Declarative Semantics for Active Integrity Constraints

Luciano Caroprese[1] and Mirosław Truszczyński[2]

[1] Università della Calabria, 87030 Rende, Italy
[2] Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA

Abstract. We study *active integrity constraints*, a formalism designed to describe integrity constraints on databases *and* to specify preferred ways to enforce them. The original semantics proposed for active integrity constraints is based on the concept of a *founded repair*. We point out that groundedness underlying founded repairs does not prevent cyclic justifications and so, may be inappropriate in some applications. Thus, using a different notion of grounding, with roots in logic programming and revision programming, we introduce two new semantics: of *justified weak repairs*, and of *justified repairs*. We study properties of these semantics, relate them to earlier semantics of active integrity constraints, and establish the complexity of basic decision problems.

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 preferred ways to update databases.

A database can be viewed formally 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* is 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.

Let us consider the database $\mathcal{I} = \{a, b\}$ and the integrity constraint $\neg a \vee \neg b$. As, $\mathcal{I}$ does not satisfy $\neg a \vee \neg b$, it 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 (cf. [1]). 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, we study a recent formalism of that type: *active integrity constraints* (*aic*'s, for short) [2].

Aic's *explicitly* encode both integrity constraints and preferred basic actions to repair them, when the constraints are violated. To specify the meaning of sets of aic's, [2] proposed the concept of *foundedness*, and combined it with the requirement of the minimality of change to get the semantics of *founded repairs*. Foundedness reflects a certain groundedness condition. We show that in some cases this groundedness condition is too weak to prevent *cyclic* justifications. To address the problem, we introduce a new semantics for aic's, which we call the semantics of *justified repairs*.

The semantics of *justified repairs* uses a stronger concept of groundedness than that behind founded repairs. In general, it is also too weak to imply the minimality of change property and so, we impose this property on justified repairs *explicitly*. We show that the class of justified repairs is a subclass of the class of founded repairs.

We also consider a broader class of ways to enforce integrity constraints by dropping the minimality of change postulate. We refer to the elements of that class as *weak repairs*. Combining the concept with the appropriate groundedness condition yields the semantics of *founded weak repairs* and *justified weak repairs*. While the minimality of change condition is a natural requirement to impose on preferred ways to enforce integrity constraints, including weak repairs in the considerations offers a richer perspective. In particular, it brings up the question of identifying classes of aic's, for which the groundedness condition alone is sufficient to guarantee change-minimality. We exhibit here two classes of aic's, for which the groundednes condition behind justified repairs ensures the minimality of change.

A fundamental property of semantics describing database updates is the invariance under *shifting* [3, 4]. Informally, shifting consists of changing the membership status of some facts in a database and the corresponding modification (systematic renaming) of the integrity constraints. We show that all semantics of aic's we consider here are invariant under shifting.

Although we consider just the propositional case our results can be easily lifted to the first-order case via grounding (like in the case of stable-model semantics for Logic Programming). As a consequence, our framework is able to handle numerical expressions in the body of the constraints. Indeed, grounding eliminates them and leaves us with the basic propositional case presented here.

A richer field of semantics of database updates gives rise to a trade-off. On the one hand the semantics differ in how much they refine the class of (weak) repairs that enforce integrity constraints, and on the other hand, in the complexity of the existence of a repair problem. We discuss ways in which this trade-off can be exploited in practice.

The semantics discussed in the paper are motivated by connections to the semantics of answer sets for disjunctive logic programming on the one hand, and to revision programming [3], on the other. We exploit them to develop the definition of groundedness for justified weak repairs, and to establish the complexity of deciding the existence of repairs of particular types. We develop a detailed discussion of these connections in another paper [5]. Due to space limits, we omit the proofs (they can be found at `www.cs.uky.edu/ai/aic-full.pdf`).

2 Integrity Constraints and Database Repairs — Basic Concepts

We consider a finite set At of propositional atoms. We represent databases as subsets of At. 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 actions $+a$ and $-b$ state that a and b are to be inserted and deleted, respectively. 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 $\mathcal{D}$ be a database and $\mathcal{U}$ a consistent set of update actions. The result of *updating $\mathcal{D}$ by means of $\mathcal{U}$* is the database

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

We have the following straightforward property of the operator $\circ$, which asserts that if a set o 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 $\mathcal{I}$, $\mathcal{I} \circ (\mathcal{U}_1 \cup \mathcal{U}_2) = (\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2$.* □

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 (expressions of the form a and $not\, a$, where a is an atom) 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. 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. In the context of aic's the subject of this paper, the negation operator has some similarities to the default negation operator in logic programming and so, as it is common in the logic programming literature, we denote it with not rather than $\neg$.

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

Definition 1. [WEAK REPAIR AND REPAIR] *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).* □

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 replacing $\mathcal{I}$ with a weak repair $\mathcal{I}'$ 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. However, assuming consistency of integrity constraints does not yield any significant simplifications. Therefore, we do not make this assumption here.

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). Since repairs exist if and only if weak repairs do, the problem of the existence of a repair is NP-complete, too.

3 Active Integrity Constraints

Given no other information but 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* (*aic*'s, for short) [2] was designed to address this problem. We will now review it and offer a first extension by introducing the semantics of founded weak repairs.

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 to sets of literals and sets of update actions, as appropriate.

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 $head(r)$.

An aic 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 aic with a non-empty head functions as an integrity constraint (its body must be false) *and* it explicitly provides support for the use of update actions in its head (if its body is true).

The role of the condition (3) is to ensure that an aic 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 aic 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.

A database $\mathcal{I}$ satisfies an aic r, $\mathcal{I} \models r$, if it satisfies the corresponding integrity constraint. It is now straightforward to adapt the concept of a (weak) repair to the case of aic's. Specifically, a set $\mathcal{U}$ of update actions is a *(weak) repair* for a database $\mathcal{I}$ with respect to a set η of aic's if it is a (weak) repair for $\mathcal{I}$ with respect to the set of *standard* integrity constraints represented by η.

Let us consider the aic $r = a, b \supset -b$, 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). We select the latter as a preferred repair, since r provides support for the update action $-b$.

Repairs do not need to obey preferences expressed by the heads of aic's. To formalize the notion of "support" and translate it into a method to select "preferred" repairs, [2] proposed the concept of a *founded repair* — a repair that is *grounded* in ("implied" by) a set of aic's. The following definition, in addition to founded repairs, introduces a new semantics of founded *weak* repairs.

Definition 2. [FOUNDED (WEAK) REPAIR] *Let $\mathcal{I}$ be a database, η a set of aic's, 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 $\mathcal{I} \circ \mathcal{U} \models \beta^D$, 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 $\langle \mathcal{I}, \eta \rangle$.* □

Foundedness is indeed a formalization of a certain notion of "groundedness". Let us assume that α is founded with respect to $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$ by means of an aic $r \in \eta$. 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, if $\mathcal{U}$ is to enforce r, it must contain α. We observe that the foundedness does not imply the constraint enforcement nor the minimality of change.

Example 1. Let $\mathcal{I} = \emptyset$ and η consist of the following aic's:

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

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 it is *not* change-minimal. The set $\{+b, +c\}$ is founded but does not guarantee constraint enforcement. Therefore, in the definition of founded (weak) repairs, the property of being a (weak) repair must be enforced explicitly. 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. □

Next, we show that there could exist founded *weak* repairs even when no founded repair exists.

Example 2. Let $\mathcal{I} = \emptyset$ and η consist of the following aic's:

$$\begin{array}{ll} not\ a, b, c \supset +a & not\ b, a, c \supset +b \\ not\ c, a, b \supset +c & not\ a \quad \supset \bot \end{array}$$

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. □

Finally, we discuss the key issue arising in the context of founded repairs that motivates much of the remainder of the paper. 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:

$$\begin{array}{ll} r_1 = a, b & \supset -a \\ r_2 = a, not\ b & \supset -a \\ r_3 = not\ a, b & \supset -b. \end{array}$$

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 necessary support, while $-b$ is founded with respect to $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}$ because of 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 forcing the need for a repair. However, according to intuitions we discussed earlier, r_1 supports the foundedness of $-a$ *only if* b remains in the database. This is not the case here. Thus, the support for the foundedness of $-a$ in $\mathcal{U}$ must come entirely from r_2 and r_3. The same holds for $-b$ as it is not even mentioned in the head of r_1.

It follows that the foundedness of $-a$ is supported solely by r_2, and it *requires* that $-b$ be included in the repair. In the same way, the foundedness of $-b$ is supported solely by r_3, and it depends on $-a$ being included in the repair. Thus, 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 $\mathcal{U}$, but 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 $-b$. □

To summarize this section, the semantics of repairs for aic's 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 so the associated concept of groundedness is weak. We revisit this issue in the next section.

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 [2]. For the sake of completeness, we also note that the problem of the existence of a founded weak repair is again "only" NP-complete (the proof is simple and we omit it).

4 Justified repairs

In this section, we will introduce another semantics for aic's 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 aic's. Let η be a set of aic's 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 $head(r)$.

Definition 3. [CLOSED SET 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 aic's if it is closed under every $r \in \eta$.* □

If a set of update actions is not closed under a set η of aic's, executing its elements may fail to enforce constraints. Therefore, closed sets of update actions are important. We regard *minimal* such sets 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).

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 database $\mathcal{R}$, $\mathcal{R} \circ ne(\mathcal{I}, \mathcal{R}) = \mathcal{R}$*
2. *For every set $\mathcal{E}$ of update actions such that $\mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E})$ is consistent, $\mathcal{I} \circ \mathcal{E} = \mathcal{I} \circ (\mathcal{E} \cup ne(\mathcal{I}, \mathcal{I} \circ \mathcal{E}))$.* □

Our semantics of justified repairs is based on the knowledge-representation principle (a form of the frame axiom) 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 4. [JUSTIFIED WEAK REPAIR] *Let $\mathcal{I}$ be a database and η a set of aic's. 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.

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 aic's and $\mathcal{E}$ a consistent set of update actions. Then $\mathcal{E}$ is a 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 $\langle \mathcal{I}, \eta \rangle$.* □

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 aic's, and $\mathcal{E}$ a justified weak repair for $\langle \mathcal{I}, \eta \rangle$. Then*

1. *For every atom a, exactly one of $+a$ or $-a$ 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 $\langle \mathcal{I}, \eta \rangle$.* □

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

Corollary 1. *Let $\mathcal{I}$ be a database, η a set of aic's, and $\mathcal{E}$ a justified weak repair for $\langle \mathcal{I}, \eta \rangle$. Then, $\mathcal{E}$ is a founded weak repair for $\langle \mathcal{I}, \eta \rangle$.* □

The converse to Corollary 1 does not hold. That is, there are founded weak repairs that are not justified weak repairs.

Example 4. The database and aic's from Example 3 illustrate the point. As we noted there, $\mathcal{U} = \{-a, -b\}$ is a founded repair. Thus, it is also a founded weak repair.

As pointed out, the support for the foundedness of $\mathcal{U}$ is circular. The semantics of justified weak repairs resolves the problem. Indeed, $\mathcal{U}$ 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. But it is not a minimal set of update actions containing

$ne(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ and closed under η. Indeed, $\emptyset$ has these two properties, too. Thus, the notion of groundedness employed by justified weak repairs is stronger.

In Example 3, the problem is caused by r_1. Let us consider a situation, 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. □

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

Example 5. Let $\mathcal{I}' = \emptyset$, and η_3 be a set of aic's consisting of

$$r_1 = not\, a, b \supset +a| - b$$
$$r_2 = a, not\, b \supset -a| + b$$

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 as $\mathcal{I}'$ is consistent with η_3, 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 5. [JUSTIFIED REPAIR] *Let $\mathcal{I}$ be a database and η a set of aic's. 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 aic's, and $\mathcal{E}$ a justified repair for $\langle \mathcal{I}, \eta \rangle$. Then, $\mathcal{E}$ is a founded repair for $\langle \mathcal{I}, \eta \rangle$.* □

Example 4 shows that the inclusion asserted by Corollary 2 is proper. Indeed, we argued there that $\{-a, -b\}$ is a founded repair but not a justified weak repair. Thus, $\{-a, -b\}$ is not a justified repair, either.

As illustrated by Example 5, in general, justified weak repairs form a proper subclass of justified 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 aic's 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 $\langle \mathcal{I}, \eta \rangle$.* □

This theorem concerns the case when each update action in the head of an aic, if executed, would change the status of the underlying atom in the database. For instance, if the initial database is empty and all update actions prescribed by aic's are insert actions, then justified weak repairs are guaranteed to be minimal and so, are justified repairs.

5 Normal Active Integrity Constraints and Normalization

An aic r is *normal* if $|head(r)| = 1$. We will now study properties of normal aic's. The next result shows 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 aic's. If $\mathcal{E}$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$ then $\mathcal{E}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$.* □

Next, we introduce the operation of *normalization* of aic's, which consists of eliminating disjunctions from the heads of rules. For an aic $r = \phi \supset \alpha_1 | \ldots | \alpha_n$, by r^n we denote the set of *normal* aic's $\{\phi \supset \alpha_1, \ldots, \phi \supset \alpha_n\}$. For a set η of aic's, we set $\eta^n = \bigcup_{r \in \eta} r^n$. It is shown in [6] 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 (weak) repairs.

Theorem 5. *Let $\mathcal{I}$ be a database and η a set of aic's.*

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 $\langle \mathcal{I}, \eta \rangle$.* □

The following example shows that the inclusion in the previous theorem is, in general, proper.

Example 6. Let us consider an empty database $\mathcal{I}' = \emptyset$, the set η_4 of aic's

$$\begin{array}{ll} r_1 = not\, a, not\, b \supset +a | + b \\ r_2 = a, not\, b \quad \supset +b \\ r_3 = not\, a, b \quad \supset +a \end{array}$$

its normalized version η_4^n

$$\begin{array}{ll} r_{1,1} = not\, a, not\, b \supset +a & \quad r_{2,1} = a, not\, b \supset +b \\ r_{1,2} = not\, a, not\, b \supset +b & \quad r_{3,1} = not\, a, b \supset +a \end{array}$$

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. □

6 Shifting Theorem

We will now study the concept of shifting [3]. Shifting 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 empty. In many cases it allows us to relate a semantics of database repairs to some semantics of logic programs with negation.

Example 7. 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 7 is not coincidental. In this section we present results showing that the semantics of (weak) repairs, founded (weak) repairs and justified (weak) repairs satisfy the shifting property. 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 of atom $\mathcal{W}$ (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 aic's. To this end, we introduce a *shifting* operator $T_{\mathcal{W}}$.

Definition 6. *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 aic $r = \phi \supset \psi$, we set $T_{\mathcal{W}}(r) = T_{\mathcal{W}}(\phi) \supset T_{\mathcal{W}}(\psi)$ and we extend the notation to sets aic's 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 aic $r = L_1, \ldots, L_n \supset \alpha_1| \ldots |\alpha_m$ we obtain

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

Clearly, we overload the notation $T_{\mathcal{W}}$ and interpret it based on the type of the argument. We have the following two results.

Theorem 6. [SHIFTING THEOREM FOR (WEAK) REPAIRS AND FOUNDED (WEAK) REPAIRS] *Let $\mathcal{I}$ and $\mathcal{W}$ be databases. For every set η of aic's 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 $\langle \mathcal{I} \div \mathcal{W}, T_{\mathcal{W}}(\eta) \rangle$.* □

Theorem 7. [SHIFTING THEOREM FOR JUSTIFIED (WEAK) REPAIRS] *Let $\mathcal{I}$ and $\mathcal{W}$ be databases. For every set η of aic's and for every set $\mathcal{E}$ of update actions, $\mathcal{E}$ 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 $\langle \mathcal{I}, T_{\mathcal{W}}(\eta) \rangle$.* □

Theorems 6 and 7 imply 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 the empty initial database. That property simplifies studies of these semantics as it allows us to eliminate one parameter (the initial database) from considerations.

Corollary 3. *Let $\mathcal{I}$ be a database and η a set of aic's. Then $\mathcal{E}$ is a weak repair (repair, weak 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, weak repair, founded weak repair, founded repair, justified weak repair, justified repair, respectively) for $\langle \emptyset, T_{\mathcal{I}}(\eta) \rangle$.* □

Example 8. Let us look at one of the instances of the database repair problem considered in Example 4, specifically, at $\langle \mathcal{I}, \eta_2 \rangle$. We recall that $\mathcal{I} = \{a, b\}$ and η_2 consists of the constraints:

$$\begin{array}{ll} a, b & \supset -a | -b \\ a, not\, b & \supset -a \\ not\, a, b & \supset -b. \end{array}$$

The set $\{-a, -b\}$ is the only weak repair for $\langle \mathcal{I}, \eta_2 \rangle$ and, as we noted earlier, it is a (weak) founded repair and a (weak) justified repair for $\langle \mathcal{I}, \eta_2 \rangle$, as well. Let us "shift" this instance to $\mathcal{I}' = \emptyset$, To this end, we shift with respect to $W = \mathcal{I} \div \mathcal{I}' = \{a, b\}$. One can check that $\emptyset = T_{\{a,b\}}(\{a, b\})$, that is, $\mathcal{I}' = T_W(\mathcal{I})$. Moreover, $T_W(\eta_2) = \eta_4$, where η_4 is the set of aic's considered in Example 6 above. Thus, indeed, by shifting $\langle \mathcal{I}, \eta_2 \rangle$ with respect to W, we obtain the database repair problem $\langle \mathcal{I}', \eta_4 \rangle$. It is easy to verify that $T_{\{a,b\}}(\{-a, -b\}) = \{+a, +b\}$ and that $\{+a, +b\}$ is the only (weak) repair for $\langle \mathcal{I}', \eta_4 \rangle$, which happens also to be a (weak) founded repair and a (weak) justified repair for $\langle \mathcal{I}', \eta_4 \rangle$, in agreement with the results of this section. □

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^2_P-complete [2]. 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 [7], and of the disjunctive logic program — Σ^P_2-complete [8]. The proofs in [7, 8] 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

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

We define

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

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 [9] for details). The following result states a property of the translation needed for hardness arguments.

Theorem 8. *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 $\langle \emptyset, aic(P) \rangle$.* □

Example 9. Let us consider Example 6. We observe that η_4 is equal to $aic(P)$ where P is the simple disjunctive logic program consisting of the rules: $\rho_1 = a \mid b$, $\rho_2 = b \leftarrow a$ and $\rho_3 = a \leftarrow b$.We know that $\mathcal{E} = \{+a, +b\}$ is the unique justified repair for $\langle \mathcal{I}', \eta_4 \rangle$, where $\mathcal{I}' = \emptyset$. Moreover, one can check that $M = \{a, b\}$, for which $\mathcal{E} = ua(M)$, is the unique answer set of P. Furthermore, since the instance $\langle I', \eta_4 \rangle$ is the result of shifting $\langle \mathcal{I}, \eta_2 \rangle$, also the repairs of $\langle \mathcal{I}, \eta_2 \rangle$ can be expressed in terms of answer sets of the disjunctive logic program $aic(P)$. This points to a general translation of instances of the database repair problem into disjunctive logic programs by combining shifting with the mapping aic. A detailed study of this relationship is a subject of a separate paper. □

We now state main results of the section.

Theorem 9. *Let $\mathcal{I}$ be a database and η a set of normal aic's. Then checking if there exists a justified repair (justified weak repair, respectively) for $\langle \mathcal{I}, \eta \rangle$ is an NP-complete problem.* □

Theorem 10. *Let $\mathcal{I}$ be a database and η a set of aic's. The problem of the existence of a justified weak repair for $\langle \mathcal{I}, \eta \rangle$ is a Σ^P_2-complete problem.* □

Theorem 11. *Let $\mathcal{I}$ be a database and η a set of aic's. The problem of the existence of a justified repair for $\langle \mathcal{I}, \eta \rangle$ is a Σ^P_2-complete problem.* □

8 Discussion

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.

$$
\begin{array}{ccccccccc}
 & & & & \mathbf{FR}(\mathcal{I}, \eta^n) & & & & \\
 & & & & \| & & & & \\
\mathbf{JR}(\mathcal{I}, \eta^n) & \subseteq & \mathbf{JR}(\mathcal{I}, \eta) & \subseteq & \mathbf{FR}(\mathcal{I}, \eta) & \subseteq & \mathbf{R}(\mathcal{I}, \eta) & = & \mathbf{R}(\mathcal{I}, \eta^n) \\
\| & & \cap\| & & \cap\| & & \cap\| & & \cap\| \\
\mathbf{JWR}(\mathcal{I}, \eta^n) & \subseteq & \mathbf{JWR}(\mathcal{I}, \eta) & \subseteq & \mathbf{FWR}(\mathcal{I}, \eta) & \subseteq & \mathbf{WR}(\mathcal{I}, \eta) & = & \mathbf{WR}(\mathcal{I}, \eta^n) \\
 & & & & \| & & & & \\
 & & & & \mathbf{FWR}(\mathcal{I}, \eta^n) & & & &
\end{array}
$$

Fig. 1. Relationships among classes of repairs

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 this set 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 — the existence problem for $\mathbf{JR}(\mathcal{I}, \eta)$ is Σ_P^2-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_2^P-complete and, then founded weak repairs for $\langle \mathcal{I}, \eta \rangle$, where the existence problem is NP-complete.

9 Conclusion

We studied the formalism of aic's [2], designed for enforcing integrity constraints on databases in the presence of preferences on alternative ways to do so. The original semantics proposed for aic's is based on the concept of a *founded repair*. Founded repairs are sets of update actions to be performed over the database in order to make it consistent. They are minimal w.r.t. change and supported by aic's. In some cases, elements of founded repairs cyclically support each other, which often is undesirable. Therefore, we introduced several new semantics for aic's. Two most important of them are the semantics of justified weak repairs and justified repairs. They are based on the concept of groundedness similar to that underlying the answer-set semantics of logic programs. We established the relationship of the two new semantics to that of founded repairs. For each semantics we determined the complexity of the basic existence of repair problem. Furthermore, we proved that each semantics satisfies the *shifting property*. Shifting consists of transforming an instance of a database repair problem to another syntactically isomorphic one by changing aic's to reflect the "shift" from the original database to the new one. These latter results are essential for relating repair formalism we studied with the formalism of Lifschitz-Woo programs [10], a subject of our future work.

Acknowledgments

This work was partially supported by the NSF grant IIS-0325063 and the KSEF grant KSEF-1036-RDE-008.

References

1. Winslett, M.: Updating Logical Databases. Cambridge University Press (1990)
2. Caroprese, L., Greco, S., Sirangelo, C., Zumpano, E.: Declarative semantics of production rules for integrity maintenance. In Etalle, S., Truszczynski, M., eds.: Logic Programming, 22nd International Conference, ICLP 2006, Proceedings. Volume 4079 of LNCS., Springer (2006) 26–40
3. Marek, W., Truszczyński, M.: Revision programming. Theoretical Computer Science **190** (1998) 241–277
4. Pivkina, I.: Revision programming: a knowledge representation formalism. PhD thesis, Department of Computer Science, University of Kentucky (2001) http://lib.uky.edu/ETD/ukycosc2001d00022/pivkina.pdf.
5. Caroprese, L., Truszczyński, M.: Declarative Semantics for Revision Programming and Connections to Active Integrity Constraints (2008) Submitted.
6. Caroprese, L., Greco, S., Zumpano, E.: Active integrity constraints for database consistency maintenance (2008) Manuscript, submitted to IEEE TKDE.
7. Marek, W., Truszczyński, M.: Autoepistemic logic. Journal of the ACM **38** (1991) 588–619
8. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: propositional case. Annals of Mathematics and Artificial Intelligence **15** (1995) 289–323
9. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9** (1991) 365–385
10. Lifschitz, V., Woo, T.: Answer sets in general nonmonotonic reasoning. In: Proceedings of the 3rd international conference on principles of knowledge representation and reasoning, KR '92, San Mateo, CA, Morgan Kaufmann (1992) 603–614