Example 11 is founded.

We note that condition (3) of the definition guarantees that founded (weak) revisions enforce constraints of the revision program. Next, directly from the definition, it follows that founded weak revisions are weak revisions. Similarly, founded revisions are revisions and so, they are change-minimal. Furthermore, founded revisions are founded weak revisions. However, there are (weak) revisions that are not founded, and founded weak revisions are not necessarily founded revisions, that is, are not change-minimal. The latter observation shows that foundedness is too weak a condition to guarantee change-minimality.

Example 12
Let P be the revision program containing the rules $\{\mathbf{in}(b) \leftarrow \mathbf{in}(a),\ \mathbf{in}(a) \leftarrow \mathbf{in}(b), \mathbf{in}(c) \leftarrow \mathbf{out}(d)\}$ and $\mathcal{I}$ the empty database. The set $\{\mathbf{in}(d)\}$ is a revision of $\mathcal{I}$ wrt P. Therefore it is a weak revision of $\mathcal{I}$ wrt P. However, it is not a P-founded weak revision for $\mathcal{I}$. Therefore, it is not a P-founded revision for $\mathcal{I}$, either. The set $\{\mathbf{in}(c), \mathbf{in}(a), \mathbf{in}(b)\}$ is a P-founded weak revision for $\mathcal{I}$ but not a P-founded revision for $\mathcal{I}$. Indeed, $\{\mathbf{in}(c)\}$ is also a revision of $\mathcal{I}$ wrt P. □

In the case of normal revision programs, founded weak revisions coincide with supported revisions.

Theorem 10
Let P be a normal revision program and $\mathcal{I}$ a database. A set $\mathcal{E}$ of revision literals is a P-founded weak revision of $\mathcal{I}$ if and only if $\mathcal{E}$ is a P-supported revision of $\mathcal{I}$. □

Proof:($\Rightarrow$) Let $\mathcal{E}$ be a P-founded weak revision of $\mathcal{I}$ and let $\mathcal{U} = \mathcal{E} \cup (I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E}) \cap head(P_{\mathcal{I}\oplus\mathcal{E}}))$. As $\mathcal{E}$ is a weak revision of $\mathcal{I}$ with respect to P, $\mathcal{E} \cap I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E}) = \emptyset$. Therefore, $\mathcal{E} = \mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E})$ and $\mathcal{I} \oplus \mathcal{E} = \mathcal{I} \oplus \mathcal{U}$. It follows that $\mathcal{E} = \mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$ and so, it will suffice to prove that $\mathcal{U}$ is a supported update of $\mathcal{I}$ with respect to P.

To this end, we first note that $\mathcal{U}$ is consistent. Indeed:

1. $\mathcal{E}$ is consistent (it is a weak revision);
2. $I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E})$ is consistent;
3. If $\alpha \in \mathcal{E}$ then the literal $\alpha^D \notin I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E})$.

Next, we prove that $\mathcal{U} = head(P_{\mathcal{I}\oplus\mathcal{U}})$. Let $\alpha \in \mathcal{U}$. We have two cases: either $\alpha \in I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E}) \cap head(P_{\mathcal{I}\oplus\mathcal{E}})$ or $\alpha \in \mathcal{E}$. The first case trivially verifies the assertion. In the second case, as $\mathcal{E}$ is a P-founded weak revision of $\mathcal{I}$, there exists $r \in P$ such that $\alpha = head(r)$ and $\mathcal{I} \oplus \mathcal{E} \models body(r)$ (cf. Definition 5). Thus, $r \in P_{\mathcal{I}\oplus\mathcal{E}}$ and $\alpha \in head(P_{\mathcal{I}\oplus\mathcal{E}})$. As $\mathcal{I} \oplus \mathcal{E} = \mathcal{I} \oplus \mathcal{U}$ we have $\alpha \in head(P_{\mathcal{I}\oplus\mathcal{U}})$.

Conversely, let $\alpha \in head(P_{\mathcal{I}\oplus\mathcal{U}})$. We have two cases: $\alpha \in I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E})$, and $\alpha \notin I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E})$. In the first case, $\alpha \in \mathcal{U}$ (by the definition of $\mathcal{U}$). In the second case, we reason as follows. Since $\alpha \in head(P_{\mathcal{I}\oplus\mathcal{U}})$, there exists $r \in P$ such that $\alpha = head(r)$ and $\mathcal{I} \oplus \mathcal{U} \models body(r)$. Thus, $\mathcal{I} \oplus \mathcal{E} \models body(r)$. As $\mathcal{E}$ is a weak revision, $\mathcal{I} \oplus \mathcal{E} \models r$. Consequently, $\mathcal{I} \oplus \mathcal{E} \models \alpha$. Since $\alpha \notin I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E})$, $\mathcal{I} \not\models \alpha$ and so, $\alpha \in \mathcal{E}$. Thus, $\alpha \in \mathcal{U}$.

($\Leftarrow$) Let $\mathcal{E}$ be a P-supported revision of $\mathcal{I}$. It follows that $\mathcal{E} = \mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$, where $\mathcal{U}$ is a P-supported update of $\mathcal{I}$ wrt P. It follows that $\mathcal{I} \oplus \mathcal{E} = \mathcal{I} \oplus \mathcal{U}$. Consequently, $\mathcal{E} \cap I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E}) = \emptyset$ and, by Proposition 3, $\mathcal{I} \oplus \mathcal{E} \models P$. Since $\mathcal{E} \subseteq \mathcal{U}$, $\mathcal{E}$ is consistent and so, $\mathcal{E}$ is a weak revision of P.

Let $\alpha \in \mathcal{E}$. As $\mathcal{E} \subseteq \mathcal{U}$, there exists $r \in P$ such that $\alpha = head(r)$ and $\mathcal{I} \oplus \mathcal{U} \models body(r)$. Thus, $\mathcal{I} \oplus \mathcal{E} \models body(r)$, too. Consequently, α is P-founded wrt $\mathcal{I}$ and $\mathcal{E}$. It follows that $\mathcal{E}$ is a P-founded weak revision of $\mathcal{I}$. □

At an intuitive level, we already argued earlier that foundedness is less restrictive than the condition defining justified updates, which is behind justified (weak) revisions. We will now make this intuition formal.

Theorem 11
Let P be a revision program and let $\mathcal{I}$ be a database. If a set $\mathcal{E}$ of revision literals is a P-justified (weak) revision of $\mathcal{I}$, then it is a P-founded (weak) revision of $\mathcal{I}$.

Proof: Let $\mathcal{E}$ be a P-justified weak revision of $\mathcal{I}$. By Proposition 4, $\mathcal{I} \oplus \mathcal{E} \models P$. Moreover, there is a P-justified update $\mathcal{U}$ for $\mathcal{I}$ such that $\mathcal{E} = \mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$. It follows that $\mathcal{I} \oplus \mathcal{U} = \mathcal{I} \oplus \mathcal{E}$ and $\mathcal{E} \cap I(\mathcal{I}, \mathcal{I} \oplus \mathcal{E}) = \emptyset$. Since $\mathcal{U}$ s consistent (by the definition), $\mathcal{E}$ is consistent and so, $\mathcal{E}$ is a weak revision of $\mathcal{I}$ with respect to P.

To show that $\mathcal{E}$ is a P-founded weak revision of $\mathcal{I}$, we need to prove that $\mathcal{E}$ is P-founded wrt $\mathcal{I}$. Let $\alpha \in \mathcal{E}$. We recall that by the definition, $\mathcal{U}$ is a minimal set closed under $P \cup I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$. As $\mathcal{U}$ is minimal, $\mathcal{U}' = \mathcal{U} \setminus \{\alpha\}$ is not closed under $P \cup I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$. As $\alpha \notin I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$ there is a revision rule $r \in P$ such that $body(r) \subseteq \mathcal{U}'$ and $head(r) \cap \mathcal{U}' = \emptyset$. Since $\mathcal{U}' \subseteq \mathcal{U}$, $body(r) \subseteq \mathcal{U}$. It follows that $\mathcal{I} \oplus \mathcal{U} \models body(r)$ and so, $\mathcal{I} \oplus \mathcal{E} \models bd(r)$.

We recall that $\mathcal{U}$ is closed under P. Thus, $head(r) \cap \mathcal{U} = \{\alpha\}$. Let $\beta \in head(r) \setminus \{\alpha\}$. It follows that $\beta \notin \mathcal{U}$ and so, $\beta \notin \mathcal{E}$ and $\beta \notin I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$. If $\mathcal{I} \models \beta$, then $\beta \notin I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$ implies that $\mathcal{I} \oplus \mathcal{U} \not\models \beta$. If $\mathcal{I} \not\models \beta$, then $\beta \notin \mathcal{E}$ implies $\mathcal{I} \oplus \mathcal{E} \not\models \beta$. In each case $\mathcal{I} \oplus \mathcal{E} \models \beta^D$. It follows that α is P-founded wrt $\mathcal{I}$. Thus, $\mathcal{E}$ is P-founded wrt $\mathcal{I}$ and so, it is a P-founded weak revision of $\mathcal{I}$

Next, let us assume that $\mathcal{E}$ is a P-justified revision of $\mathcal{I}$. Then, $\mathcal{E}$ is a P-justified weak revision of $\mathcal{I}$ and so, a P-founded weak revision of $\mathcal{I}$ (by the argument above). In particular, it is P-founded wrt $\mathcal{I}$. Moreover, since $\mathcal{E}$ is a P-justified revision of $\mathcal{I}$, it is a revision of $\mathcal{I}$ wrt P. Therefore, $\mathcal{E}$ it is a P-founded revision of $\mathcal{I}$ wrt P. □

The converse implications do not hold in general (cf. Example 11).

As in the case of active integrity constraints, revision rules can be *normalized*. Namely, for a revision rule $r = \alpha_1 | \dots | \alpha_k \leftarrow \phi$ by r^n we denote the set of *normal* revision rules as follows: $r^n = \{r\}$, if $k \leq 1$ or, if $k \geq 2$, $r^n = \{r_1, \dots, r_k\}$, where $r_i = \alpha_i \leftarrow \alpha_1^D, \dots, \alpha_{i-1}^D, \alpha_{i+1}^D, \dots, \alpha_k^D, \phi$. For a revision program P, we define $P^n = \bigcup_{r \in P} r^n$. One can prove the following result (we omit the details as they are quite similar to those we presented above).

Theorem 12
Let P be a revision program and let $\mathcal{I}$ be a database. A set $\mathcal{E}$ of revision literals is a (weak) revision of $\mathcal{I}$ with respect to P^n (P^n-founded (weak) revision of $\mathcal{I}$, respectively) if and only if it is a (weak) revision of $\mathcal{I}$ with respect to P (P-founded (weak) revision of $\mathcal{I}$, respectively). Moreover, if $\mathcal{E}$ is a P^n-justified (weak) revision of $\mathcal{I}$, then it is a P-justified (weak) revision of $\mathcal{I}$.

To summarize our discussion so far, revision programs can be assigned the semantics of (weak) revisions, justified (weak) revisions and founded (weak) revisions. Thanks to Theorem 12, we can also assign to a revision program P the semantics of P^n-justified revisions. Let us denote the classes of the corresponding types of revisions by $\mathbf{Rev}(\mathcal{I}, P)$, $\mathbf{WRev}(\mathcal{I}, P)$, $\mathbf{JRev}(\mathcal{I}, P)$, $\mathbf{JWRev}(\mathcal{I}, P)$, $\mathbf{FRev}(\mathcal{I}, P)$ and $\mathbf{FWRev}(\mathcal{I}, P)$. The relationships between the semantics we discussed above are demonstrated in Figure 2. One can show that none of the containment relations can be replaced with the equality.

$$
\begin{array}{ccccccccc}
 & & & & \mathbf{FRev}(\mathcal{I}, P^n) & & & & \\
 & & & & \parallel & & & & \\
\mathbf{JRev}(\mathcal{I}, P^n) & \subseteq & \mathbf{JRev}(\mathcal{I}, P) & \subseteq & \mathbf{FRev}(\mathcal{I}, P) & \subseteq & \mathbf{Rev}(\mathcal{I}, P) & = & \mathbf{Rev}(\mathcal{I}, P^n) \\
\parallel & & \cap\!\!\!| & & \cap\!\!\!| & & \cap\!\!\!| & & \cap\!\!\!| \\
\mathbf{JWRev}(\mathcal{I}, P^n) & \subseteq & \mathbf{JWRev}(\mathcal{I}, P) & \subseteq & \mathbf{FWRev}(\mathcal{I}, P) & \subseteq & \mathbf{WRev}(\mathcal{I}, P) & = & \mathbf{WRev}(\mathcal{I}, P^n) \\
 & & & & \parallel & & & & \\
 & & & & \mathbf{FWRev}(\mathcal{I}, P^n) & & & &
\end{array}
$$

Fig. 2. The containment relations for the semantics of revision programs

The similarities revision programs show to sets of active integrity constraints are striking. In the next section, we will now establish the precise connection.

11 Connections between Revision Programs and Active Integrity Constraints

To relate revision programs and active integrity constraints, we first note that we can restrict the syntax of revision programs without affecting their expressivity.

A *proper revision rule* is a revision rule that satisfies the following condition: any literal in the head is not the dual of any literal in the body.

Let P be a revision program and let r_1 and r_2 be revision rules

$$\alpha|\alpha_1|\dots|\alpha_k \leftarrow \alpha^D, \beta_1, \dots, \beta_m$$

and

$$\alpha_1|\dots|\alpha_k \leftarrow \alpha^D, \beta_1, \dots, \beta_m,$$

respectively (that is, r_2 differs from r_1 in that it drops α from the head).

Lemma 5

Let $\mathcal{I}$ be a database. Under the notation introduced above, a set of revision literals $\mathcal{U}$ is a (weak) revision of $\mathcal{I}$ with respect to $P \cup \{r_1\}$ ($P \cup \{r_1\}$-founded (weak) revision, $P \cup \{r_1\}$-justified (weak) revision of $\mathcal{I}$, respectively) if and only if $\mathcal{U}$ is a (weak) revision of $\mathcal{I}$ with respect to $P \cup \{r_2\}$ ($P \cup \{r_2\}$-founded (weak) revision, $P \cup \{r_2\}$-justified (weak) revision of $\mathcal{I}$, respectively).

Proof: The claim is evident for the case of weak revisions and revisions. The case of justified (weak) revisions follows from the observation that a consistent set $\mathcal{U}$ of revision literals is a closed set for $P \cup \{r_1\} \cup I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$ if and only if $\mathcal{U}$ is a closed set for $P \cup \{r_2\} \cup I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$.

For the case of founded (weak) revisions, it is enough to prove that a set $\mathcal{U}$ of revision literals is $P \cup \{r_1\}$-founded wrt $\mathcal{I}$ if and only if $\mathcal{U}$ is $P \cup \{r_2\}$-founded wrt $\mathcal{I}$. When proceeding in either direction, we have that $\mathcal{U}$ is consistent.

Let $\beta \in \mathcal{U}$ be $P \cup \{r_1\}$-founded wrt $\mathcal{I}$ and $\mathcal{U}$, and let $r \in P \cup \{r_1\}$ be the rule providing support to β. If $r \neq r_1$, $r \in P$ and so, β is $P \cup \{r_2\}$-founded wrt $\mathcal{I}$ and $\mathcal{U}$. Thus, let us assume that $r = r_1$. If $\beta = \alpha$, then $\alpha \in \mathcal{U}$ and, consequently, $\mathcal{I} \oplus \mathcal{U} \models \alpha$. Since $\mathcal{I} \oplus \mathcal{U} \models \mathit{body}(r_1)$, $\mathcal{I} \oplus \mathcal{U} \models \alpha^D$, a contradiction. Thus, $\beta \neq \alpha$. It is easy to see that in such case, r_2 supports β (given $\mathcal{U}$). Thus, β is $P \cup \{r_2\}$-founded wrt $\mathcal{I}$ in this case, too. It follows that $\mathcal{U}$ is $P \cup \{r_2\}$-founded wrt $\mathcal{I}$.

Conversely, let $\beta \in \mathcal{U}$ be $P \cup \{r_2\}$-founded wrt $\mathcal{I}$ and $\mathcal{U}$, and let $r \in P \cup \{r_2\}$ be the rule providing support to β. As before, if $r \neq r_2$, the claim follows. If $r = r_2$, then $\beta \neq \alpha$. Since r_2 supports β, one can check that r_1 supports, β, too. Thus, β is $P \cup \{r_1\}$-founded wrt $\mathcal{I}$ and $\mathcal{U}$. Consequently, $\mathcal{U}$ is $P \cup \{r_1\}$-founded wrt $\mathcal{I}$ □

Lemma 5 shows that the literals in the head of a revision rule which are dual of literals in the body are useless and can be dropped. In other words, there is no loss of generality in considering just proper revision programs.

Example 13
Let P be the revision program containing the rules $\{\mathbf{in}(b)|\mathbf{out}(a) \leftarrow \mathbf{in}(a), \mathbf{out}(d)| \mathbf{in}(c) \leftarrow \mathbf{out}(c)\}$. Its properized version is $\{\mathbf{in}(b) \leftarrow \mathbf{in}(a),\ \mathbf{out}(d) \leftarrow \mathbf{out}(c)\}$. □

Theorem 13
Let P be a revision program. There is a proper revision program P' such that for every database $\mathcal{I}$, (weak) revisions of $\mathcal{I}$ with respect to P (P-founded (weak) revisions, P-justified (weak) revisions of $\mathcal{I}$, respectively) coincide with (weak) revisions of $\mathcal{I}$ with respect to P' (P'-founded (weak) revisions, P'-justified (weak) revisions of $\mathcal{I}$, respectively).

Proof: Lemma 5 implies that the program P' obtained from P by repeated application of the process described above (replacement of rules of the form r_1 with the corresponding rules of the form r_2) has the required property. □
We denote the "properized" version of a revision program P as $prop(P)$.
We extend to revision literals the operator $ua(\cdot)$ defined for propositional literals. If $\alpha = \mathbf{in}(a)$, we define $ua(\alpha) = +a$. If $\alpha = \mathbf{out}(a)$, we define $ua(\alpha) = -a$.

Definition 11
Given a proper revision rule r of the form

$$\alpha_1| \dots |\alpha_k \leftarrow \beta_1, \dots \beta_m$$

we denote by $AIC(r)$ the active integrity constraint

$$lit(\beta_1), \dots, lit(\beta_m), lit(\alpha_1)^D, \dots, lit(\alpha_k)^D \supset ua(\alpha_1)| \dots |ua(\alpha_k)\cdot$$ □

For example, given the proper revision rule $r\ :\ \mathbf{in}(a) \leftarrow \mathbf{out}(b)$, the corresponding active integrity constraint $AIC(r)$ is of the form $not\ b, not\ a \supset +a$. We note that if r is a constraint ($k = 0$), $AIC(r)$ is simply an integrity constraint. The operator $AIC(\cdot)$ is extended to proper revision programs in the standard way. It is easy to show that for each database $\mathcal{I}$, $\mathcal{I} \models P$ if and only if $\mathcal{I} \models AIC(P)$. The following lemma establishes a direct connection between the concepts of closure under active integrity constraints and revision programs.

Lemma 6
Let r be a proper revision rule. A set $\mathcal{E}$ of revision literals is closed under P if and only if $ua(\mathcal{E})$ is closed under $AIC(r)$.

Proof: First, we observe that as r is proper, $nup(AIC(r)) = lit(body(r))$. Moreover $head(AIC(r)) = ua(head(r))$. We know that $\mathcal{E}$ is closed under r if and only if $body(r) \not\subseteq \mathcal{E}$ or $head(r) \cap \mathcal{E} \neq \emptyset$. This holds if and only if $lit(body(r)) \not\subseteq lit(\mathcal{E}) = lit(ua(\mathcal{E}))$ or $ua(head(r)) \cap ua(\mathcal{E}) \neq \emptyset$, which is equivalent to $nup(AIC(r)) \not\subseteq lit(ua(\mathcal{E}))$ or $head(AIC(r)) \cap ua(\mathcal{E}) \neq \emptyset$. This, however, is the definition of $AIC(r)$ closed under $ua(\mathcal{E})$). □

Corollary 3
Let P be a proper revision program. A set $\mathcal{E}$ of revision literals is a minimal set closed under P if and only if $ua(\mathcal{E})$ is a minimal set closed under $AIC(r)$.

Proof: Straightforward from Lemma 6. □

Theorem 14
Let P be a proper revision program. A set $\mathcal{E}$ of revision literals is a (weak) revision (respectively, P-justified (weak) revision, P-founded (weak) revision) of $\mathcal{I}$ wrt P if and only if $ua(\mathcal{E})$ is a (weak) repair (respectively, justified (weak) repair, founded (weak) repair) for $\langle \mathcal{I}, AIC(P) \rangle$.

Proof:
(1) A set $\mathcal{E}$ of revision literals is a weak revision of $\mathcal{I}$ wrt P if and only if $ua(\mathcal{E})$ is a weak repair for $\langle \mathcal{I}, AIC(P) \rangle$.
Indeed, by the definition, $\mathcal{E}$ is a weak revision of $\mathcal{I}$ with respect to P if and only if

(a) $\mathcal{I} \cap \{a \mid \mathbf{in}(a) \in \mathcal{E}\} = \emptyset$, $\{a \mid \mathbf{out}(a) \in \mathcal{E}\} \subseteq \mathcal{I}$; and
(b) $\mathcal{I} \oplus \mathcal{E} \models P$.

Similarly, $ua(\mathcal{E})$ is a weak repair for $\langle \mathcal{I}, AIC(P) \rangle$ if and only if

(a) $\mathcal{I} \cap \{a \mid +a \in ua(\mathcal{E})\} = \emptyset$, $\{a \mid -a \in ua(\mathcal{E})\} \subseteq \mathcal{I}$; and
(b) $\mathcal{I} \circ ua(\mathcal{E}) \models AIC(P)$.

By our earlier comments, for every database $\mathcal{J}$, $\mathcal{J} \models P$ if and only if $\mathcal{J} \models AIC(P)$. Since $\mathcal{I} \oplus \mathcal{E} = \mathcal{I} \circ ua(\mathcal{E})$, the assertion follows.

(2) Next, we prove that $\mathcal{E}$ is a revision of $\mathcal{I}$ wrt P if and only if $ua(\mathcal{E})$ is a repair for $\langle \mathcal{I}, AIC(P) \rangle$.
By (1), $\mathcal{E}$ is a weak revision of $\mathcal{I}$ wrt P if and only if $ua(\mathcal{E})$ is a weak repair for $\langle \mathcal{I}, AIC(P) \rangle$. Moreover, we have that the mapping $\mathcal{E} \mapsto ua(\mathcal{E})$ is a bijection between sets of revision literals and sets of update actions such that $\mathcal{I} \oplus \mathcal{E} \models P$ if and only if $\mathcal{I} \circ ua(\mathcal{E}) \models AIC(P)$. Thus, a set $\mathcal{E}$ of revision literals is such that for each $\mathcal{E}' \subseteq \mathcal{E}$ the fact $\mathcal{I} \oplus \mathcal{E}' \models P$ implies $\mathcal{E}' = \mathcal{E}$ (minimality of $\mathcal{E}$) if and only if $ua(\mathcal{E})$ is a set of update actions such that for each $\mathcal{F}' \subseteq ua(\mathcal{E})$ the fact $\mathcal{I} \circ \mathcal{F}' \models AIC(P)$ implies $\mathcal{F}' = ua(\mathcal{E})$ (minimality of $ua(\mathcal{E})$.

(3) We now prove that $\mathcal{E}$ is a P-justified weak revision of $\mathcal{I}$ if and only if $ua(\mathcal{E})$ is a justified weak repair for $\langle \mathcal{I}, AIC(P) \rangle$.

($\Rightarrow$) Since $\mathcal{E}$ is a P-justified weak revision of $\mathcal{I}$, there exists a P-justified weak update of $\mathcal{I}$, say $\mathcal{U}$, such that $\mathcal{E} = \mathcal{U} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$. By the definition, $\mathcal{U}$ is consistent and it is a minimal set containing $I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})$ and closed under P. It follows that the action set $ua(\mathcal{U})$ is consistent and, by Corollary 3, it is a minimal set containing $ua(I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U}))$ and closed under $AIC(P)$. We now observe that $ua(I(\mathcal{I}, \mathcal{I} \oplus \mathcal{U})) = ne(\mathcal{I}, \mathcal{I} \circ ua(\mathcal{U}))$. Thus, $ua(\mathcal{U})$ is a justified action set for $\langle \mathcal{I}, AIC(P) \rangle$ and $ua(\mathcal{U}) \setminus ne(\mathcal{I}, \mathcal{I} \circ ua(\mathcal{U})) = ua(\mathcal{E})$ is a justified weak repair for $\langle \mathcal{I}, AIC(P) \rangle$.

($\Leftarrow$) There exists a justified action set for $\langle \mathcal{I}, AIC(P) \rangle$, say $\mathcal{U}$, such that $ua(\mathcal{E}) = \mathcal{U} \setminus ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$. The action set $\mathcal{U}$ is consistent, contains $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ and it is closed under $AIC(P)$. By our comments above, there is a set of revision literals $\mathcal{V}$ such that $ua(\mathcal{V}) = \mathcal{U}$. Moreover, $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) = ua(I(\mathcal{I}, \mathcal{I} \oplus \mathcal{V}))$. It follows that the set $\mathcal{V}$ is consistent and, by Corollary 3, it is a minimal set containing $I(\mathcal{I}, \mathcal{I} \oplus \mathcal{V})$ and closed under P. Thus, $\mathcal{V}$ is a P-justified weak update for $\mathcal{I}$ and $\mathcal{V} \setminus I(\mathcal{I}, \mathcal{I} \oplus \mathcal{V}) = \mathcal{E}$ is a P-justified weak revision for $\mathcal{I}$.

(4) By (3) and by the argument we used in (2) to show that the minimality of $\mathcal{E}$ is equivalent to the minimality of $ua(\mathcal{E})$, $\mathcal{E}$ is a P-justified revision of $\mathcal{I}$ if and only if $ua(\mathcal{E})$ is a justified repair for $\langle \mathcal{I}, AIC(P) \rangle$.

(5) Finally, we prove that $\mathcal{E}$ is a P-founded (weak) revision of $\mathcal{I}$ if and only if $ua(\mathcal{E})$ is a founded (weak) repair for $\langle \mathcal{I}, AIC(P) \rangle$.

($\Rightarrow$) Let $\mathcal{E}$ be a P-founded (weak) revision of $\mathcal{I}$. By (1) and (2), $ua(\mathcal{E})$ is a (weak) repair for $\langle \mathcal{I}, AIC(P) \rangle$. Therefore, we have to show that $ua(\mathcal{E})$ is founded wrt $\langle \mathcal{I}, AIC(P) \rangle$. Let us consider an arbitrary element of $ua(\mathcal{E})$. It is of the form $ua(\alpha)$, for some revision literal $\alpha \in \mathcal{E}$.

Since $\mathcal{E}$ is P-founded wrt $\mathcal{I}$, there exists $r \in P$ such that $\mathcal{I} \oplus \mathcal{E} \models body(r)$, and $\mathcal{I} \oplus \mathcal{E} \models \gamma^D$, for every $\gamma \in head(r)$ different from α. Let ρ be the corresponding active integrity constraint in $AIC(P)$, that is, $\rho = AIC(r)$. Since r is proper, $lit(body(r)) = nup(\rho)$. Thus, $\mathcal{I} \circ ua(\mathcal{E}) \models nup(\rho)$. Moreover, since $head(\rho) = ua(head(r))$, for every $\delta \in head(\rho)$ other than $ua(\alpha)$, $\mathcal{I} \circ ua(E) \models \delta^D$.

Thus, $ua(\alpha)$ is founded wrt $\langle \mathcal{I}, AIC(P) \rangle$ and $ua(\mathcal{E})$ and so, $ua(\mathcal{E})$ is founded with respect to $\langle \mathcal{I}, AIC(P) \rangle$.

($\Leftarrow$) This implication can be proved by a similar argument. We omit the details. □

The results of this section show that proper revision programs can be interpreted as sets of active integrity constraints so that the corresponding semantics match. However, it is easy to see that the mapping $AIC(\cdot)$ is a one-to-one and onto mapping between the collection of proper revision programs and the collections of sets of active integrity constraints. Thus, also conversely, sets of active integrity constraints can be interpreted as revision programs.

Example 14
Let η be the following set of active integrity constraints:

$$
\begin{aligned}
r_1 = \quad & a, b, not\ c & \supset & \quad -a \mid +c \\
r_2 = \quad & not\ d & \supset & \quad +d \\
r_3 = \quad & a & \supset & \quad \bot
\end{aligned}
$$

The corresponding revision program is:

$$
\begin{aligned}
\rho_1 = \quad & \mathbf{out}(a) | \mathbf{in}(c) & \leftarrow & \quad \mathbf{in}(b) \\
\rho_2 = \quad & \mathbf{in}(d) & \leftarrow & \\
\rho_3 = \quad & \bot & \leftarrow & \quad \mathbf{in}(a) \cdot
\end{aligned}
$$

The correspondence between sets of active integrity constraints and proper revision programs allows us to adapt results from one setting to another and conversely. Moreover, in many cases, once we have a result for proper revision programs, we can lift it to the general case, too. For instance, as in the case of sets of active integrity constraints and justified (weak) repairs, a special structure of a revision program with respect to the original database ensures minimality of justified weak revisions. Specifically, we have the following corollary of Theorem 3.

Theorem 15
Let $\mathcal{I}$ be a database and P a revision program such that for each revision literal α appearing in the head of a rule in P, $\mathcal{I} \models \alpha^D$. If $\mathcal{E}$ is a P-justified weak revision for $\mathcal{I}$, then $\mathcal{E}$ is a P-justified revision for $\mathcal{I}$.

Proof: (Sketch) Clearly, the properized version P' of P also satisfies the assumption of the theorem. By the correspondence results between proper revision programs and sets of aic's, it follows that E is a P'-justified revision for $\mathcal{I}$. As $\mathcal{I}$ has the same justified revisions with respect to P' and P, the result follows. □

Moreover, for normal revision programs justified weak revisions are justified revisions no matter what the initial database, as stated in the following corollary to Theorem 4. The argument is essentially the same as the one above and we omit it.

Theorem 16
Let $\mathcal{I}$ be a database and P a normal revision program. If $\mathcal{E}$ is a P-justified weak revision for $\mathcal{I}$, then $\mathcal{E}$ is a justified revision for $\mathcal{I}$.

12 Computation and Complexity Results for Revision Programming

Thanks to the equivalence properties reported in Section 11 we can derive the results about computation and complexity for revision programming from the corresponding results for active integrity constraints presented in Section 7.

Theorem 17
Let $\mathcal{I}$ be a database and P a normal revision program. Then checking if there exists a P-justified revision (P-justified weak revision, respectively) for $\mathcal{I}$ is an NP-complete problem.

Proof. By Theorem 13 we know that this problem is equivalent to check if there exists a P'-justified revision (P'-justified weak revision, respectively) for $\mathcal{I}$ where P' is the properized version of P that can be computed in polynomial time. The result follows from Theorems 7 and Theorem 14. □

Theorem 18
Let $\mathcal{I}$ be a database and P a revision program. Then checking if there exists a P-justified revision (P-justified weak revision, respectively) for $\mathcal{I}$ is a Σ_2^P-complete problem.

Proof. By Theorem 13 we know that this problem is equivalent to check if there exists a P'-justified revision (P'-justified weak revision, respectively) for $\mathcal{I}$ where P' is the properized version of P that can be computed in polynomial time. The result follows from Theorems 8, 9 and 14. □

Theorem 19
Let $\mathcal{I}$ be a database and P a revision program. Then checking if there exists a P-founded revision (P-founded weak revision, respectively) for $\mathcal{I}$ is a Σ_2^P-complete (NP-complete, respectively) problem.

				$\mathbf{FRev}(\mathcal{I}, P^n)$ [Σ^2_P-c]				
				$\parallel$				
$\mathbf{JRev}(\mathcal{I}, P^n)$ [NP-c]	$\subseteq$	$\mathbf{JRev}(\mathcal{I}, P)$ [Σ^2_P-c]	$\subseteq$	$\mathbf{FRev}(\mathcal{I}, P)$ [Σ^2_P-c]	$\subseteq$	$\mathbf{Rev}(\mathcal{I}, P)$ [NP-c]	$=$	$\mathbf{Rev}(\mathcal{I}, P^n)$ [NP-c]
$\parallel$		$\subseteq$		$\subseteq$		$\subseteq$		$\subseteq$
$\mathbf{JWRev}(\mathcal{I}, P^n)$ [NP-c]	$\subseteq$	$\mathbf{JWRev}(\mathcal{I}, P)$ [Σ^2_P-c]	$\subseteq$	$\mathbf{FWRev}(\mathcal{I}, P)$ [NP-c]	$\subseteq$	$\mathbf{WRev}(\mathcal{I}, P)$ [NP-c]	$=$	$\mathbf{WRev}(\mathcal{I}, P^n)$ [NP-c]
				$\parallel$				
				$\mathbf{FWRev}(\mathcal{I}, P^n)$ [NP-c]				

Fig. 3. Complexity results for the semantics of revision programs

Proof. By Theorem 13 we know that this problem is equivalent to check if there exists a P'-founded revision (P'-founded weak revision, respectively) for $\mathcal{I}$ where P' is the properized version of P that can be computed in polynomial time. The result follows from complexity results by Caroprese et al. (2006) and Theorem 14. □

We summarize the complexity results obtained in this section in Figure 3.

We note that comments we made at the end of Section 8 apply here as well. In a nutshell, a semantics of justified revisions, reflecting the principles of groundedness (no circular "self-justifications") and minimality of change, seems to be well motivated and so most appealing for applications. However, as we pointed out earlier, it may be too restrictive. Thus, in all these cases, when consistency of a database needs to be restored and justified revisions do not exist, other semantics may provide an acceptable solution. The discussion of that issue, involving also computational complexity trade-offs, follows essentially the same line as that in Section 8.

13 Shifting Theorem

In this section we study the *shifting* transformation (Marek and Truszczyński 1998). The process consists of transforming an instance $\langle \mathcal{I}, \eta \rangle$ of the database repair problem to a syntactically isomorphic instance $\langle \mathcal{I}', \eta' \rangle$ by changing integrity constraints to reflect the "shift" of $\mathcal{I}$ into $\mathcal{I}'$. A semantics for database repair problem has the *shifting property* if the repairs of the "shifted" instance of the database update problem are precisely the results of modifying the repairs of the original instance according to the shift from $\mathcal{I}$ to $\mathcal{I}'$. The shifting property is important. If a semantics of database updates has it, the study of that semantics can be reduced to the case when the input database is the empty set, a major conceptual simplification.

Example 15

Let $\mathcal{I} = \{a, b\}$ and let $\eta_5 = \{a, b \supset -a | -b\}$. There are two founded repairs for $\langle \mathcal{I}, \eta_5 \rangle$: $\mathcal{E}_1 = \{-a\}$ and $\mathcal{E}_2 = \{-b\}$. Let $\mathcal{W} = \{a\}$. We will now "shift" the instance $\langle \mathcal{I}, \eta_5 \rangle$ with respect to $\mathcal{W}$. To this end, we will first modify $\mathcal{I}$ by changing the status in $\mathcal{I}$ of elements in $\mathcal{W}$, in our case, of a. Since $a \in \mathcal{I}$, we will remove it. Thus, $\mathcal{I}$ "shifted" with respect to $\mathcal{W}$ becomes $\mathcal{J} = \{b\}$. Next, we will modify η_5 correspondingly, replacing literals and update actions involving a by their duals. That results in $\eta'_5 = \{not\ a, b \supset +a | -b\}$. One can check that the resulting instance $\langle \mathcal{J}, \eta'_5 \rangle$ of the update problem has two founded repairs: $\{+a\}$ and $\{-b\}$. Moreover, they can be obtained from the founded repairs for $\langle \mathcal{I}, \eta_5 \rangle$ by consistently replacing $-a$ with $+a$ and $+a$ with $-a$ (the latter does not apply in this example). In other words, the original update problem and its shifted version are isomorphic. □

The situation presented in Example 15 is not coincidental. In this section we will show that the semantics of (weak) repairs, founded (weak) repairs and justified (weak) repairs satisfy the shifting property. To facilitate the presentation, we placed proofs of all the results in the appendix.

We start by observing that *shifting* a database $\mathcal{I}$ to a database $\mathcal{I}'$ can be modeled by means of the symmetric difference operator. Namely, we have $\mathcal{I}' = \mathcal{I} \div \mathcal{W}$, where $\mathcal{W} = \mathcal{I} \div \mathcal{I}'$. This identity shows that one can shift any database $\mathcal{I}$ into any database $\mathcal{I}'$ by forming a symmetric difference of $\mathcal{I}$ with some set $\mathcal{W}$ of atoms (specifically, $\mathcal{W} = \mathcal{I} \div \mathcal{I}'$). We will now extend the operation of shifting a database with respect to $\mathcal{W}$ to the case of literals, update actions and integrity constraints. To this end, we introduce a *shifting* operator $T_{\mathcal{W}}$.

Definition 12

Let $\mathcal{W}$ be a database and ℓ a literal or an update action. We define

$$T_{\mathcal{W}}(\ell) = \begin{cases} \ell^D & \text{if the atom of } \ell \text{ is in } \mathcal{W} \\ \ell & \text{if the atom of } \ell \text{ is not in } \mathcal{W} \end{cases}$$

and we extend this definition to sets of literals or update actions, respectively.

Furthermore, if *op* is an operator on sets of literals or update actions (such as conjunction or disjunction), for every set X of literals or update actions, we define

$$T_{\mathcal{W}}(op(X)) = op(T_{\mathcal{W}}(X)).$$

Finally, for an active integrity constraint $r = \phi \supset \psi$, we set

$$T_{\mathcal{W}}(r) = T_{\mathcal{W}}(\phi) \supset T_{\mathcal{W}}(\psi).$$

We extend the notation to sets active integrity constraints in the standard way. □

To illustrate the last two parts of the definition, we note that when *op* stands for the conjunction of a set of literals and $X = \{L_1, \ldots, L_n\}$, where every L_i is a literal, $T_{\mathcal{W}}(op(X)) = op(T_{\mathcal{W}}(X))$ specializes to

$$T_{\mathcal{W}}(L_1, \ldots, L_n) = T_{\mathcal{W}}(L_1), \ldots, T_{\mathcal{W}}(L_n).$$

Similarly, for an active integrity constraint

$$r = L_1, \ldots, L_n \supset \alpha_1 | \ldots | \alpha_m$$

we obtain

$$T_{\mathcal{W}}(r) = T_{\mathcal{W}}(L_1), \ldots, T_{\mathcal{W}}(L_n) \supset T_{\mathcal{W}}(\alpha_1) | \ldots | T_{\mathcal{W}}(\alpha_m).$$

To summarize, we overload the notation $T_{\mathcal{W}}$ and interpret it based on the type of the argument.

Theorem 20 (Shifting theorem for (weak) repairs and founded repairs)

Let $\mathcal{I}$ and $\mathcal{W}$ be databases. For every set η of active integrity constraints and for every consistent set $\mathcal{E}$ of update actions, we have

1. $\mathcal{E}$ is a weak repair for $\langle \mathcal{I}, \eta \rangle$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a weak repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$
2. $\mathcal{E}$ is a repair for $\langle \mathcal{I}, \eta \rangle$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$
3. $\mathcal{E}$ is founded for $\langle \mathcal{I}, \eta \rangle$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is founded for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$
4. $\mathcal{E}$ is a founded (weak) repair for $\langle \mathcal{I}, \eta \rangle$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a founded (weak) repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$
5. $\mathcal{E}$ is an justified (weak) repair for $\langle \mathcal{I}, \eta \rangle$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a justified (weak) repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$.

Theorem 20 implies that in the context of (weak) repairs, founded (weak) repairs or justified (weak) repairs, an instance $\langle \mathcal{I}, \eta \rangle$ of the database update problem can be shifted to the instance with the empty initial database. That property can simplify studies of these semantics as well as the development of algorithms for computing repairs and for consistent query answering, as it allows us to eliminate one of the parameters (the initial database) from considerations. In many cases it also allows us to relate semantics of database repairs to some semantics of logic programs with negation. Formally, we have the following corollary.

Corollary 4
Let $\mathcal{I}$ be a database and η a set of active integrity constraints. Then $\mathcal{E}$ is a weak repair (repair, founded weak repair, founded repair, justified weak repair, justified repair, respectively) for $\langle \mathcal{I}, \eta \rangle$ if and only if $T_{\mathcal{I}}(\mathcal{E})$ is a weak repair (repair, founded weak repair, founded repair, justified weak repair, justified repair, respectively) for $\langle \emptyset, T_{\mathcal{I}}(\eta) \rangle$.

The concept of of shifting can also be stated for revision programming. First, we note that the operator $T_{\mathcal{W}}(\cdot)$ defined above can be extended to revision literals, revision rules and revision programs. Its formal definition and many properties have been presented by Marek et al. (1999). The following theorem gathers those results, as well as their extensions to the case of new semantics we introduced in our paper.

Theorem 21
(Shifting theorem for revision programs) Let $\mathcal{I}$ and $\mathcal{W}$ be databases. For every revision program G and every consistent set $\mathcal{E}$ of revision literals, we have

1. $\mathcal{E}$ is a (weak) revision for $\mathcal{I}$ with respect to G if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a (weak) revision for $\mathcal{I}$ with respect to $T_{\mathcal{W}}(G)$
2. $\mathcal{E}$ is a G-justified (weak) revision for $\mathcal{I}$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a $T_{\mathcal{W}}(G)$-justified (weak) revision for $\mathcal{I}$
3. $\mathcal{E}$ is a G-founded (weak) revision for $\mathcal{I}$ if and only if $T_{\mathcal{W}}(\mathcal{E})$ is a $T_{\mathcal{W}}(G)$-founded (weak) revision for $\mathcal{I}$

14 Conclusion

In the paper we studied two formalisms for describing policies on enforcing integrity constraints on databases in the presence of preferences on alternative ways to do so: *active integrity constraints* (Caroprese et al. 2006) and *revision programming* (Marek and Truszczyński 1998).

The original semantics proposed for active integrity constraints is based on the concept of a *founded repair*. A founded repair is a set of *update actions* (*insertions* and *deletions*) to be performed over the database in order to make it consistent, that is minimal and *supported* by active integrity constraints. The original semantics for revision programs is based on the concept of *justified revision*. A justified revision is a set of *revision literals* that can be inferred by means of the revision program and by the *inertia set*, that is the set of all atoms that do not change their state of *presence* in or *absence* from a database during the revision process.

We proved that in the context of their original semantics, these two formalisms differ. That is, under some natural interpretation of revision programs as sets of active integrity constraints the set of repairs corresponding to justified revisions is contained in the set of founded repairs (and the containment is, in general, proper). This observation demonstrated that basic intuitions behind the two semantics are essentially different and opened a possibility of expanding each formalism by semantics grounded in the ideas developed in the other one.

Following this direction, we introduced a new semantics for active integrity constraints, based on ideas underlying revision programming and, conversely, a new semantics for revision programs based on intuitions behind founded repairs. With the new semantics available, we showed that the interpretation of revision programs as sets of active integrity constraints, mentioned above, establishes a precise match between these two formalism: it preserves their semantics once they are correctly aligned. In other words, we proved that the two formalisms are equivalent through a simple modular (rule-wise) syntactic transformation. That offers a strong indication of the adequacy of each formalism as the foundation for declarative specifications of policies for enforcing integrity constraints. Moreover, the broad frameworks of semantics we have available in each case provide us with means of handling the problem of "non-executability" of the policies encoded into integrity constraints under a particular semantics: once that turns out to be the case, one can chose to select a less restrictive one.

For each formalism and each semantics we established the complexity of the basic existence of repair (revision) problem. Furthermore, we proved that each formalism and each semantics satisfies the *shifting property*. Shifting consists of transforming an instance of a database repair problem to another syntactically isomorphic instance by changing active integrity constraints or revision programs to reflect the "shift" from the original database to the new one.

These latter results are essential for relating repair (revision) formalisms we studied with logic programming and, specifically, with programs that generalize standard disjunctive logic programs by allowing default literals also in the heads of disjunctive rules (the Lifschitz-Woo programs (Lifschitz and Woo 1992); cf. work by Marek at al. (1999) and Pivkina (2001) for some early results exploiting shifting to relate revision and logic programming).

Our work opens and forms the foundation for several research directions. The first of them concerns implementations of algorithms for computing repairs (revisions) under the semantics discussed here in the first-order setting covering built-in predicates and aggregates. An important aspect of that research is to identify classes of databases and integrity constraints, for which the existence and the uniqueness of repairs (revisions) of particular types is assured.

The second problem concerns consistent query answering in the setting of active integrity constraints. The problem is to compute answers to queries to a database that is inconsistent with respect to its active integrity constraints without computing the repairs explicitly, thus extending the approach of consistent query answering (Arenas et al. 1999; Arenas et al. 2003; Chomicki 2007) to the setting of active integrity constraints.

Next, there is the question whether a still narrower classes of repairs could be identified based on the analysis of all active integrity constraints (revision program rules) that would resolve conflicts among them (multiple possible repairs or revisions result precisely from the need to choose which constraint or rule to use when several are applicable) either based on their specificity (an approach used with some success in default logic) or on explicit rankings of the relative importance of active integrity constraints and revision rules.

Finally, we note that all the semantics discussed in the paper give rise to knowledge base operators that could be analyzed from the standpoint of Katsuno-Mendelzon postulates. To this end, we observe that we can view a set of databases as the set of models of some formula and so, as a knowledge base in the sense of Katsuno and Mendelzon (Katsuno and Mendelzon 1991). Let η be a set of active integrity constraints and, for the sake of illustration, let us focus our attention on the semantics of justified repairs. Given a set of databases (a knowledge base), $\mathcal{I}$, we can assign to it another set of databases (knowledge base), $\mathcal{I}'$, consisting of all η-justified repairs of all databases in $\mathcal{I}$. In that way we obtain a knowledge base update operator determined by η and the semantics justified repairs. It is an interesting problem to determine which of the Katsuno-Mendelzon postulates are satisfied by that operator (and by the other ones that arise by choosing a different update semantics).

Acknowledgments

The authors thank anonymous reviewers for many insightful comments that resulted in substantial improvements to the original manuscript. This work was partially supported by the NSF grants IIS-0325063 and IIS-0913459, and the KSEF grant KSEF-1036-RDE-008.

References

Arenas, M., Bertossi, L. E., and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems - PODS 1999*. ACM Press, 68–79.

Arenas, M., Bertossi, L. E., and Chomicki, J. 2003. Answer sets for consistent query answering in inconsistent databases. *TPLP 3,* 4-5, 393–424.

Bravo, L. and Bertossi, L. 2006. Semantically correct query answers in the presence of null values.

Caroprese, L., Greco, S., Sirangelo, C., and Zumpano, E. 2006. Declarative semantics of production rules for integrity maintenance. In *Proceedings of 22th International Conference on Logic Programming (ICLP-2006)*. Springer, 26–40.

Caroprese, L. and Truszczyński, M. 2008a. Declarative semantics for active integrity constraints. In *Proceedings of 24th International Conference on Logic Programming (ICLP-2008)*. Springer, 269–283.

Caroprese, L. and Truszczyński, M. 2008b. Declarative semantics for revision programming and connections to active integrity constraints. In *Proceedings of 11th European Conference on Logics in Artificial Intelligence (JELIA-2008)*. Springer, 100–112.

Chakravarthy, U. S., Grant, J., and Minker, J. 1990. Logic-based approach to semantic query optimization. *ACM Trans. Database Syst. 15,* 2, 162–207.

Chomicki, J. 2007. Consistent query answering: Five easy pieces. In *Proceedings of the 11th International Conference on Database Theory - ICDT 2007*. Lecture Notes in Computer Science, vol. 4353. Springer, 1–17.

Eiter, T. and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: propositional case. *Annals of Mathematics and Artificial Intelligence 15*, 289–323.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 365–385.

Greco, G., Greco, S., and Zumpano, E. 2003. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. Knowl. Data Eng. 15,* 6, 1389–1408.

Jagadish, H. V., Mendelzon, A. O., and Mumick, I. S. 1999. Managing conflicts between rules. *Journal of Computer and System Sciences 58,* 1, 13–28.

Katsuno, H. and Mendelzon, A. O. 1991. On the difference between updating a knowledge base and revising it. In *KR*. 387–394.

Lifschitz, V. and Woo, T. 1992. Answer sets in general nonmonotonic reasoning. In *Proceedings of 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR-1992)*. Springer, 603–614.

Lopatenko, A. and Bertossi, L. 2006. Consistent query answering by minimal-size repairs. *Database and Expert Systems Applications, International Workshop on 0*, 558–562.

Marek, W., Pivkina, I., and Truszczyński, M. 1999. Revision programming = logic programming + integrity constraints. In *Proceedings of 12th International Workshop on Computer Science Logic (CSL-1998)*. Springer, 73–89.

Marek, W. and Truszczyński, M. 1991. Autoepistemic logic. *Journal of the ACM 38*, 588–619.

Marek, W. and Truszczyński, M. 1994. Revision specifications by means of programs. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA-1994)*. Springer, 122–136.

Marek, W. and Truszczyński, M. 1998. Revision programming. *Theoretical Computer Science 190*, 241–277.

Marileo, M. C. and Bertossi, L. E. 2007. The consistency extractor system: Querying inconsistent databases using answer set programs. In *Proceedings of the 1st International Conference on Scalable Uncertainty Management - SUM 2007*, H. Prade and V. S. Subrahmanian, Eds. Lecture Notes in Computer Science, vol. 4772. Springer, 74–88.

McCarthy, J. and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, B. Meltzer and D. Michie, Eds. Edinburgh University Press, 463–502.

Pivkina, I. 2001. Revision programming: a knowledge representation formalism. Ph.D. thesis, Department of Computer Science, University of Kentucky.

Widom, J. and Ceri, S. 1996. Conclusions and future directions. In *Active Database Systems: Triggers and Rules For Advanced Database Processing*. Morgan Kaufmann, 293–302.

Winslett, M. 1990. *Updating Logical Databases*. Cambridge University Press.

Appendix

We present here the proofs of the two shifting theorems. The proofs are based on several auxiliary results.

Lemma 7
Let $\mathcal{W}$ be a database.

1. For every update action α, $T_{\mathcal{W}}(lit(\alpha)) = lit(T_{\mathcal{W}}(\alpha))$
2. For every set A of literals (update actions, active integrity constraints, respectively) $T_{\mathcal{W}}(T_{\mathcal{W}}(A)) = A$
3. For every consistent set $\mathcal{A}$ of literals (update actions, respectively), $T_{\mathcal{W}}(\mathcal{A})$ is consistent
4. For every databases $\mathcal{I}$ and $\mathcal{R}$, $T_{\mathcal{W}}(ne(\mathcal{I},\mathcal{R})) = ne(\mathcal{I} \div \mathcal{W}, \mathcal{R} \div \mathcal{W})$
5. For every active integrity constraint r, $nup(T_{\mathcal{W}}(r)) = T_{\mathcal{W}}(nup(r))$.

Proof: (1) - (3) follow directly from the definitions. We omit the details.

4. Let $\alpha \in ne(\mathcal{I} \div \mathcal{W}, \mathcal{R} \div \mathcal{W})$. If $\alpha = +a$, then it follows that $a \in (\mathcal{I} \div \mathcal{W}) \cap (\mathcal{R} \div \mathcal{W})$. Let us assume that $a \in \mathcal{W}$. Then $a \notin \mathcal{I} \cup \mathcal{R}$ and, consequently, $-a \in ne(\mathcal{I},\mathcal{R})$. Since $a \in \mathcal{W}$, $+a = T_{\mathcal{W}}(-a)$. Thus, $\alpha \in T_{\mathcal{W}}(ne(\mathcal{I},\mathcal{R}))$. The case when $\alpha = -a$ can be dealt with in a similar way. It follows that $ne(\mathcal{I} \div \mathcal{W}, \mathcal{R} \div \mathcal{W}) \subseteq T_{\mathcal{W}}(ne(\mathcal{I},\mathcal{R}))$.
   Let $\mathcal{I}' = \mathcal{I} \div \mathcal{W}$ and $\mathcal{R}' = \mathcal{R} \div \mathcal{W}$. Then $\mathcal{I} = \mathcal{I}' \div \mathcal{W}$, $\mathcal{R} = \mathcal{R}' \div \mathcal{W}$ and, by applying the inclusion we just proved to $\mathcal{I}'$ and $\mathcal{R}'$, we obtain
   $$ne(\mathcal{I},\mathcal{R}) = ne(\mathcal{I}' \div \mathcal{W}, \mathcal{R}' \div \mathcal{W}) \subseteq T_{\mathcal{W}}(ne(\mathcal{I}',\mathcal{R}'))\cdot$$
   Consequently,
   $$T_{\mathcal{W}}(ne(\mathcal{I},\mathcal{R})) \subseteq T_{\mathcal{W}}(T_{\mathcal{W}}(ne(\mathcal{I}',\mathcal{R}'))) = ne(\mathcal{I} \div \mathcal{W}, \mathcal{R} \div \mathcal{W})\cdot$$
   Thus, the claim follows.
5. Let $L \in nup(T_{\mathcal{W}}(r))$. We have $L \in body(T_{\mathcal{W}}(r))$ and $L^D \notin lit(head(T_{\mathcal{W}}(r))$. Clearly, $head(T_{\mathcal{W}}(r)) = T_{\mathcal{W}}(head(r))$ and $body(T_{\mathcal{W}}(r)) = T_{\mathcal{W}}(body(r))$. Thus, $L \in T_{\mathcal{W}}(body(r))$ and $L^D \notin T_{\mathcal{W}}(head(r))$. Consequently, $T_{\mathcal{W}}(L) \in body(r)$. Moreover, since $T_{\mathcal{W}}(L^D) = (T_{\mathcal{W}}(L))^D$, $(T_{\mathcal{W}}(L))^D \notin head(r)$. It follows that $T_{\mathcal{W}}(L) \in nup(r)$ and so, $L \in T_{\mathcal{W}}(nup(r))$. Hence, $nup(T_{\mathcal{W}}(r)) \subseteq T_{\mathcal{W}}(nup(r))$.
   Applying this inclusion to an active integrity constraint $s = T_{\mathcal{W}}(r)$, we

obtain $nup(r) \subseteq T_{\mathcal{W}}(nup(T_{\mathcal{W}}(r)))$. This, in turn, implies $T_{\mathcal{W}}(nup(r)) \subseteq T_{\mathcal{W}}(T_{\mathcal{W}}(nup(T_{\mathcal{W}}(r)))) = nup(T_{\mathcal{W}}(r))$. Thus, the equality $nup(T_{\mathcal{W}}(r)) = T_{\mathcal{W}}(nup(r))$ follows. □

Lemma 8
Let $\mathcal{I}$ and $\mathcal{W}$ be databases and let L be a literal or an update action. Then $\mathcal{I} \models L$ if and only if $\mathcal{I} \div \mathcal{W} \models T_{\mathcal{W}}(L)$.

Proof: ($\Rightarrow$) Let us assume that $\mathcal{I} \models L$. If $L = a$, where a is an atom, then $a \in \mathcal{I}$. There are two cases: $a \in \mathcal{W}$ and $a \notin \mathcal{W}$. In the first case, $a \notin \mathcal{I} \div \mathcal{W}$ and $T_{\mathcal{W}}(a) = \mathit{not}\, a$. In the second case, $a \in \mathcal{I} \div \mathcal{W}$ and $T_{\mathcal{W}}(a) = a$. In each case, $\mathcal{I} \div \mathcal{W} \models T_{\mathcal{W}}(a)$, that is, $\mathcal{I} \div \mathcal{W} \models T_{\mathcal{W}}(L)$.

The case $L = \mathit{not}\, a$, where a is an atom, is similar. First, we have that $a \notin \mathcal{I}$. If $a \in \mathcal{W}$ then $a \in \mathcal{I} \div \mathcal{W}$ and $T_{\mathcal{W}}(\mathit{not}\, a) = a$. If $a \notin \mathcal{W}$ then $a \notin \mathcal{I} \div \mathcal{W}$ and $T_{\mathcal{W}}(\mathit{not}\, a) = \mathit{not}\, a$. In each case, $\mathcal{I} \div \mathcal{W} \models T_{\mathcal{W}}(\mathit{not}\, a)$, that is, $\mathcal{I} \div \mathcal{W} \models T_{\mathcal{W}}(L)$.

($\Leftarrow$) Let us assume that $\mathcal{I} \div \mathcal{W} \models T_{\mathcal{W}}(L)$. Then, $(I \div \mathcal{W}) \div \mathcal{W} = \mathcal{I}$ and $T_{\mathcal{W}}(T_{\mathcal{W}}(L)) = L$. Thus, $\mathcal{I} \models L$ follows by the implication ($\Rightarrow$). □

Lemma 9
Let $\mathcal{I}$ and $\mathcal{W}$ be databases, and let $\mathcal{U}$ be a consistent set of update actions. Then $(\mathcal{I} \circ \mathcal{U}) \div \mathcal{W} = (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})$.

Proof: We note that since $\mathcal{U}$ is consistent, $T_{\mathcal{W}}(\mathcal{U})$ is consistent, too. Thus, *both* sides of the identity are well defined.

Let $a \in (\mathcal{I} \circ \mathcal{U}) \div \mathcal{W}$. If $+a \in T_{\mathcal{W}}(\mathcal{U})$, then $a \in (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})$. Thus, let us assume that $+a \notin T_{\mathcal{W}}(\mathcal{U})$. We have two cases.
Case 1: $a \notin \mathcal{W}$. From the definition of $T_{\mathcal{W}}$, $+a \notin \mathcal{U}$. Since $a \in (\mathcal{I} \circ \mathcal{U}) \div \mathcal{W}$, $a \in \mathcal{I} \circ \mathcal{U}$ and, consequently, $a \in \mathcal{I}$ and $-a \notin \mathcal{U}$. Thus, $a \in (\mathcal{I} \div \mathcal{W})$ and $-a \notin T_{\mathcal{W}}(\mathcal{U})$ (otherwise, as $T_{\mathcal{W}}(-a) = -a$, we would have $-a \in \mathcal{U}$). Consequently, $a \in (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})$.
Case 2: $a \in \mathcal{W}$. From the definition of $T_{\mathcal{W}}$, $-a \notin \mathcal{U}$. Since $a \in (\mathcal{I} \circ \mathcal{U}) \div \mathcal{W}$, $a \notin \mathcal{I} \circ \mathcal{U}$. Thus, $a \notin \mathcal{I}$ and $+a \notin \mathcal{U}$. It follows that $a \in \mathcal{I} \div \mathcal{W}$ and $-a \notin T_{\mathcal{W}}(\mathcal{U})$ (otherwise we would have $+a \in \mathcal{U}$, as $T_{\mathcal{W}}(-a) = +a$, in this case). Hence, $a \in (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})$.

If $a \notin (\mathcal{I} \circ \mathcal{U}) \div \mathcal{W}$, we reason similarly. If $-a \in T_{\mathcal{W}}(\mathcal{U})$, then $a \notin (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})$. Therefore, let us assume that $-a \notin T_{\mathcal{W}}(\mathcal{U})$. As before, there are two cases.
Case 1: $a \notin \mathcal{W}$ and thus $-a \notin \mathcal{U}$. Since $a \notin (\mathcal{I} \circ \mathcal{U}) \div \mathcal{W}$, $a \notin \mathcal{I} \circ \mathcal{U}$ and, consequently, $a \notin \mathcal{I}$ and $+a \notin \mathcal{U}$. Thus, $a \notin (\mathcal{I} \div \mathcal{W})$ and $+a \notin T_{\mathcal{W}}(\mathcal{U})$. Consequently, $a \notin (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})$.
Case 2: $a \in \mathcal{W}$ and thus $+a \notin \mathcal{U}$. In this case, $a \in \mathcal{I} \circ \mathcal{U}$. Thus, $a \in \mathcal{I}$ and $-a \notin \mathcal{U}$. It follows that $a \notin \mathcal{I} \div \mathcal{W}$ and $+a \notin T_{\mathcal{W}}(\mathcal{U})$. Hence, $a \notin (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})$. □

Lemma 10
Let $\mathcal{I}$ and $\mathcal{W}$ be databases, $\mathcal{U}$ a consistent set of update actions, and L a literal or an action update. Then $\mathcal{I} \circ \mathcal{U} \models L$ if and only if $(\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U}) \models T_{\mathcal{W}}(L)$. □

Proof: By Lemma 8, $\mathcal{I} \circ \mathcal{U} \models L$ if and only if $(\mathcal{I} \circ \mathcal{U}) \div \mathcal{W} \models T_{\mathcal{W}}(L)$. By Lemma 9, the latter condition is equivalent to the condition $(\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U}) \models T_{\mathcal{W}}(L)$. □

Lemma 11
Let $\mathcal{I}$ and $\mathcal{W}$ be databases. For every set η of active integrity constraints and for every set $\mathcal{U}$ of update actions, $\mathcal{U}$ is a justified action set for $\langle \mathcal{I}, \eta \rangle$ if and only if $T_{\mathcal{W}}(\mathcal{U})$ is a justified action set for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$.

Proof: ($\Rightarrow$) We have to prove that $T_{\mathcal{W}}(\mathcal{U})$ is consistent, and minimal among all supersets of $ne(\mathcal{I} \div \mathcal{W}, (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U}))$ that are closed under $T_{\mathcal{W}}(\eta)$.

Since $\mathcal{U}$ is a justified action set for $\langle \mathcal{I}, \eta \rangle$, $\mathcal{U}$ is consistent and $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) \subseteq \mathcal{U}$. The former implies that $T_{\mathcal{W}}(\mathcal{U})$ is consistent (cf. Lemma 7(1)). The latter implies that $ne(\mathcal{I} \div \mathcal{W}, (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})) \subseteq T_{\mathcal{W}}(\mathcal{U})$ (cf. Lemma 7(2) and 9).

Next, we prove that $T_{\mathcal{W}}(\mathcal{U})$ is closed under $T_{\mathcal{W}}(\eta)$. Let r be an active integrity constraint in $T_{\mathcal{W}}(\eta)$ such that $body(r)$ is consistent, $nup(r) \subseteq lit(T_{\mathcal{W}}(\mathcal{U}))$. Then, there exists $s \in \eta$ such that $r = T_{\mathcal{W}}(s)$. By Lemma 7(5), $nup(r) = T_{\mathcal{W}}(nup(s))$. As $T_{\mathcal{W}}(nup(s)) \subseteq lit(T_{\mathcal{W}}(\mathcal{U}))$, we have that $nup(s) \subseteq lit(\mathcal{U})$. Since $\mathcal{U}$ is closed under s, there exists $\alpha \in head(s)$ such that $\alpha \in \mathcal{U}$. Thus, we obtain that $T_{\mathcal{W}}(\alpha) \in T_{\mathcal{W}}(head(s)) = head(r)$, and that $T_{\mathcal{W}}(\alpha) \in T_{\mathcal{W}}(\mathcal{U})$. Consequently, $head(r) \cap T_{\mathcal{W}}(\mathcal{U}) \neq \emptyset$. It follows that $T_{\mathcal{W}}(\mathcal{U})$ is closed under r and so, also under $T_{\mathcal{W}}(\eta)$.

Finally, let us consider a set $\mathcal{V}$ of update actions such that $ne(\mathcal{I} \div \mathcal{W}, (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})) \subseteq \mathcal{V} \subseteq T_{\mathcal{W}}(\mathcal{U})$ and closed under $T_{\mathcal{W}}(\eta)$. By Lemma 7(2) and 9, $ne(\mathcal{I} \div \mathcal{W}, (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{U})) = T_{\mathcal{W}}(ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$. Thus, $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) \subseteq T_{\mathcal{W}}(\mathcal{V}) \subseteq \mathcal{U}$. From the fact that $\mathcal{V}$ is closed under $T_{\mathcal{W}}(\eta)$ it follows that $T_{\mathcal{W}}(\mathcal{V})$ is closed under η (one can show it reasoning similarly as in the previous paragraph). As $\mathcal{U}$ is minimal in the class of supersets of $ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ closed under η, $T_{\mathcal{W}}(\mathcal{V}) = \mathcal{U}$ and so, $\mathcal{V} = T_{\mathcal{W}}(\mathcal{U})$. This completes the proof of the implication ($\Rightarrow$).

($\Leftarrow$) If $T_{\mathcal{W}}(\mathcal{U})$ is a justified action set for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$, the implication ($\Rightarrow$) yields that $T_{\mathcal{W}}(T_{\mathcal{W}}(\mathcal{U})) = \mathcal{U}$ is a justified action set for $\langle (\mathcal{I} \div \mathcal{W}) \div \mathcal{W} = \mathcal{I}, \eta \rangle$. □

Proof of Theorem 20:

1. Let us assume that $\mathcal{E}$ is a weak repair for $\langle \mathcal{I}, \eta \rangle$. It follows that $\mathcal{E}$ is consistent. Since $\mathcal{I} \circ \mathcal{E} \models \eta$, by Lemma 10, $(\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{E}) \models T_{\mathcal{W}}(\eta)$. The converse implication follows from the one we just proved by Lemma 7(2).
2. As before, it suffices to show only one implication. Let $\mathcal{E}$ be a repair for $\langle \mathcal{I}, \eta \rangle$. Then, $\mathcal{E}$ is a weak repair for $\langle \mathcal{I}, \eta \rangle$. By (1), $\mathcal{E}$ is a weak repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$. Let $\mathcal{E}' \subseteq T_{\mathcal{W}}(\mathcal{E})$ be such that $(\mathcal{I} \div \mathcal{W}) \circ \mathcal{E}' \models T_{\mathcal{W}}(\eta)$. It follows that $T_{\mathcal{W}}(\mathcal{E}') \subseteq T_{\mathcal{W}}(T_{\mathcal{W}}(\mathcal{E})) = \mathcal{E}$. Since $\mathcal{E}$ is consistent, $T_{\mathcal{W}}(\mathcal{E}')$ is consistent, too. By Lemma 10 and Lemma 7(2), since $(\mathcal{I} \div \mathcal{W}) \circ \mathcal{E}' \models T_{\mathcal{W}}(\eta)$, then $\mathcal{I} \circ T_{\mathcal{W}}(\mathcal{E}') \models \eta$. Since $\mathcal{E}$ is a repair and $T_{\mathcal{W}}(\mathcal{E}') \subseteq \mathcal{E}$, $T_{\mathcal{W}}(\mathcal{E}') = \mathcal{E}$. Thus, $\mathcal{E}' = T_{\mathcal{W}}(\mathcal{E})$ and so, $T_{\mathcal{W}}(\mathcal{E})$ is a repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$.
3. As in two previous cases, we show only one implication. Thus, let us assume that $\mathcal{E}$ is founded for $\langle \mathcal{I}, \eta \rangle$. Let $\alpha \in T_{\mathcal{W}}(\mathcal{E})$. It follows that there is $\beta \in \mathcal{E}$ such that $\alpha = T_{\mathcal{W}}(\beta)$. Since $\mathcal{E}$ is founded with respect to $\langle \mathcal{I}, \eta \rangle$, there is an active integrity constraint r such that $\beta \in head(r)$, $\mathcal{I} \circ \mathcal{E} \models nup(r)$, and for every $\gamma \in head(r) \setminus \{\beta\}$, $\mathcal{I} \circ \mathcal{E} \models \gamma^D$.
Clearly, the active integrity constraint $T_{\mathcal{W}}(r)$ belongs to $T_{\mathcal{W}}(\eta)$ and $\alpha = T_{\mathcal{W}}(\beta)$ is an element of $head(T_{\mathcal{W}}(r))$. By Lemma 7(5), we have $nup(r) =$

$nup(T_{\mathcal{W}}(r))$. Thus, by Lemma 10, $(\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{E}) \models nup(T_{\mathcal{W}}(r))$. Next, let $\gamma \in head(T_{\mathcal{W}}(r)) \setminus \{\alpha\}$. Then, there is $\delta \in head(r) \setminus \{\beta\}$ such that $\gamma = T_{\mathcal{W}}(\delta)$. Since $\mathcal{I} \circ \mathcal{E} \models \gamma^D$, it follows that $(\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{E}) \models T_{\mathcal{W}}(\delta^D)$, that is, $(\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{E}) \models \gamma^D$. Thus, α is founded with respect to $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta)\rangle$ and $T_{\mathcal{W}}(\mathcal{E})$ and $T_{\mathcal{W}}(\mathcal{E})$ is founded with respect to $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta)\rangle$.
4. This property is a direct consequence of (1), (2), and (3).
5. If $\mathcal{E}$ is a justified weak repair for $\langle \mathcal{I}, \eta\rangle$, then $\mathcal{E} \cap ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}) = \emptyset$ and $\mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ is a justified action set for $\langle \mathcal{I}, \eta\rangle$ (Theorem 1). It follows that $T_{\mathcal{W}}(\mathcal{E}) \cap T_{\mathcal{W}}(ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})) = \emptyset$. Moreover, by Lemma 11, $T_{\mathcal{W}}(\mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}))$ is a justified action set for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta)\rangle$.
   We have $T_{\mathcal{W}}(ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})) = ne(\mathcal{I} \div \mathcal{W}, (\mathcal{I} \div \mathcal{W}) \circ T_{\mathcal{W}}(\mathcal{E}))$. Thus, again by Theorem 1, $T_{\mathcal{W}}(\mathcal{E})$ is a justified weak repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta)\rangle$.
   If $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta\rangle$, then our argument shows that $T_{\mathcal{W}}(\mathcal{E})$ is a justified weak repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta)\rangle$. Moreover, since $\mathcal{E}$ is a repair for $\mathcal{I}$, by Theorem 20(2) we have that $T_{\mathcal{W}}(\mathcal{E})$ is a repair for $\mathcal{I} \div \mathcal{W}$. It follows that $T_{\mathcal{W}}(\mathcal{E})$ is a justified repair for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta)\rangle$. The other implication can now be argued in the same way as in several other similar cases in the paper. □

Proof of Corollary 4: The assertion follows directly from Theorem 20. □

Next we turn to the shifting properties of revision programs. We will derive Theorem 21 from Theorem 20. To this end we need one more lemma.

Lemma 12
Let $\mathcal{I}$ and $\mathcal{W}$ be databases, $\mathcal{E}$ a set of revision literals, G a revision program and P a proper revision program. Then $T_{\mathcal{W}}(prop(G)) = prop(T_{\mathcal{W}}(G))$, $T_{\mathcal{W}}(ua(\mathcal{E})) = ua(T_{\mathcal{W}}(\mathcal{E}))$ and $T_{\mathcal{W}}(AIC(P)) = aic(T_{\mathcal{W}}(P))$.

Proof: Straightforward from the definitions of $prop(\cdot)$, $T_{\mathcal{W}}(\cdot)$, $ua(\cdot)$ and $AIC(\cdot)$. □

Proof of Theorem 21: Let $P = prop(G)$ (that is the "properized" version of G). The following properties are equivalent:

1. $\mathcal{E}$ is a (weak) revision for $\mathcal{I}$ with respect to G (respectively, G-justified (weak) revision for $\mathcal{I}$, G-founded (weak) revision for $\mathcal{I}$)
2. $\mathcal{E}$ is a (weak) revision for $\mathcal{I}$ with respect to P (respectively, P-justified (weak) revision for $\mathcal{I}$, P-founded (weak) revision for $\mathcal{I}$)
3. $ua(\mathcal{E})$ is a (weak) repair (respectively, justified (weak) repair, founded (weak) repair) for $\langle \mathcal{I}, AIC(P)\rangle$
4. $T_{\mathcal{W}}(ua(\mathcal{E}))$ is a (weak) repair (respectively, justified (weak) repair, founded (weak) repair) for $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(AIC(P))\rangle$
5. $T_{\mathcal{W}}(\mathcal{E})$ is a (weak) revision for $\mathcal{I} \div \mathcal{W}$ with respect to $T_{\mathcal{W}}(P)$ (respectively, $T_{\mathcal{W}}(P)$-justified (weak) revision for $\mathcal{I} \div \mathcal{W}$, $T_{\mathcal{W}}(P)$-founded (weak) revision for $\mathcal{I} \div \mathcal{W}$)
6. $T_{\mathcal{W}}(\mathcal{E})$ is a (weak) revision for $\mathcal{I} \div \mathcal{W}$ with respect to $T_{\mathcal{W}}(G)$ (respectively, $T_{\mathcal{W}}(G)$-justified (weak) revision for $\mathcal{I} \div \mathcal{W}$, $T_{\mathcal{W}}(G)$-founded (weak) revision for $\mathcal{I} \div \mathcal{W}$).

Indeed, (1) and (2) are equivalent by Theorem 13, (2) and (3) are equivalent by Theorem 14, (3) and (4) — by Theorems 6 and 7 of [8] (*the shifting theorem for (weak) repairs, founded (weak) repairs and justified (weak) repairs*). Next, (4) and (5) are equivalent by Theorem 14, as well as Lemma 12, and (5) and (6) — by Theorem 13 and Lemma 12. Thus, the assertion follows. □