Active Integrity Constraints and Operational Aspects of the Problem of Database Repair

LUÍS CRUZ-FILIPE

Department of Mathematics and Computer Science, University of Southern Denmark
(*e-mail:* lcf@imada.sdu.dk)

PATRÍCIA ENGRÁCIA

DGEEC, Ministério da Ciência, Tecnologia e Ensino Superior, Ministério da Educação, Lisboa
Universidade Aberta, Lisboa
(*e-mail:* p.engracia@gmail.com)

GRAÇA GASPAR, ISABEL NUNES

BioISI—Biosystems & Integrative Sciences Institute, Faculty of Sciences, Univ. Lisbon
(*e-mail:* {gg,in}@di.fc.ul.pt)

PETER SCHNEIDER-KAMP

Department of Mathematics and Computer Science, University of Southern Denmark
(*e-mail:* petersk@imada.sdu.dk)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

Repairing an inconsistent database is a well-known problem for which several partial approaches have been proposed and implemented in the past. One of the existing proposals uses active integrity constraints (AICs), which are consistency requirements that also allow one to restrict the possible ways to repair inconsistencies. The typical semantics for AICs is declarative, and thus suggests no algorithms to compute repairs.

In this article, we show how AICs may be used directly to repair inconsistent databases in an efficient way. First, we introduce notions of independence and precedence on AICs that allow the search for repairs to be, respectively, parallelized and sequentialized. Afterwards, we present an operational semantics and show that the different kinds of repairs for AICs can be effectively computed as leaves of specific kinds of trees. We show that this operational semantics is complete when existence of a repair is an *NP*-complete problem. Finally we discuss the applicability of these techniques and results to integrity constraints in general.

KEYWORDS: active integrity constraints, operational semantics, database repair

1 Introduction

Maintaining consistency of knowledge bases is a major challenge in the face of growing size and complexity of automated reasoning systems. In the particular field of relational databases, this problem has been the focus of intensive research for over thirty years. Redundancies in the data being stored give rise to semantic dependencies that have to be enforced at all times. These dependencies are formalized as logical formulas that are traditionally called integrity constraints (Abiteboul et al. 1995).

An early step in systematizing the work on integrity constraints was the classification, by Beeri and Vardi (1981), of the types of constraints most typically found. These authors identified in particular three main types of such dependencies. A *tuple-generating dependency* is a first-order formula $\forall X.(\varphi(X) \to \exists Y.\psi(X,Y))$, where φ and ψ are conjunctions of relation atoms. Intuitively, if the database satisfies $\varphi(X)$, then it must also satisfy $\psi(X,Y)$ for some Y – if this is not the case, a new such tuple has to be generated and added to the database. The particular case where ψ is the empty conjunction of atoms yields *negative constraints* $\forall X.\,(\varphi(X) \to \bot)$, which simply state that $\varphi(X)$ can never hold. An *equality-generating dependency* has the form $\forall X.(\varphi(X) \to (x_i = x_j))$, where $x_i, x_j \in X$. Although logically this fomula is a particular case of tuple-generating dependencies, its meaning is slightly different, as it requires identifying two objects in the database. A more comprehensive overview of different classes of integrity constraints used in practice was given by Thalheim (1991).

As the intuition given above suggests, whenever an integrity constraint is violated, the database must be altered in order to regain consistency. Traditionally, there are three types of update actions that can be used for this purpose (Abiteboul 1988): insertion of new facts, deletion of existing facts, and modification of tuples. A set of update actions that restores consistency of the database is called a *repair*. Intuitively, one can compute repairs by reading integrity constraints as rules that suggest applicable update actions (Abiteboul 1988). Throughout the years, several algorithms for computing repairs of inconsistent databases have been proposed and studied, focusing on the different ways integrity constraints are specified and on the different types of databases under consideration (Kakas and Mancarella 1990; Marek and Truszczynski 1995; Mayol and Teniente 1999b; Naqvi and Krishnamurthy 1988; Przymusinski and Turner 1997). This multitude of approaches is not an accident: deciding whether an inconsistent database can be repaired is typically a Π_2^P- (or co-Σ_2^P-) complete problem, and there is no reason to believe in the existence of general-purpose algorithms for this problem, but one should rather focus on developing specific algorithms for particular interesting cases (Eiter and Gottlob 1992).

A particularly successful algorithmic approach uses event-condition-action rules (Teniente and Olivé 1995; Widom and Ceri 1996), where actions are triggered by specific events, for which a procedural semantics has been defined. This approach is widely used in practice in database management systems in the form of active rules. However, the lack of declarative semantics for event-condition-action rules makes it difficult to understand their joint behaviour when several rules act together and to evaluate, in a principled way, the rule-processing algorithms.

Active integrity constraints (AICs) were introduced by Flesca et al. (2004) to address this issue. They are similar to event-condition-action rules in that they encode both an integrity constraint and preferred update actions to be performed whenever the former is violated, but their semantics, as given by Caroprese and Truszczyński (2011), is declarative. This semantics defines several interesting classes of repairs, and allows for an elegant theoretical understanding of their interactions. Although AICs are not able to express the plethora of integrity constraints defined in the literature, they are still expressive enough for most practical applications. In particular, they can capture tuple-generating dependencies, negative constraints, equality-generating dependencies, and, more generally, any integrity constraint that can be written in denial clausal form.

The declarative semantics of AICs inherently enforces two properties that are generally accepted as desirable of repairs: minimality of change (Chomicki 2007; Eiter and Gottlob 1992; Winslett 1990) – one should change as little as possible – and the common sense law of inertia (see e.g. Przymusinski and Turner 1997) – one should only change something if there is a reason for it. Even with these restrictions, there are typically many possible repairs for a given inconsistent database, and it is usually assumed that some human interaction will be required to choose the "best" possible repair (Teniente and Olivé 1995). The different classes of repairs defined in the semantics for AICs help in establishing a "preference" hierarchy between repairs that can be used to help with this task (Caroprese and Truszczyński 2011).

The major drawback of AICs is their exclusively declarative semantics, which hinders the computation of repairs and, thus, their use in practice. In this work, we endow AICs with an equivalent operational semantics that computes the same classes of repairs for inconsistent databases. We develop algorithms to compute the different classes of repairs, and show that these algorithms have optimal complexity. All algorithms have been implemented in a proof-of-concept prototype for repairing SQL-based relational databases, described in (Cruz-Filipe et al. 2016). This work also establishes the foundation for a generalization of AICs to general-purpose knowledge bases (Cruz-Filipe et al. 2016).

Publication history. The propositional versions of the results in this article have been published in conferences as (Cruz-Filipe et al. 2013) and (Cruz-Filipe 2014). The discussion of the general case is novel, and required adapting the proofs of most results. A proof-of-concept tool implementing the algorithms described herein has been developed and is described in (Cruz-Filipe et al. 2016).

Contribution. This article departs from the previous publications by generalizing the results from (Cruz-Filipe et al. 2013; Cruz-Filipe 2014) to the full first-order language of AICs, addressing the new problems that arise in this more expressive context.

The original definitions of AICs, due to Flesca et al. (2004) are set in a realistic database scenario, in the sense that they are built upon a first-order language. However, later work on this topic, namely dealing with the semantics – the classes of founded and justified repairs – focused on a simpler scenario, where the underlying logic is propositional (Caroprese and Truszczyński 2011). In this work, we again consider a first-order signature, and we address the following two problems.

- Can we split a set of AICs into several components that can be processed independently (in parallel or at least sequentially)?
- Can we compute founded or justified repairs directly using the language of AICs and the database? Previous work by Caroprese and Truszczyński (2011) showed that such repairs could be computed as semantics of suitably defined revision programs, but did not discuss how to compute them directly.

In the propositional setting, we gave positive answers to both these questions in earlier work (Cruz-Filipe et al. 2013; Cruz-Filipe 2014). However, the constructions we proposed transfer to the first-order case only partially.

In what concerns splitting a set of AICs into independent sets, a direct translation of previous work would require explicitly generating all closed instances of every constraint,

which is unrealistic in practice. We therefore introduce heuristics and sufficient conditions to compute approximate partitions of a set of AICs in an efficient way.

Regarding the second problem, we earlier introduced tree algorithms for computing different types of repairs directly in the propositional scenario; by restricting the construction of descendant nodes in particular ways, we showed that we obtained trees that produced specific kinds of repairs. Though these tree algorithms can be straightforwardly adapted to a realistic database setting, their complexity changes due to the presence of variables in rules.

Some of these results also apply in the setting of general integrity constraints, thereby extending previous work on algorithms for regaining database consistency. We point these out at the end of each section.

Structure The structure of this article is as follows. Section 2 introduces the framework of AICs. Sections 3 and 4 introduce the hierarchization and stratification mechanisms that allow repairs to be computed modularly. Section 5 introduces the operational semantics for AICs, as well as a new type of repair motivated by the analysis of this semantics. Section 6 discusses our results, reviews some further related work, and shortly presents some extensions and generalizations of our contributions. Section 7 summarizes the achievements described in this paper.

2 Active integrity constraints

Active integrity constraints (AICs) were first proposed by Flesca et al. (2004), and their declarative semantics – *founded* and *justified* (weak) repairs, defined below – was studied by Caroprese et al. (2006) and Caroprese and Truszczyński (2011), together with the connection with revision programming. In particular, founded/justified repairs can be computed as answer sets of particular revision programs. Although this connection was only studied in a propositional context, AICs were introduced in the more general setting of first-order logic, and the current work brings the discussion again to this context. In particular, all examples throughout the paper are truly first-order, consisting of universally quantified integrity constraints, unlike the propositional examples found in previous work.

We assume fixed a function-free first-order typed[1] signature for integrity constraints, consisting of a set $\mathcal{T}$ of types, a $\mathcal{T}$-indexed set of sets of constants $\mathcal{C} = \{C_T \mid T \in \mathcal{T}\}$, a $\mathcal{T}$-indexed set of sets of variables $\mathcal{V} = \{V_T \mid T \in \mathcal{T}\}$, a set of *base* predicate symbols $\mathcal{P}$, and a set of *built-in* predicates $\mathcal{R}$. The base predicates are the ones used in the database; the built-in predicates are typically comparison operators (e.g. equality), used only to specify constraints that the data must satisfy (Flesca et al. 2004). A constant $c \in C_T$ (or a variable $x \in V_T$) is said to have type T. Each predicate in $\mathcal{P} \cup \mathcal{R}$ has an arity, defined as a finite sequence of elements of $\mathcal{T} \cup \{\top\}$. In this context, the types are simply "names" for the sets of constants, and $\top$ stands for "any type".

[1] The choice of a typed signature is natural in the context of relational databases, where relations have clearly defined domains. The use of types eliminates spurious instantiations of variables that make no sense in practice, but whose presence introduces additional complexity in the theoretical analysis of integrity constraints. This very simple typing mechanism does not achieve the full power of a relational signature.

A constant or variable is a *term*. If $p \in \mathcal{P} \cup \mathcal{R}$ is a predicate of arity $T_1, \ldots, T_n$ and, for $1 \leq i \leq n$, t_i is a term such that (i) t_i has type $T_i \neq \top$ or (ii) $T_i = \top$, then $p(t_1, \ldots, t_n)$ is an *atom*. A literal is an atom or its negation, and it is a *base* literal if it is built from a base predicate symbol, and a *built-in* literal otherwise. In this work we assume equality $(=)$ to be a built-in predicate.

A *database* is a finite set of ground base atoms. This purely logical view of a database as a set of formulas, abstracting from its structure, suffices for the discussion of integrity constraints.

Every negative constraint (as defined by Beeri and Vardi (1981), see Introduction) can be written in clausal (or rule) form as

$$A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m \supset$$

with the proviso that every variable occurs at least once in $A_1, \ldots, A_n$, or, more generally, as $L_1, \ldots, L_p \supset$ with the proviso that: if L_j is a negated literal, then all variables in L_j occur at least once in $L_1, \ldots, L_{j-1}$, for $1 \leq j \leq p$. If the set $fv(r)$ of the free variables in rule r is empty, then r is said to be *ground*.

An instantiation of a rule r is a function $\theta : fv(r) \to \mathcal{C}$ such that x and $\theta(x)$ have the same type for every x. The *instance* $r\theta$ of r by θ is the ground rule obtained by replacing each $x \in fv(r)$ by $\theta(x)$.

Entailment from a database $\mathcal{I}$ is defined for ground base literals as (i) $\mathcal{I} \models A$ if $A \in \mathcal{I}$ and (ii) $\mathcal{I} \models \neg B$ if $B \notin \mathcal{I}$. Entailment of built-in literals is independent of $\mathcal{I}$ and has to be defined for each predicate symbol, e.g. $\mathcal{I} \models (a = b)$ iff a and b are the same (syntactic) constant. If r is a ground rule, then $\mathcal{I}$ *satisfies* r, $\mathcal{I} \models r$, if $\mathcal{I} \not\models L$ for some literal L occurring in r. In general, $\mathcal{I} \models r$ if $\mathcal{I} \models (r\theta)$ for every instance $r\theta$ of r. Finally, if η is a set of integrity constraints, then $\mathcal{I} \models \eta$ if $\mathcal{I} \models r$ for every $r \in \eta$.

Whenever $\mathcal{I} \not\models \eta$, the *problem of database repair* is to determine how to transform $\mathcal{I}$ into $\mathcal{I}'$ such that $\mathcal{I}' \models \eta$ by means of update actions. An *update action* is $+A$ or $-A$, where A is a base atom; if A is ground, this action indicates that A should be added (resp. removed) from $\mathcal{I}$. The correspondence between literals and update actions is made precise by means of two operators: if α is an action, then $lit(\alpha)$ is the literal corresponding to α, defined as $lit(+A) = A$ and $lit(-A) = \neg A$; if L is a base literal, then $ua(L)$ is the update action corresponding to L, defined as $ua(A) = +A$ and $ua(\neg A) = -A$. The *dual* of a literal or an update action is defined as $A^D = \neg A$, $(\neg A)^D = A$, $+A^D = -A$ and $-A^D = +A$. These operators extend to sets in the natural way.

There are typically many possible sets of ground actions that will make an inconsistent database consistent. Historically, there are a number of criteria that suggest that some of these sets may be better than others. The formalism of active integrity constraints, on the other hand, provides an explicit means to express preferences in each individual rule. The definitions we give here are slightly simplified from the original ones (Flesca et al. 2004) in view of the discussion before Theorem 3 in (Caroprese et al. 2009).

Definition 1 An *active integrity constraint* (AIC) is a rule r of the form

$$L_1, \ldots, L_m \supset \alpha_1 \mid \ldots \mid \alpha_k$$

such that $L_1, \ldots, L_m \supset$ is an integrity constraint and α_j are update actions whose

dual literals occur in $\{L_1, \ldots, L_m\}$, i.e. $\{\alpha_1, \ldots, \alpha_k\}^D \subseteq \{L_1, \ldots, L_m\}$. The *body* of r, $body\,(r)$, is $L_1, \ldots, L_m$, and the *head* of r, $head\,(r)$, is $\alpha_1 \mid \ldots \mid \alpha_k$.

The set $lit(head\,(r))^D$ contains the *updatable* literals of r. The *non-updatable* literals of r form the set $nup(r) = body\,(r) \setminus lit\,(head\,(r))^D$. Any built-in literals occuring in $body\,(r)$ are necessarily non-updatable.

When $\mathcal{I} \not\models r$, we say that r is *applicable* in $\mathcal{I}$. This reflects the "operational" view of AICs: if the body of a rule holds, then some action must be taken to guarantee that the corresponding integrity constraint becomes satisfied. AICs generalize (normal) integrity constraints, as these are equivalent to AICs with maximal head, where all allowed actions are present.

Given a set of ground update actions $\mathcal{U}$, the result of updating database $\mathcal{I}$ by $\mathcal{U}$, denoted $\mathcal{I} \circ \mathcal{U}$, is

$$(\mathcal{I} \cup \{A \mid +A \in \mathcal{U}\}) \setminus \{A \mid -A \in \mathcal{U}\}\ .$$

In order for this operation to be unambiguously defined, $\mathcal{U}$ is required to be *consistent*, i.e. not to contain any dual atoms (although it may contain $+p(\overline{t})$ and $-p(\overline{t'})$ for different sequences of terms $\overline{t}$ and $\overline{t'}$).

Definition 2 Let $\mathcal{I}$ be a database, η be a set of AICs, and $\mathcal{U}$ be a consistent set of ground update actions.

- An update action α is *founded* w.r.t. $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$ if there is $r \in \eta$ and an instantiation θ such that: (i) $\alpha \in head\,(r\theta)$, (ii) $\mathcal{I} \circ \mathcal{U} \models L$ for every $L \in body\,(r\theta) \setminus \left\{lit(\alpha)^D\right\}$.
- $\mathcal{U}$ is *founded* w.r.t. $\langle \mathcal{I}, \eta \rangle$ if all of its elements are founded w.r.t. $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$.
- $\mathcal{U}$ is a *founded weak repair* for $\langle \mathcal{I}, \eta \rangle$ if $\mathcal{I} \circ \mathcal{U} \models \eta$ and $\mathcal{U}$ is founded w.r.t. $\langle \mathcal{I}, \eta \rangle$. $\mathcal{U}$ is a *founded repair* if furthermore there is no set $\mathcal{U}' \subsetneq \mathcal{U}$ such that $\mathcal{I} \circ \mathcal{U}' \models \eta$.

These notions embody two of the main principles of database repair discussed in the Introduction: minimality of change(Chomicki 2007; Winslett 1990) and the common-sense law of inertia (Przymusinski and Turner 1997). A founded repair is guaranteed to make the smallest possible amount of changes to the database (minimimality of change); a founded set of update actions only contains changes that are motivated by the actual violations of the integrity constraints (common-sense law of inertia).

Just as AICs generalize traditional integrity constraints, the notion of founded (weak) repair coincides with the usual notion of (weak) repair for these.

In general, the problem of existence of a founded weak repair for $\langle \mathcal{I}, \eta \rangle$ is *NP*-complete, whereas the problem of existence of a founded repair is Σ_2^P-complete (Caroprese and Truszczyński 2011). If η contains only *normal* AICs (AICs whose head contains at most one action), then both problems are *NP*-complete.

Example 1 The running example for this article considers the following setting: a company's database keeps information about personnel, such as salaries, professional category, and insurances.

The following are examples of active integrity constraints specifying the company's

policies.

$$\mathsf{employee}(X), \neg\mathsf{hasInsurance}(X, \text{`}\mathtt{basic}'\text{)} \supset +\mathsf{hasInsurance}(X, \text{`}\mathtt{basic}'\text{)} \qquad (r_1)$$

$$\mathsf{employee}(X), \mathsf{salary}(X, \text{`}\mathtt{0}'\text{)}, \neg\mathsf{onLeave}(X) \supset +\mathsf{onLeave}(X) \qquad (r_2)$$

$$\mathsf{employee}(X), \mathsf{onLeave}(X), \neg\mathsf{salary}(X, \text{`}\mathtt{0}'\text{)} \supset +\mathsf{salary}(X, \text{`}\mathtt{0}'\text{)} \qquad (r_3)$$

$$\mathsf{salary}(X, Y), \mathsf{salary}(X, Z), Y \neq Z \supset -\mathsf{salary}(X, Y) \qquad (r_4)$$

Rule r_1 states that every employee must have a valid `'basic'` insurance; rules r_2 and r_3 state that the only employees who receive no salary are those on extended leave; and rule r_4 states that salary is a functional relation. The premise $\mathsf{employee}(X)$ in the first three rules is included so that the database is not restricted to current employees: for example, the company can keep a track of its former employees (who will typically be uninsured), or have interns that are not formally employees (and may be working *pro bono*).

The actions in the heads of the rules express preferences regarding how inconsistencies should be fixed. It is not reasonable, for example, to solve them by removing employees from the database. On the other hand, in rules r_2 and r_3, we choose to assume that it is more likely for information to be missing than for there to be too much information in the database.

Note that we do not need to include $-\mathsf{salary}(X, Z)$ in *head* (r_4), since it can be obtained by interchanging the values for Y and Z in any of its applicable instantiations.

Suppose that the database $\mathcal{I}$ contained the information

$$\{\mathsf{employee}(\text{`}\mathtt{john}'\text{)}, \mathsf{salary}(\text{`}\mathtt{john}'\text{, `}\mathtt{500}'\text{)}, \mathsf{onLeave}(\text{`}\mathtt{john}'\text{)}\}\,,$$

which contradicts rules r_1 and r_2. To become consistent w.r.t. rule r_1, we are required to add $\mathsf{hasInsurance}(\text{`}\mathtt{john}'\text{, `}\mathtt{basic}'\text{)}$. Likewise, to regain consistency w.r.t. rule r_2, we need to add $\mathsf{salary}(\text{`}\mathtt{john}'\text{, `}\mathtt{0}'\text{)}$. This change now breaks rule r_4, which must then be repaired by removing $\mathsf{salary}(\text{`}\mathtt{john}'\text{, `}\mathtt{500}'\text{)}$. One thus arrives at the following founded repair for $\mathcal{I}$.

$$\mathcal{U}_1 = \{+\mathsf{hasInsurance}(\text{`}\mathtt{john}'\text{, `}\mathtt{basic}'\text{)}, +\mathsf{salary}(\text{`}\mathtt{john}'\text{, `}\mathtt{0}'\text{)}, -\mathsf{salary}(\text{`}\mathtt{john}'\text{, `}\mathtt{500}'\text{)}\}$$

In order to check that this is indeed a founded set, note that that its actions are founded respectively by rules r_1, r_2 and r_4. Removing either of them leaves the database inconsistent, so this set is also minimal.

If we changed *head* (r_3) to $-\mathsf{onLeave}(X) \mid +\mathsf{salary}(X, \text{`}\mathtt{0}'\text{)}$, there would be two possible founded repairs, namely $\mathcal{U}_1$ above and

$$\mathcal{U}_2 = \{+\mathsf{hasInsurance}(\text{`}\mathtt{john}'\text{, `}\mathtt{basic}'\text{)}, -\mathsf{onLeave}(\text{`}\mathtt{john}'\text{)}\}$$

□

It has been observed by Caroprese and Truszczyński (2011) that founded repairs allow the support for the actions in a founded repair to be circular. The next example is an illustration of this situation.

Example 2 Consider the following AICs.

$$\mathsf{category}(X, \texttt{'junior'}), \mathsf{workTime}(X, Y), Y < 36, \mathsf{onLeave}(X) \supset -\mathsf{onLeave}(X) \qquad (r_5)$$

$$\neg\mathsf{category}(X, \texttt{'junior'}), \mathsf{workDept}(X, \texttt{'critical'}), \mathsf{onLeave}(X) \supset -\mathsf{onLeave}(X) \qquad (r_6)$$

$$\neg\mathsf{onLeave}(X), \mathsf{category}(X, \texttt{'junior'}), \mathsf{unsupervised}(X) \supset -\mathsf{category}(X, \texttt{'junior'}) \qquad (r_7)$$

Rule r_5 says that junior employees can only go on leave after being employed for at least 36 months. Rule r_6 says that non-junior staff on the critical department cannot go on leave (at all). Rule r_7 states that unsupervised employees may not be juniors, unless they are on leave.

Suppose that the database is as follows.

$$\mathcal{I} = \{\mathsf{category}(\texttt{'mike'}, \texttt{'junior'}), \mathsf{onLeave}(\texttt{'mike'}), \mathsf{workTime}(\texttt{'mike'}, \texttt{'18'}), \mathsf{workDept}(\texttt{'mike'}, \texttt{'critical'}), \mathsf{unsupervised}(\texttt{'mike'})\}$$

Rule r_5 is not satisfied; $\mathcal{I}$ can be made consistent by applying the repair

$$\mathcal{U} = \{\underbrace{-\mathsf{onLeave}(\texttt{'mike'})}_{\alpha}, \underbrace{-\mathsf{category}(\texttt{'mike'}, \texttt{'junior'})}_{\beta}\},$$

which is founded: letting $\theta = \{X/\texttt{'mike'}\}$, if $\mathcal{U}' = \mathcal{U} \setminus \{\alpha\}$, then $\mathcal{I} \circ \mathcal{U}' \not\models r_6\theta$, and $\alpha \in \mathit{head}(r_6\theta)$; similarly, if $\mathcal{U}'' = \mathcal{U} \setminus \{\beta\}$, then $\mathcal{I} \circ \mathcal{U}'' \not\models r_7\theta$, and $\beta \in \mathit{head}(r_7\theta)$. □

Caroprese and Truszczyński (2011) considered such circular dependencies to be problematic, and introduced *justified repairs* in order to avoid them.

Definition 3 Let $\mathcal{I}$ be a database and η a set of AICs. A set $\mathcal{U}$ of update actions is *closed* under a ground rule r if $\mathit{nup}(r) \subseteq \mathit{lit}(\mathcal{U})$ implies $\mathit{head}(r) \cap \mathcal{U} \neq \emptyset$, and $\mathcal{U}$ is closed under η if it is closed under every instance of every rule in η.

An update action $+A$ (respectively, $-A$) is a *no-effect* action w.r.t. $(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ if $A \in \mathcal{I} \cap (\mathcal{I} \circ \mathcal{U})$ (respectively $A \notin \mathcal{I} \cup (\mathcal{I} \circ \mathcal{U})$) – the action does not change either $\mathcal{I}$ or $\mathcal{I} \circ \mathcal{U}$. The set of all no-effect actions with respect to $(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is denoted by $\mathit{ne}(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$.

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 $\mathit{ne}(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ and closed under η.

A weak repair $\mathcal{U}$ for $\langle \mathcal{I}, \eta \rangle$ is a *justified weak repair* if (i) $\mathcal{U} \cap \mathit{ne}(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) = \emptyset$ and (ii) $\mathcal{U} \cup \mathit{ne}(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is a justified action set.

All justified weak repairs are founded, but Caroprese and Truszczyński (2011) claim that they successfully avoid circularity of support. Section 5 gives an operational characterization of founded and justified repairs.

In spite of the minimality required of justified weak repairs, these may contain a justified repair as a proper subset, since the minimality involved in this definition is within a different universe.

Example 3 The repair $\mathcal{U}_1$ from Example 1 is a justified weak repair: the set of no-effect actions w.r.t. $\mathcal{I}$ and $\mathcal{U}_1$ contains $\mathsf{employee}(\texttt{'john'})$ and $\mathsf{salary}(\texttt{'john'}, \texttt{'500'})$ among others. The only instances of rules that satisfy the condition $\mathit{nup}(r\theta) \subseteq \mathit{lit}(\mathcal{U}_1 \cup \mathit{ne}(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1))$ are $r_1\theta$ and $r_3\theta$, and $\mathcal{U}_1$ contains precisely the actions in the heads of those rules,

so $\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ is a justified action set. For minimality, note that $nup(r\theta) \subseteq lit(ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1))$ for the same two rules, so no proper subset of $\mathcal{U}_1$ can yield a justified action set when joined with $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$.

Consider now the repair $\mathcal{U}$ from Example 2. Here, the set $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ contains the facts $\mathsf{workTime}(\text{`}\mathtt{mike}'\text{, `}\mathtt{18}'\text{)}$, $\mathsf{workDept}(\text{`}\mathtt{mike}'\text{, }\mathsf{critical}\text{)}$ and $\mathsf{unsupervised}(\text{`}\mathtt{mike}'\text{)}$. Since $nup(r\theta) \subseteq lit(ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$ never holds, the set $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is a justified action set properly contained in $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$, and $\mathcal{U}$ is not a justified repair. □

The following example shows a different case of circularity of support.

Example 4 The company also has a security department, whose non-administrative staff is required to have both a security certification and a special risk insurance. These two qualifications are simultaneous: if an employee has the certification, then the company automatically subscribes the special insurance, and it will never book that insurance for an uncertified employee.

These employees are also eligible for a yearly prize specific to employees with high-risk duties. The rules below encode these restrictions.

$$\mathsf{workDept}(X, \text{`}\mathtt{security}'\text{),}\neg\mathsf{isCertified}(X, \text{`}\mathtt{security}'\text{), }\mathsf{hasInsurance}(X, \text{`}\mathtt{risk}'\text{)} \supset +\mathsf{isCertified}(X, \text{`}\mathtt{security}'\text{)} \qquad (r_8)$$

$$\mathsf{workDept}(X, \text{`}\mathtt{security}'\text{),}\mathsf{isCertified}(X, \text{`}\mathtt{security}'\text{), }\neg\mathsf{hasInsurance}(X, \text{`}\mathtt{risk}'\text{)} \supset +\mathsf{hasInsurance}(X, \text{`}\mathtt{risk}'\text{)} \qquad (r_9)$$

$$\mathsf{workDept}(X, \text{`}\mathtt{security}'\text{),}\mathsf{currYearPrize}(X, \text{`}\mathtt{risk}'\text{), }\neg\mathsf{hasInsurance}(X, \text{`}\mathtt{risk}'\text{)} \supset -\mathsf{currYearPrize}(X, \text{`}\mathtt{risk}'\text{)} \qquad (r_{10})$$

Suppose that the database contains the following information about an employee Jack.

$$DB = \{\mathsf{employee}(\text{`}\mathtt{jack}'\text{), }\mathsf{workDept}(\text{`}\mathtt{jack}'\text{, `}\mathtt{security}'\text{), }\mathsf{currYearPrize}(\text{`}\mathtt{jack}'\text{, `}\mathtt{risk}'\text{)}\}$$

There is a simple way to repair the database, namely by applying

$$\mathcal{U}_1 = \{-\mathsf{currYearPrize}(\text{`}\mathtt{jack}'\text{, `}\mathtt{risk}'\text{)}\}\,,$$

which is founded because of rule r_{10}. There is however another repair, namely

$$\mathcal{U}_2 = \{+\mathsf{isCertified}(\text{`}\mathtt{jack}'\text{, `}\mathtt{security}'\text{), }+\mathsf{hasInsurance}(\text{`}\mathtt{jack}'\text{, `}\mathtt{risk}'\text{)}\}\,,$$

and this is again a founded repair that is not justified, as in Example 2. Unlike that example, however, there is no clear way to derive this repair by analyzing which rules are not satisfied. □

This example illustrates that a repair can be founded and still violate the common sense law of inertia, by including subsets of actions that are not motivated by either the database or other repair actions.

In Section 5.3 we will introduce a new notion of repair that distinguishes between the two kinds of circularity in Examples 2 and 4.

Existence of justified weak repairs or justified repairs for $\langle \mathcal{I}, \eta \rangle$ is again a Σ_2^P-complete problem (Caroprese and Truszczyński 2011) in the general case, and NP-complete if η contains only normal AICs.

The major problem with this framework lies in the high complexity of deciding whether

an inconsistent database can be repaired, and in computing an adequate repair; therefore, techniques to lower the size of the problem are extremely useful in practice, and Sections 3 and 4 discuss how a set of AICs η can be divided into smaller sets such that repairs can be computed separately for each of those sets and combined in polynomial time.

Expressing tuple-generating dependencies. The notion of AIC presented above can express both equality-generating constraints and universal constraints. However, general tuple-generating dependencies (TGDs) are not directly translatable in this framework, since the possible update actions would require introduction of new variables.

It is possible to extend the language of AICs with the capability to express TGDs by allowing *quantified literals* of the form $\forall Y.(\neg A(X,Y))$ in the body. Similarly to negated atoms, it is fundamental to require that all free variables in a quantified literal appear earlier in the clause, so that they are instantiated. This is sufficient to guarantee that the evaluation of a quantified literal is no more complex than that of a regular literal, since it still corresponds to a simple query to the database.

The simplest way to treat quantified literals is to see them as control conditions, in the same way as built-in predicates were handled: there are no update actions corresponding to quantified literals. However, it is also possible to allow *default repairs*: any update action of the form $+A(X,t)$ actually serves as a repair for the generalized literal $\forall Y.(\neg A(X,Y))$.

This type of default action is quite common: in our setting, it could be used e.g. to express that any junior employee without a supervisor is by default supervised by Alice.

$$\mathsf{category}(X, \texttt{`junior'}), \forall W.(\neg \mathsf{hasSupervisor}(X,W)) \supset +\mathsf{hasSupervisor}(X, \texttt{`alice'})$$

It is straightforward to generalize the notion of dual action to a binary relation such that the results in this paper still hold.

A more general option would be to allow actions with unnamed individuals, in the style of (Caroprese et al. 2012). However, this quickly leads to problems with decidability, and requires implementation of chase algorithms to keep repair trees finite. The restricted version proposed is applicable in sufficiently many contexts to make this framework interesting in practice; however, for simplicity quantified literals will not be used in the remainder of the paper.

3 Independence and parallelization

In practical settings, the number of integrity constraints imposed on a given database can be quite large, but only a few of them will be relevant to repair each particular inconsistent database. This section discusses how a set η of (general) ICs in clausal form can be partitioned in distinct sets $\eta_1, \ldots, \eta_n$ such that the search for repairs for a database $\mathcal{I}$ and η can be parallelized among the η_i. For AICs, this notion of independence can be made stronger, so that more (smaller) independent sets will appear.

3.1 Independence of integrity constraints

Two ground integrity constraints are independent if they do not share base atoms among their literals, so that satisfaction of one does not affect satisfaction of the other.

Recall that base atoms are those built from the database's predicates (the set $\mathcal{P}$), and they are the only ones that can be changed by repair actions. Case in point, the following two AICs should be independent because $\pm(t_1 = t_2)$ is not a valid update action, since equality is a built-in predicate.

$$\mathsf{salary}(X, Y), \mathsf{salary}(X, Z), Y \neq Z \supset$$
$$\mathsf{workTime}(X, Y), \mathsf{workTime}(X, Z), Y \neq Z \supset$$

Definition 4

1. The atom *underlying* a literal L is $|L|$, defined as $|a| = |\neg a| = a$.
2. Let r_1 and r_2 be two ground ICs, where r_1 is $L_1, \ldots, L_n \supset$ and r_2 is $M_1, \ldots, M_m \supset$. Then r_1 and r_2 are *independent*, $r_1 \perp r_2$, if $\{|L_i| \mid 1 \leq i \leq n\} \cap \{|M_j| \mid 1 \leq j \leq m\}$ does not contain any base literals.
3. Two (not necessarily ground) ICs r_1 and r_2 are independent, $r_1 \perp r_2$, if $r_1\theta_1 \perp r_2\theta_2$ for every instantiation $r_1\theta_1$ and $r_2\theta_2$ of r_1 and r_2.
4. Two sets of ICs η_1 and η_2 are *independent*, $\eta_1 \perp \eta_2$, if $r \perp s$ whenever $r \in \eta_1$ and $s \in \eta_2$.

In particular, if the bodies of r_1 and r_2 share no base predicate symbols, then $r_1 \perp r_2$. However, this condition is not necessary: the following two rules are independent, although they both use the base predicate symbol workDept.

$$\mathsf{intern}(X), \mathsf{workDept}(X, \texttt{`security'}) \supset$$
$$\mathsf{employee}(X), \mathsf{workDept}(X, \texttt{`admin'}), \neg\mathsf{category}(X, \texttt{`director'}) \supset$$

Independence only depends on the rules. Thus, dividing a set of ICs into independent subsets may require some computation time, but it does not have to be repeated when the *database* changes. This property is expressed by Theorem 1 below.

Definition 5 A *partition* of a set of integrity constraints η is a set $\vec{\eta} = \{\eta_1, \ldots, \eta_n\}$ such that $\eta = \cup_{i=1}^{n} \eta_i$ and $\eta_i \perp \eta_j$ for $i \neq j$.

Theorem 1 Let $\vec{\eta}$ be a partition of η.

1. If $\mathcal{U}$ is a (weak) repair for $\langle \mathcal{I}, \eta \rangle$, then there exist sets $\mathcal{U}_1, \ldots, \mathcal{U}_n$ with $\mathcal{U} = \cup_{i=1}^{n} \mathcal{U}_i$ such that $\mathcal{U}_i$ is a (weak) repair for $\langle \mathcal{I}, \eta_i \rangle$.
2. If $\mathcal{U}_i$ is a (weak) repair for $\langle \mathcal{I}, \eta_i \rangle$ for $i = 1, \ldots, n$ and $\mathcal{U} = \cup_{i=1}^{n} \mathcal{U}_i$, then $\mathcal{U}$ is a (weak) repair for $\langle \mathcal{I}, \eta \rangle$.

The proof of this result is given in Appendix A.1. Its practical significance is a parallelization algorithm: if $\eta = \eta_1 \cup \eta_2$ with $\eta_1 \perp \eta_2$, then (weak) repairs for $\langle \mathcal{I}, \eta \rangle$ can be expressed as unions of (weak) repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I}, \eta_2 \rangle$, so one can search for these instead and combine them at the end; the theorem guarantees that no spurious results are obtained. None of these results speak about *how* (weak) repairs are computed. Therefore, they can be applied together with *any* existing techniques to find repairs of inconsistent databases (see e.g. (Mayol and Teniente 1999b)).

3.2 Computing independent sets of ICs

In order to benefit as much as possible from Theorem 1 in practice, it is useful to be able to divide η into as many independent sets as possible. To achieve this, we focus on the *negation* of the independence relation.

Definition 6 Two AICs r_1 and r_2 are *dependent*, $r_1 \leftrightharpoons r_2$, if there exist instances $r_1\theta_1$ and $r_2\theta_2$ of r_1 and r_2 and literals $L_1 \in \mathit{body}(r_1\theta_1)$ and $L_2 \in \mathit{body}(r_2\theta_2)$ such that $|L_1| = |L_2|$.

Lemma 1 Let $\vec{\eta}$ be a partition of η. Then η_i is closed under $\leftrightharpoons$ for every i, i.e. for every rule $r, r' \in \eta$, if $r \in \eta_i$ and $r \leftrightharpoons r'$, then $r' \in \eta_i$.

Proof
Let $r \in \eta_i$ and $r' \in \eta$ be such that $r \leftrightharpoons r'$. Since $\vec{\eta}$ is a partition of η, $r' \in \eta_k$ for some k. But $i \neq k$ would contradict $\eta_i \perp \eta_k$, hence $i = k$. Therefore η_i is closed under $\leftrightharpoons$. □

This relation is reflexive and symmetric, so its transitive closure is an equivalence relation, which we represented by $\not\perp$ in view of the next result.

Theorem 2 The quotient set $\eta/\not\perp$ is a partition of η. Furthermore, for any other partition $\vec{\eta}'$ of η, if $\eta'_i \in \vec{\eta}'$, there exists $\eta_j \in \eta/\not\perp$ such that $\eta_j \subseteq \eta'_i$.

Proof
Let $\eta/\not\perp = \{\eta_1, \ldots, \eta_n\}$. Then $\bigcup_{i=1}^{n} \eta_i = \eta$ and $\eta_i \perp \eta_j$ by definition of quotient set. Given η'_i as in the statement of the theorem and choosing $r \in \eta'_i$, it follows that $[r] \subseteq \eta'_i$ by Lemma 1 and the fact that $[r]$ is the minimal set containing r and closed under $\leftrightharpoons$. □

Unlike in the propositional case (Cruz-Filipe 2014), the direct algorithm for computing $\eta/\not\perp$ is computationally heavy because it needs to consider all possible instances of all rules.

Theorem 3 Let η be a set of closed ICs such that every rule in η contains at most k literals in its body. Then $\eta/\not\perp$ can be computed in $\mathcal{O}(k \times |\eta|)$.

Proof
Consider the undirected graph whose nodes are both the rules in η and the atoms occurring in those rules, and where there is an edge between an atom and a rule if that atom occurs in that rule. This graph has at most $k \times |\eta|$ nodes and can be constructed in $\mathcal{O}(k \times |\eta|)$ time; it is a well-known fact that its connected components can again be computed in $\mathcal{O}(k \times |\eta|)$ time, and the rules in each component coincide precisely with the equivalence classes in $\eta/\not\perp$. □

The problem is that, for a general set of ICs η, this complexity bound is polynomial on the number of its grounded instances – which are typically exponential in η. On the otherhand, this complexity bound is independent of the underlying database, which is useful since the database typically changes more often than η. Furthermore, when new rules are added to η one can reuse the existing partition for η as a starting point, rather

than computing $\eta/_{\not\perp}$. We also observe that k typically does not grow with η and is usually small, so essentially this complexity bound depends only on the size of η.

The criterium presented after Definition 4 – $r_1 \perp r_2$ if r_1 and r_2 share no base predicate symbols – does not necessarily yield an optimal partition of η. However, it is much more efficient to implement, since it does not require grounding η. Our proof-of-concept prototype tool uses only this method to split a set of ICs into independent sets, already obtaining substantial improvements in running times (see (Cruz-Filipe et al. 2016) and Section 6).

Any algorithm for computing (weak) repairs for a database $\mathcal{I}$ w.r.t. a set of integrity constraints η can then be parallelized in the following way.

1. Find a partition $\vec{\eta}$ of η.
2. For each $\eta_i \in \vec{\eta}$, find the weak repairs of $\langle \mathcal{I}, \eta_i \rangle$.
3. Return all $\mathcal{U} = \cup_i \mathcal{U}_i$ where each $\mathcal{U}_i$ is a weak repair for $\langle \mathcal{I}, \eta_i \rangle$.

3.3 Independence of active integrity constraints

All the results above can also be applied to active integrity constraints. However, the presence of update actions in the head of a rule means that only those actions should be considered when updating the database. This leads to a stronger notion of independence.

Definition 7

1. Let r_1 and r_2 be ground AICs. Then r_1 *affects* r_2, $r_1 \preceq r_2$, if there are $\alpha \in \mathit{head}(r_1)$ and $L \in \mathit{body}(r_2)$ such that $|\mathit{lit}(\alpha)| = |L|$.
2. Let r_1 and r_2 be (not necessarily ground) AICs. Then $r_1 \preceq r_2$ if $r_1\theta_1 \preceq r_2\theta_2$ for some instances $r_1\theta_1$ and $r_2\theta_2$ of r_1 and r_2.

Intuitively, r_1 affects r_2 if applying r_1 may affect applicability of r_2. In particular, $r_1 \preceq r_2$ implies $r_1 \leftrightharpoons r_2$, but not conversely: in fact, $\preceq$ induces a stronger independence relation.

Definition 8 Two sets of AICs η_1 and η_2 are *strongly independent*, $\eta_1 \perp\!\!\!\perp \eta_2$, if $r_1 \not\preceq r_2$ and $r_2 \not\preceq r_1$ for every $r_1 \in \eta_1$ and $r_2 \in \eta_2$.

In particular, if the actions in the heads of rules of η_i cannot add or remove atoms that underlie literals in the body of rules of η_{3-i}, then $\eta_1 \perp\!\!\!\perp \eta_2$. Thus, $\eta_1 \perp \eta_2$ implies $\eta_1 \perp\!\!\!\perp \eta_2$, but not conversely.

Example 5 Combining the rules in Examples 1 and 2, the sets $\{r_1\}$ and $\{r_2, r_3, r_4, r_5, r_6, r_7\}$ are independent, although rules r_1, r_2 and r_3 all include $\mathsf{employee}(X)$ in their body (and so, for any θ, e.g. $r_1\theta$ and $r_2\theta$ both include $\mathsf{employee}(\theta(X))$ in their body). □

The algorithm in the proof of Theorem 2 can be readily adapted for AICs.

In general, it is not the case that repairs for $\langle \mathcal{I}, \eta \rangle$ can be divided in repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and for $\langle \mathcal{I}, \eta_2 \rangle$ when $\eta = \eta_1 \cup \eta_2$ with $\eta_1 \perp\!\!\!\perp \eta_2$; however, in the framework of AICs the focus is typically not on arbitrary repairs. Indeed, item 1 on Theorem 1 applies directly to sets $\mathcal{U}_1$ and $\mathcal{U}_2$ such that every update action in $\mathcal{U}_i$ occurs in the head of an instance of a rule in η_i, since this implies the hypothesis. Conversely, if $\mathcal{U}$ is a (weak) repair for $\langle \mathcal{I}, \eta \rangle$

containing only actions in the head of instances of rules in η, then the proof of item 2 also goes through. This variant of Theorem 1 is the one that it is natural to consider, since the specific semantics for AICs only consider actions in heads of rules.

Furthermore, strong independence also fits in nicely with the concepts of founded and justified (weak) repairs. Given a set of update actions $\mathcal{U}$ and a set of AICs η, define the *restriction* of $\mathcal{U}$ to η to be

$$\mathcal{U}_\eta = \{\alpha \in \mathcal{U} \mid \alpha \in \mathit{head}\,(r\theta) \text{ for some instance } r\theta \text{ of } r \in \eta\}\,.$$

Theorem 4 Let $\mathcal{I}$ be a database, η_1 and η_2 be sets of AICs with $\eta_1 \perp\!\!\!\perp \eta_2$, and $\eta = \eta_1 \cup \eta_2$. Then:

1. If $\mathcal{U}_1$ and $\mathcal{U}_2$ are founded (weak) repairs for $\langle \mathcal{I}, \eta_1\rangle$ and $\langle \mathcal{I}, \eta_2\rangle$, respectively, then $\mathcal{U}_1 \cup \mathcal{U}_2$ is a founded (weak) repair for $\langle \mathcal{I}, \eta\rangle$.
2. If $\mathcal{U}_1$ and $\mathcal{U}_2$ are justified (weak) repairs for $\langle \mathcal{I}, \eta_1\rangle$ and $\langle \mathcal{I}, \eta_2\rangle$, respectively, then $\mathcal{U}_1 \cup \mathcal{U}_2$ is a justified (weak) repair for $\langle \mathcal{I}, \eta\rangle$.
3. If $\mathcal{U}$ is a founded (weak) repair for $\langle \mathcal{I}, \eta\rangle$, then each $\mathcal{U}_{\eta_i}$ is a founded (weak) repair for $\langle \mathcal{I}, \eta_i\rangle$.
4. If $\mathcal{U}$ is a justified (weak) repair for $\langle \mathcal{I}, \eta\rangle$, then each $\mathcal{U}_{\eta_i}$ is a justified (weak) repair for $\langle \mathcal{I}, \eta_i\rangle$.

The proof of this result is given in Appendix A.2. The practical significance of this result is that the parallel computation of repairs of databases w.r.t. a given set of integrity constraints extends to the case of founded or justified repairs w.r.t. a set of AICs. Furthermore, the parallelization algorithm at the end of the previous section can be applied to strong independence. Note that Theorem 4 can be generalized to partitions (w.r.t. strong independence), similar to Theorem 1.

4 Stratification of active integrity constraints

Strong independence can also be used to split a set of AICs in several sets such that repairs can be computed incrementally.

Recall that an AIC r_1 *affects* another AIC r_2, $r_1 \preceq r_2$, if there is an action in the head of an instance of r_1 whose underlying atom occurs (possibly negated) in the body of an instance of r_2. For a given η, let $\preceq_\eta$ be the transitive closure of $\preceq$ in η, and $\approx_\eta$ be the equivalence relation induced by $\preceq$, i.e. $r_1 \approx_\eta r_2$ iff $r_1 \preceq_\eta r_2$ and $r_2 \preceq_\eta r_1$. Throughout the remainder of this section, the set η is fixed and we omit the subscripts in $\preceq$ and $\approx$.

Definition 9 Let $\eta_1, \eta_2 \subseteq \eta$ be closed under $\approx$. Then η_1 *precedes* η_2, written $\eta_1 \prec \eta_2$, if (i) $r_1 \preceq r_2$ for some $r_1 \in \eta_1$ and $r_2 \in \eta_2$, but (ii) $r_2 \not\preceq r_1$ for every $r_1 \in \eta_1$ and $r_2 \in \eta_2$.

In particular, if $\eta_1 \prec \eta_2$, then η_1 and η_2 must be disjoint.

Example 6 In the setting of Example 4, the set $\{r_8, r_9, r_{10}\}$ can be divided into two subsets $\eta_1 = \{r_8, r_9\}$ and $\eta_2 = \{r_{10}\}$ with $\eta_1 \prec \eta_2$. □

It is well-known that $\langle {}^{\eta}\!/_{\approx}, \preceq\rangle$ is a partial order, where $[r_1]_\approx \preceq [r_2]_\approx$ iff $r_1 \preceq r_2$. Sets that are closed under $\approx$ can be written as a union of elements of ${}^{\eta}\!/_{\approx}$; in the particular case that $\eta_1, \eta_2 \in {}^{\eta}\!/_{\approx}$, then $\eta_1 \prec \eta_2$ iff $\eta_1 \preceq \eta_2$ and $\eta_1 \neq \eta_2$.

The construction in Theorem 2 can again be readily adapted to compute $\eta/_{\approx}$, using directed graphs and strongly connected components.

The term *precedes* stems from the fact that $\eta_1 \prec \eta_2$ implies that different types of repairs for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$ can be computed by first considering only the AICs in η_1, and afterwards considering the repairs in η_2. This is formally stated in the following two theorems.

Theorem 5 Let $\eta_1, \eta_2 \subseteq \eta$ with $\eta_1 \prec \eta_2$, $\mathcal{I}$ be a database and $\mathcal{U}$ be a set of update actions such that all actions in $\mathcal{U}$ occur in the head of some instance of a rule in $\eta_1 \cup \eta_2$, and take $\mathcal{U}_1 = \mathcal{U}_{\eta_1}$ and $\mathcal{U}_2 = \mathcal{U}_{\eta_2}$.

1. If $\mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$, then $\mathcal{U}_1$ and $\mathcal{U}_2$ are weak repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, respectively.
2. If $\mathcal{U}$ is a founded weak repair for $\langle \mathcal{I}, \eta \rangle$, then $\mathcal{U}_1$ and $\mathcal{U}_2$ are founded weak repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, respectively.
3. If $\mathcal{U}$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$, then $\mathcal{U}_1$ and $\mathcal{U}_2$ are justified weak repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, respectively.

The proof of Theorem 5 is given in Appendix A.3. The analogous result for repairs does not hold in this setting: it may happen that $\mathcal{U}$ is a repair, but $\mathcal{U}_1$ is a weak repair, since there may be a repair for $\langle \mathcal{I}, \eta_1 \rangle$ such that there is no (weak) repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$.

Example 7 Consider the setting where we have rules r_6 and r_7 from Example 2, together with the two new rules r_{11} and r_{12}, specifying supervisor policy within the critical department.

$$\neg \mathsf{category}(X, \texttt{`junior'}), \mathsf{workDept}(X, \texttt{`critical'}), \mathsf{onLeave}(X) \supset -\mathsf{onLeave}(X) \qquad (r_6)$$

$$\neg \mathsf{onLeave}(X), \mathsf{category}(X, \texttt{`junior'}), \mathsf{unsupervised}(X) \supset -\mathsf{category}(X, \texttt{`junior'}) \qquad (r_7)$$

$$\mathsf{workDept}(X, \texttt{`critical'}), \mathsf{category}(X, \texttt{`junior'}), \neg \mathsf{hasSupervisor}(X, \texttt{`ellen'}) \supset +\mathsf{hasSupervisor}(X, \texttt{`ellen'}) \qquad (r_{11})$$

$$\mathsf{onLeave}(X), \mathsf{hasSupervisor}(X, W) \supset -\mathsf{hasSupervisor}(X, W) \qquad (r_{12})$$

Taking $\eta_1 = \{r_6, r_7\}$ and $\eta_2 = \{r_{11}, r_{12}\}$, one has $\eta_1 \prec \eta_2$. Take

$$\mathcal{I} = \{\mathsf{unsupervised}(\texttt{`ann'}), \mathsf{category}(\texttt{`ann'}, \texttt{`junior'}), \mathsf{onLeave}(\texttt{`ann'}), \mathsf{workDept}(\texttt{`ann'}, \texttt{`critical'})\}$$

Then $\mathcal{I}$ is consistent w.r.t. η_1, but the only founded repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$ is

$$\mathcal{U} = \{-\mathsf{onLeave}(\texttt{`ann'}), -\mathsf{category}(\texttt{`ann'}, \texttt{`junior'})\}$$

which is not the union of $\emptyset$ with a repair for $\langle \mathcal{I}, \eta_2 \rangle$. □

In the converse direction, the relationship between $\mathcal{U}$, $\mathcal{U}_1$ and $\mathcal{U}_2$ also holds for repairs.

Theorem 6 Let $\eta_1, \eta_2 \subseteq \eta$ with $\eta_1 \prec \eta_2$, $\mathcal{I}$ be a database, and $\mathcal{U}_1$ and $\mathcal{U}_2$ be sets of update actions such that all actions in $\mathcal{U}_i$ occur in the head of some instance of a rule in η_i. Let $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

1. If $\mathcal{U}_1$ is a weak repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is a weak repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.

2. If $\mathcal{U}_1$ is a repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is a repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is a repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.
3. If $\mathcal{U}_1$ is a founded (weak) repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is a founded (weak) repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is a founded (weak) repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.
4. If $\mathcal{U}_1$ is a justified (weak) repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is a justified (weak) repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is a justified (weak) repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.

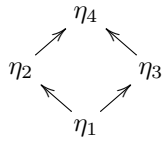
The proof of Theorem 6 can be found in Appendix A.4.

Using Theorem 5, the search for (weak) repairs can be divided into smaller steps, whose results can then be combined invoking Theorem 6. However, $\langle {}^{\eta}/_{\approx}, \preceq \rangle$ is in general not a total order. Therefore, to obtain (weak) repairs for η, it is also necessary to be able to combine weak repairs of sets that are not related via $\prec$; for example, if $\eta_1 \prec \eta_2$ and $\eta_1 \prec \eta_3$, then the first step would be to search for a weak repair for $\langle \mathcal{I}, \eta_1 \rangle$, and then extend this to weak repairs for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$ and $\langle \mathcal{I}, \eta_1 \cup \eta_3 \rangle$; but now these weak repairs must be combined in a single one.

Consider a weak repair $\mathcal{U}$ for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \cup \eta_3 \rangle$. By Theorem 5, $\mathcal{U}_{\eta_1}$ is a weak repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_{\eta_2 \cup \eta_3}$ is a weak repair for $\langle \mathcal{I} \circ \eta_1, \eta_2 \cup \eta_3 \rangle$. The key is that, in this situation, $\eta_2 \perp\!\!\!\perp \eta_3$, so Theorem 1 and 4 also apply.

We illustrate the general technique by an example, which also explains why we need to consider, in general, sets that are closed under $\approx$, rather than only elements of ${}^{\eta}/_{\approx}$.

Example 8 Consider sets $\eta_1, \ldots, \eta_4$ of AICs with precedence relation as summarized in the diagram on the right. In order to find e.g. a founded weak repair for $\langle \mathcal{I}, \eta \rangle$, where $\eta = \bigcup_{i=1}^{4} \eta_i$, we can apply the following sequence of steps:



1. find all founded weak repairs for $\langle \mathcal{I}, \eta_1 \rangle$;
2. extend each such $\mathcal{U}$ to founded weak repairs for $\langle \mathcal{I} \circ \mathcal{U}, \eta_2 \rangle$ and $\langle \mathcal{I} \circ \mathcal{U}, \eta_3 \rangle$;
3. for each pair of weak repairs $\mathcal{U}_2$ for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$ and $\mathcal{U}_3$ for $\langle \mathcal{I}, \eta_1 \cup \eta_3 \rangle$ such that $\mathcal{U}_2$ and $\mathcal{U}_3$ coincide on the actions from heads of rules in η_1, find weak repairs for $\langle \mathcal{I} \circ (\mathcal{U}_2 \cup \mathcal{U}_3), \eta_4 \rangle$.

The last step relies on the fact that any weak repair $\mathcal{U}$ for $\langle \mathcal{I}, \eta \rangle$ must contain a weak repair $\mathcal{U}'$ for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \cup \eta_3 \rangle$, which can be split into a weak repair $\mathcal{U}_1$ for $\langle \mathcal{I}, \eta_1 \rangle$ and weak repairs $\mathcal{U}'_2$ for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$ and $\mathcal{U}'_3$ for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_3 \rangle$. Defining $\mathcal{U}_2 = \mathcal{U}'_2 \cup \mathcal{U}_1$ and $\mathcal{U}_3 = \mathcal{U}'_3 \cup \mathcal{U}_1$, it must be the case that $\mathcal{U}' = \mathcal{U}_1 \cup \mathcal{U}'_2 \cup \mathcal{U}'_3 = \mathcal{U}_2 \cup \mathcal{U}_3$.

The results in this section guarantee that this algorithm finds all founded weak repairs for $\langle \mathcal{I}, \eta \rangle$. □

The following is a general strategy for computing founded or justified weak repairs for a database inconsistent w.r.t. a set η of AICs, assuming that η cannot be split into strongly independent subsets.

1. Compute ${}^{\eta}/_{\approx}$ (or an overapproximation of this partial order).
2. Find repairs for the minimal elements of ${}^{\eta}/_{\approx}$.
3. For each non-minimal element η_j, find its repairs by (i) combining the repairs for its predecessors, (ii) applying each result to $\mathcal{I}$, with result $\mathcal{I}'$, and (iii) computing repairs for $\langle \mathcal{I}', \eta_j \rangle$ (as in Example 8).

The only catch regards the situation depicted in Example 7: in step 3, whenever a repair cannot be extended when moving upwards in $^{\eta_i}/_{\approx}$, one must also consider weak repairs extending that repair, since the end result may be a repair for the larger set.

Combining these results with those in the previous section, *any* algorithm to find founded or justified (weak) repairs for an inconsistent database may be sped up by first splitting the set of integrity constraints into strongly independent sets, and then stratifying these components. See (Cruz-Filipe et al. 2016) for an implementation of these algorithms and a discussion of their practical impact.

5 Finding repairs of inconsistent databases

We now look into a different question: how to compute repairs for a database $\mathcal{I}$ not satisfying a set of integrity constraints η. The idea is simple: attempt to fix the database by looking at each integrity constraint that is not being satisfied at a time. Of course, this may lead to other integrity constraints not being satisfied, so the procedure must be repeated until a repaired database is obtained. The first construction we presente is a generic (not very efficient) algorithm applicable to any set of clausal integrity constraints, while the later ones refine on this to obtain specific kinds of repairs for sets of active integrity constraints.

Throughout this section, $\mathcal{I}$ will always be a database and η a finite set of (active) integrity constraints. In practice, finiteness of η always holds, and it is relevant for some theoretical properties discussed below.

5.1 Computing repairs from integrity constraints

As remarked before Theorem 3 in (Caroprese et al. 2009), every repair for a given set of integrity constraints is contained in the set $\bigcup_{r\in\eta}\{ua(L)^D \mid L \in body(r\theta) \text{ for some } \theta\}$. This suggests the following way to compute them by constructing a tree.

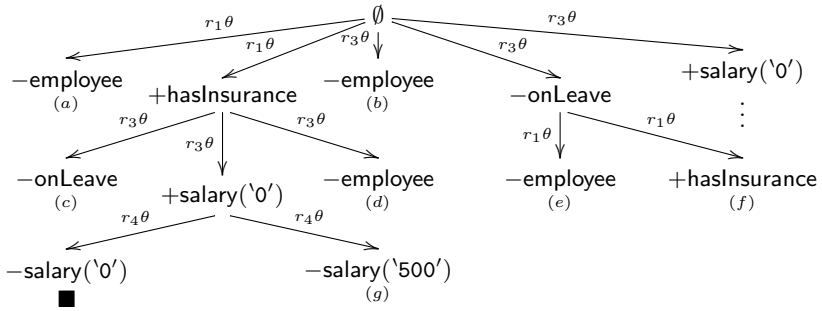
Definition 10 The *repair tree* for $\langle \mathcal{I}, \eta\rangle$, $T_{\langle \mathcal{I},\eta\rangle}$, is a tree where: nodes are sets of update actions; each edge is labeled with an instance of a rule; the root is $\emptyset$; and for each consistent node n and instance $r\theta$ of rule $r \in \eta$, if $\mathcal{I} \circ n \not\models r\theta$ then for each $L \in body(r\theta)$ the set $n' = n \cup \{ua(L)^D\}$ is a child of n, with the edge from n to n' labeled $r\theta$.

The following example illustrates this construction. We use the following convention on all repair trees: each node only indicates the update actions that are *added* to its parent, so the whole set can be read by following the branch from the root to that node.

Example 9 In the setting of Example 1, the repair tree $T_{\langle \mathcal{I},\eta\rangle}$ is partially depicted below, where $\theta = \{X/\texttt{'john'}, Y/\texttt{'0'}, Z/\texttt{'500'}\}$;[2] for legibility, the arguments `'john'` and

[2] Technically, one should also consider the substitution $\sigma = \{X/\texttt{'john'}, Y/\texttt{'500'}, Z/\texttt{'0'}\}$, as this yields a different instance of r_4; but for the purposes of building the tree the distinction is immaterial.

'basic' in the labels of the nodes are omitted.



The rightmost branch has not been fully expanded. In the picture, one can identify leaves: (i) corresponding to the three possible repairs $\mathcal{U}_1$ ((a) and (b)), $\mathcal{U}_2$ ((c) and (f)) and $\mathcal{U}_3$ ((g)), computed in different ways; (ii) corresponding to the weak repairs $\mathcal{U}_4$ ((e)) and $\mathcal{U}_5$ ((d)); and (iii) one contradictory leaf (marked ■). In particular, all repairs are included in the tree, together with some weak repairs. The missing branch contains an additional seven leaves, corresponding to $\mathcal{U}_3$ (two leaves), to the weak repair $\{-\mathsf{employee}(\text{'john'}), +\mathsf{salary}(\text{'john'}, \text{'0'}), -\mathsf{salary}(\text{'john'}, \text{'500'})\}$ (two leaves), or inconsistent (three leaves). □

Lemma 2 $T_{\langle\mathcal{I},\eta\rangle}$ is finite.

Proof
Since η is finite and there are only a finite number of individuals in $\mathcal{I}$, the number of rules not satisfied at each node is always finite. Furthermore, each rule has a finite number of literals in its body, therefore the degree of every node is finite.

If there is an edge from n to n' labeled by $r\theta$, then (by construction of n') either n' is a leaf or $I \circ n' \models r\theta$ and $\mathcal{I} \circ n'' \models r\theta$ for every descendant n'' of n'. Since the number of instances of rules is finite, this means that the depth of $T_{\langle\mathcal{I},\eta\rangle}$ is also finite.

Thus $T_{\langle\mathcal{I},\eta\rangle}$ is a finite branching tree with no infinite branches, hence by Kőnig's Lemma it is finite. □

In general, the number of nodes in $T_{\langle\mathcal{I},\eta\rangle}$ can be enormous. A simple way to prune this tree substantially is to identify nodes labeled with the same set (transforming the repair tree into a directed acyclic graph). In either case, the maximal number of nodes does not depend on the concrete database $\mathcal{I}$, as already observed by Caroprese and Truszczyński (2011). Although a graph representation is more compact, it is less understandable for human readers; therefore, this presentation will adhere to the original representation and continue to discuss (and show) repair trees.

Since inconsistent nodes cannot be weak repairs, from this point onwards we omit them from repair trees; a consistent node with only inconsistent descendants is marked by a box (■) to distinguish it from a consistent leaf.

The next result justifies the name *repair tree* for $T_{\langle\mathcal{I},\eta\rangle}$.

Theorem 7

1. Every consistent leaf of $T_{\langle\mathcal{I},\eta\rangle}$ is labeled by a weak repair for $\langle\mathcal{I},\eta\rangle$.

2. If $\mathcal{U}$ is a repair for $\langle \mathcal{I}, \eta \rangle$, then there is a branch of $T_{\langle \mathcal{I}, \eta \rangle}$ ending at a leaf labeled by $\mathcal{U}$.

Proof

1. If there is a rule in η not satisfied in node n, then either n is inconsistent or n has descendants.
2. Note that $\mathcal{U}$ is finite. The proof proceeds by showing that there is a branch of the tree whose nodes are $\emptyset = \mathcal{U}_0, \mathcal{U}_1, \ldots, \mathcal{U}_n = \mathcal{U}$, i.e. where $\mathcal{U}_{i+1}$ is obtained from $\mathcal{U}_i$ by adding an action in $\mathcal{U} \setminus \mathcal{U}_i$.
   One can always find $\mathcal{U}_{i+1}$ as described, for $i < n$. Since $\mathcal{U}$ is a repair, $\mathcal{U}_i \subsetneq \mathcal{U}$ cannot be a repair; therefore, some rules are not satisfied in $\mathcal{I} \circ \mathcal{U}_i$. Let $r\theta$ be an unsatisfied instance of one of these rules; if $ua\,(body\,(r\theta))^D \cap (\mathcal{U} \setminus \mathcal{U}_i) = \emptyset$, then $\mathcal{I} \circ \mathcal{U} \not\models r\theta$, which is absurd since $\mathcal{U}$ is a repair. Therefore, $\mathcal{U}_{i+1}$ exists. Recursive application of this argument yields the desired branch of $T_{\langle \mathcal{I}, \eta \rangle}$.
   □

In particular, if η is inconsistent, then the repair tree for $\langle \mathcal{I}, \eta \rangle$ has no consistent leaves. Thus, constructing the repair tree for $\langle \mathcal{I}, \eta \rangle$ yields a decision procedure for deciding whether there are weak repairs for $\langle \mathcal{I}, \eta \rangle$. Since the depth of the tree is polynomial in the number of grounded instances of the AICs in η, this provides an alternative proof that this problem can be solved in non-deterministic polynomial time on that set. Furthermore, by constructing all consistent leaves of the repair tree one can also find all repairs for $\langle \mathcal{I}, \eta \rangle$ – they are the labels of those leaves that are minimal w.r.t. inclusion.

5.2 Repair trees for sets of active integrity constraints

The natural adaptation of the algorithm in the previous section to AICs would be to consider only descendants of a node n that are obtained by adding to n an action in the head of an instance of a rule not satisfied in $\mathcal{I} \circ n$. However, this is too restrictive: there may be founded repairs that are no longer in the tree.

Example 10 Consider the setting of Example 4. Building the repair tree as described yields the tree in Figure 1 (left), where $\theta = \{X/\texttt{'jack'}\}$. The only leaf corresponds to the founded repair $\mathcal{U}_1$ of Example 4; however, as discussed in that example, there is another founded repair for this database. □

In order to have all founded repairs in the repair tree, one must allow all descendants obtained by applying actions that occur in the head of an instance of *any* rule in η. Case in point, rule $r_9\theta$ includes $+\texttt{hasInsurance}(\texttt{'jack'}, \texttt{'risk'})$ in its head, and this action is dual to a literal in the body of $r_{10}\theta$. Using this strategy, the repair tree for Example 4 would be the one shown in Figure 1 (right), and the new leaf (b) now contains the founded repair $\mathcal{U}_2$.

Definition 11 The *founded repair tree* for $\langle \mathcal{I}, \eta \rangle$, $T^f_{\langle \mathcal{I}, \eta \rangle}$, is obtained from $T_{\langle \mathcal{I}, \eta \rangle}$ by eliminating edges labeled with actions that do not occur on the head of any instance of a rule in η.

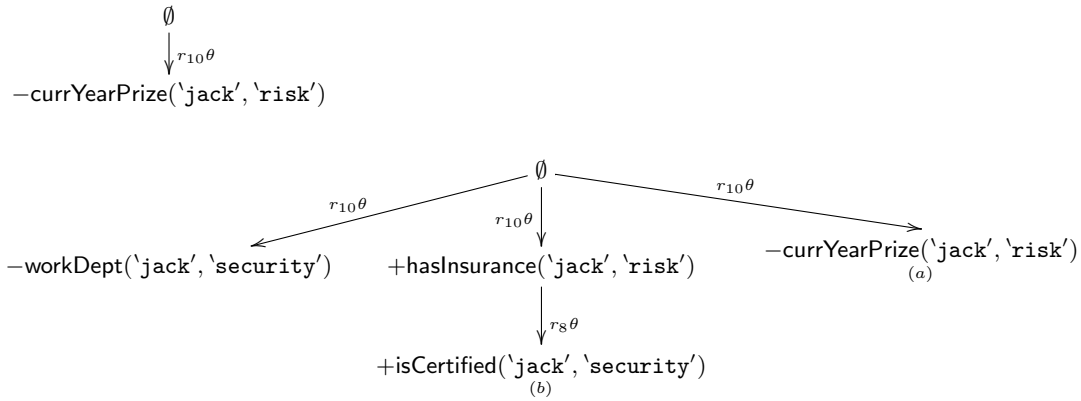


Fig. 1. Alternative possibilities for repair trees for sets of AICs.

Since $T^f_{\langle\mathcal{I},\eta\rangle}$ is a subtree of $T_{\langle\mathcal{I},\eta\rangle}$, by Lemma 2 it is finite. Furthermore, it contains all founded repairs for $\langle\mathcal{I},\eta\rangle$ by Theorem 7 and the fact that founded repairs only contain actions in the head of instances of rules in η. However, in general not all leaves correspond to founded weak repairs – nor is this to be expected, since deciding whether there is a founded repair for $\langle\mathcal{I},\eta\rangle$ is a Σ_2^P-complete problem. Hence, the best algorithm for finding founded repairs using repair trees is to choose the consistent leaves of $T^f_{\langle\mathcal{I},\eta\rangle}$ with minimal labels w.r.t. set inclusion, and to test these for foundedness.

The first attempt at a founded repair tree above actually eliminated the repair that was not justified, so one might suspect that the first construction could be used to obtain only justified weak repairs for $\langle\mathcal{I},\eta\rangle$. In fact, Example 2 shows that this is not the case; more importantly, deciding whether justified weak repairs for $\langle\mathcal{I},\eta\rangle$ exist is also Σ_2^P-complete, so it is unlikely that a tree that can be constructed in non-deterministic polynomial time can answer this question. As it turns out, one can actually prune many more spurious branches that never contain justified repairs.

The intuition behind justified repairs suggests that one should go one step further: the rule leading to the introduction of an action in a would-be repair should be the same rule supporting that action in the final repair. This is too restrictive, however, as one can easily find examples where justified repairs exist but they would not be found by this approach – this would be the case if rule r_5 in Example 2 also included $-\mathsf{category}(X, \texttt{`junior'})$ in its head.

However, the definition of justified repair does not directly refer to support for actions, but to justified action sets, which in turn look at the non-updatable literals in the rules' bodies. This motivates the use of a technique described by Antoniou et al. (2004) to keep track not only of the actions introduced at each step, but also of the non-updatable assumptions made when introducing the actions.

Definition 12 Let $\mathcal{I}$ be a database and η be a set of AICs. The *justified repair tree* for $\langle\mathcal{I},\eta\rangle$, $T^j_{\langle\mathcal{I},\eta\rangle}$, is a tree where: each node n is a pair of sets of repair actions $\langle\mathcal{U}_n,\mathcal{J}_n\rangle$ (the *update* set and the *justification* set); node n is consistent if $\mathcal{U}_n$ is consistent and $\mathcal{U}_n \cap (\mathcal{J}_n)^D = \emptyset$; each edge is labeled with an instance of a rule; the root is $\langle\emptyset,\emptyset\rangle$; and for each consistent node n and instance $r\theta$ of a rule $r \in \eta$, if $\mathcal{I} \circ \mathcal{U}_n \not\models r\theta$, then for each

$\alpha \in \mathit{head}(r\theta)$ there is a descendant n' of n, with the edge from n to n' labeled by $r\theta$, $\mathcal{U}_{n'} = \mathcal{U}_n \cup \{\alpha\}$ and $\mathcal{J}_{n'} = (\mathcal{J}_n \cup \{ua(nup(r\theta))\}) \setminus \mathcal{U}_n$.

Example 11 Consider again Example 2. Then $T^j_{\langle \mathcal{I},\eta\rangle}$ is the following, where $\theta = \{X/\texttt{`mike'}\}$. For clarity, in this example we write the labels of each node in full.

$$\emptyset, \emptyset$$
$$\downarrow r_5\theta$$
$$\{-\mathsf{onLeave}(\texttt{`mike'})\}, \{+\mathsf{category}(\texttt{`mike'}, \texttt{`junior'}), +\mathsf{workTime}(\texttt{`mike'}, \texttt{`18'})\}$$
$$\downarrow r_7\theta$$
$$\{-\mathsf{onLeave}(\texttt{`mike'}), -\mathsf{category}(\texttt{`mike'}, \texttt{`junior'})\}, \{+\mathsf{category}(\texttt{`mike'}, \texttt{`junior'}), +\mathsf{workTime}(\texttt{`mike'}, \texttt{`18'}), +\mathsf{unsupervised}(\texttt{`mike'})\}$$
■

The only leaf of this tree contains $-\mathsf{category}(\texttt{`mike'}, \texttt{`junior'})$ in its update set and the dual action in its justification set, so it is inconsistent. Therefore this tree has no consistent leaves, which agrees with the fact that there are no justified weak repairs for $\langle \mathcal{I}, \eta\rangle$. □

Indeed, every justified repair of $\langle \mathcal{I}, \eta\rangle$ is computed by $T^j_{\langle \mathcal{I},\eta\rangle}$.

Theorem 8 Let $\mathcal{U}$ be a justified repair for $\langle \mathcal{I}, \eta\rangle$. Then $T^j_{\langle \mathcal{I},\eta\rangle}$ contains a consistent leaf n such that $\mathcal{U}_n = \mathcal{U}$.

To prove this theorem it does not suffice to show that there is a branch leading to $\mathcal{U}$: there may be inconsistent leaves with $\mathcal{U}$ as update set. The usage of $\mathcal{J}$ is therefore essential to guarantee the existence of a path to a *consistent* leaf. The details of the proof are given in Appendix B.1.

As discussed above, one expects that $T^j_{\langle \mathcal{I},\eta\rangle}$ will usually contain leaves that do not correspond to justified (weak) repairs. The following example illustrates this situation.

Example 12 In the setting consisting of rules (r_8) and (r_9) from Example 4, together with

$$\mathsf{workDept}(X, \texttt{`security'}), \neg\mathsf{isCertified}(X, \texttt{`security'}), \neg\mathsf{category}(X, \texttt{`admStaff'}) \supset +\mathsf{isCertified}(X, \texttt{`security'}) \mid +\mathsf{category}(X, \texttt{`admStaff'}) \quad (r_{13})$$

$$\mathsf{secretary}(X), \neg\mathsf{category}(X, \texttt{`admStaff'}) \supset +\mathsf{category}(X, \texttt{`admStaff'}) \quad (r_{14})$$

and $\mathcal{I} = \{\mathsf{workDept}(\texttt{`Mary'}, \texttt{`security'}), \mathsf{secretary}(\texttt{`Mary'})\}$, the tree $T^j_{\langle \mathcal{I},\eta\rangle}$ contains valid leaves labeled with the following update action sets.

$$\mathcal{U}_1 = \{+\mathsf{category}(\texttt{`mary'}, \texttt{`admStaff'})\}$$
$$\mathcal{U}_2 = \{+\mathsf{isCertified}(\texttt{`mary'}, \texttt{`security'}), +\mathsf{hasInsurance}(\texttt{`mary'}, \texttt{`risk'})\} \cup \mathcal{U}_1$$

The set $\mathcal{U}_1$ is a justified repair for $\langle \mathcal{I}, \eta\rangle$ (actually, the only one), but $\mathcal{U}_2$ is a weak repair for $\langle \mathcal{I}, \eta\rangle$ that is not justified: the set $\mathcal{U}_1 \cup \mathit{ne}(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_2)$ is closed under η and contains the no-effect actions of $\mathcal{U}_2$. □

In the case of normal AICs, where heads of rules may contain at most one action, the problem of existence of justified repairs is *NP*-complete. In this case, $T^j_{\langle \mathcal{I},\eta\rangle}$ is also optimal.

Theorem 9 Let $\mathcal{I}$ be a database and η be a set of normal AICs. Then every consistent leaf of $T^j_{\langle \mathcal{I},\eta\rangle}$ contains a justified repair for $\langle \mathcal{I},\eta\rangle$.

In short: $T^j_{\langle \mathcal{I},\eta\rangle}$ contains all justified repairs and, if η is normal, it only contains justified repairs. These results cannot be improved: if η is not normal, then (1) there may be weak justified repairs that are not in $T^j_{\langle \mathcal{I},\eta\rangle}$; and (2) there may be weak justified repairs in $T^j_{\langle \mathcal{I},\eta\rangle}$, so the tree does not contain only repairs. For concrete examples see (Cruz-Filipe et al. 2013).

At this stage, direct algorithms have been introduced to compute repairs and justified repairs for inconsistent databases with (active) integrity constraints. Since the general problem of finding these repairs is at least *NP*-complete, in the worst case these algorithms are asymptotically equivalent to the techniques presented by Caroprese et al. (2006) – namely, translating the context to production rules and computing a stable model of these.

However, the general case is not the worst case, and the presentation and discussion of these algorithms serves several purposes. First, these algorithms operate directly on the database and the integrity constraints, requiring significantly less overhead in their execution. Second, they provide insight into the definitions of founded and justified repair, which are not very intuitive in the first place. Third, they allow a refinement of the notion of founded support that is interesting on its own, which we now discuss.

5.3 Well-founded repairs

Examples 2 and 4 show that circularity of support can present itself in different forms. In both examples, the repairs consist of two actions α and β such that α provides support for β and reciprocally, making these repairs not justified. However, the legitimacy of the founded repair from Example 2 can be defended by noticing that rule r_5 provides a motivation for introducing one of the problematic actions in the repair (although that same rule does not support that action at the end). This is different from the situation in Example 4, where none of the actions are "suggested" by a rule that is initially violated. This motivates looking for a new notion of repair more plausible than that of founded, but weaker than that of justified weak repair.

Definition 13 Let $\mathcal{I}$ be a database and η be a set of AICs. The *well-founded repair tree* for $\mathcal{I}$ and η, $T^{wf}_{\langle \mathcal{I},\eta\rangle}$, is constructed as $T_{\langle \mathcal{I},\eta\rangle}$ by only generating a descendant n' of a node n if $n' = n \cup \{\alpha\}$ and α occurs in the head of the instance $r\theta$ labeling the edge from n to n'.

A set of update actions $\mathcal{U}$ is a *well-founded weak repair* if $\mathcal{U}$ is a consistent leaf of $T^{wf}_{\langle \mathcal{I},\eta\rangle}$.

Since $T^j_{\langle \mathcal{I},\eta\rangle}$ is a subtree of $T^{wf}_{\langle \mathcal{I},\eta\rangle}$, we immediately get the following relation.

Lemma 3 Every justified (weak) repair for $\langle \mathcal{I}, \eta \rangle$ is a well-founded (weak) repair for $\langle \mathcal{I}, \eta \rangle$.

However, founded and well-founded weak repairs are incomparable: the well-founded repair tree for Example 4, shown in Example 10, shows that there exist founded weak repairs that are not well-founded, and the following example shows that there exist well-founded weak repairs that are not founded.

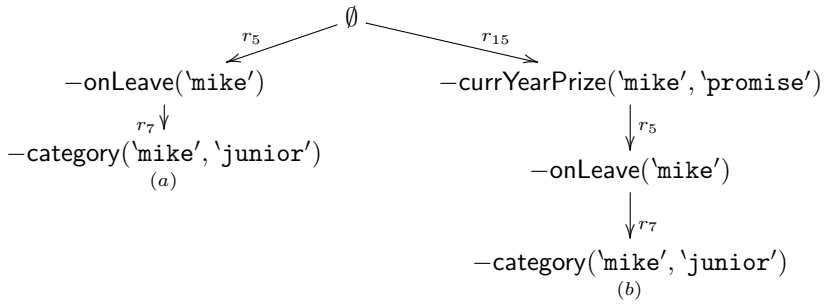
Example 13 Consider the rules in Example 2 together with

$$\mathsf{category}(X, \text{`}\mathtt{junior}'), \mathsf{onLeave}(X), \mathsf{currYearPrize}(X, \text{`}\mathtt{promise}') \supset -\mathsf{currYearPrize}(X, \text{`}\mathtt{promise}') \qquad (r_{15})$$

and the following database.

$$\mathcal{I} = \{\mathsf{category}(\text{`}\mathtt{mike}', \text{`}\mathtt{junior}'), \mathsf{onLeave}(\text{`}\mathtt{mike}'), \mathsf{workTime}(\text{`}\mathtt{mike}', \text{`}18'), \mathsf{unsupervised}(\text{`}\mathtt{mike}'),$$
$$\mathsf{workDept}(\text{`}\mathtt{mike}', \text{`}\mathtt{critical}'), \mathsf{currYearPrize}(\text{`}\mathtt{mike}', \text{`}\mathtt{promise}')\}$$

Then $T^{wf}_{\langle \mathcal{I}, \eta \rangle}$ is the tree below, and leaf (b) does not correspond to a founded weak repair. For simplicity, we omit the instantiations from the labels on the edges.



□

The parallelization results from Section 3 also extend to well-founded repairs. The proof is given in Appendix B.3.

Theorem 10 Let $\mathcal{I}$ be a database, η_1 and η_2 be two sets of AICs such that $\eta_1 \perp\!\!\!\perp \eta_2$, and $\eta = \eta_1 \cup \eta_2$.

1. If $\mathcal{U}_1$ and $\mathcal{U}_2$ are well-founded (weak) repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I}, \eta_2 \rangle$, respectively, then $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ is a well-founded (weak) repair for $\langle \mathcal{I}, \eta \rangle$.
2. Let $\mathcal{U}$ be a well-founded (weak) repair for $\langle \mathcal{I}, \eta \rangle$. Then $\mathcal{U}_1 = \mathcal{U}_{\eta_1}$ and $\mathcal{U}_2 = \mathcal{U}_{\eta_2}$ are such that $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ and each $\mathcal{U}_i$ is a well-founded (weak) repair for $\langle \mathcal{I}, \eta_i \rangle$.

However, well-founded repairs do not blend well with stratification: in the setting of Example 13, $\eta_1 = \{r_5, r_6, r_7\} \prec \{r_{15}\} = \eta_2$, but the weak repair corresponding to leaf (b) cannot be obtained by first constructing $T^{wf}_{\langle \mathcal{I}, \eta_1 \rangle}$ and then trying to extend the repair obtained using η_2, so the analog of Lemma 12 does not hold. Counter-examples involving only repairs can also be built.

For completeness, we also give a declarative semantics for well-founded weak repairs, in the style of (Caroprese et al. 2009; Caroprese and Truszczyński 2011).

Lemma 4 Given a database $\mathcal{I}$ and a set of AICs η, a set of update actions $\mathcal{U}$ is a well-founded repair w.r.t. $\langle \mathcal{I}, \eta \rangle$ if $\mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta \rangle$ and there is a sequence of actions $\alpha_1, \ldots, \alpha_n$ such that:

1. $\mathcal{U} = \{\alpha_i \mid 1 \leq i \leq n\}$;
2. for each i, α_i is founded w.r.t. $\langle \mathcal{I}, \eta \rangle$ and $\{\alpha_j \mid 1 \leq j < i\}$.

The proof of this result is straightforward, since this is simply a translation of the condition for being a consistent leaf in $T^{wf}_{\langle \mathcal{I}, \eta \rangle}$.

6 Discussion and Related Work

Theoretical bounds for the complexity of computing simple repairs, founded repairs, and justified repairs are given in (Caroprese et al. 2006; Caroprese and Truszczyński 2011). Thus, the complexity of the algorithms presented in the previous section is asymptotically optimal. For every case where finding repairs is *NP*-complete, the corresponding repair trees exactly compute the desired kinds of (weak) repairs (possibly requiring an inclusion test, which does not affect the overall complexity); in the case where this problem is Σ_2^P complete, the extra verification step is again an *NP*-complete problem. Our algorithms also provide more intuition on the different semantics of repairs, since they follow the original idea behind AICs: that the actions in their heads should "guide" the search for repairs. These algorithms are also suitable for parallel computation, since each branch is independent of the remaining ones; and all the well-known search techniques for trees can be applied, especially if one only wishes to find *one* viable repair.

In order to obtain an empirical assessment of the feasibility and impact of our contribution, we created an early-stage proof-of-concept implementation for validating and repairing SQL databases (Cruz-Filipe et al. 2016). This tool `repAIrC` has been implemented in Java, can work with any SQL database that supports JDBC, and implements repair trees as described in Section 5 for finding simple, founded, well-founded, and justified (weak) repairs. For the last, a more efficient practical criterion for validating justified repairs is used, as constructing the set of no-effect actions directly requires processing the entire database and is, thus, prohibitive.

The performance of `repAIrC` is dominated by the many database interactions required for building the repair trees. For small sets of AICs the performance is typically acceptable, with runtimes in the sub-second or seconds range. For larger sets of AICs, the high worst-case complexity of our algorithms suggests that it should pay off to split them into smaller ones. Thus, our tool implements the parallelization and stratification algorithms described in Sections 3 and 4. Indeed, these have been shown empirically to provide four orders of magnitude improvements for some sets of AICs – see (Cruz-Filipe et al. 2016) for details.

Our parallelization and stratification techniques are reminiscent of similar techniques in nearby fields. Semantic independency and syntactic precedence were introduced by Naqvi and Krishnamurthy (1988), and the latter is used to compute models in the more general scenario of deductive databases. In logic programming, notions of syntactic independence allowing parallel evaluation of rules were also discussed by Wolfson and Silberschatz (1991). Syntactic precedence between integrity constraints was also discussed with the

explicit goal of making the search for database repairs more efficient (Mayol and Teniente 1999a), but in contrast to AICs the authors did not allow for cyclic dependencies.

More recently, Bravo et al. (2007) and Fan et al. (2014) use graphs to model the dependencies in the context of cell-based updates (Fan and Geerts 2012). Parallelization is likewise exploited in this setting (Geerts et al. 2013). The use of tuple-generating dependencies allows to also model insertions in (Bohannon et al. 2005) and (Geerts et al. 2014). Our work can be seen as extending this line of work to the setting of AICs.

Approximation fixpoint theory unifies several semantics for logical frameworks, and generic stratification techniques have also been considered in this more general domain by Vennekens et al. (2006). The application of approximation fixpoint theory to AICs by Bogaerts and Cruz-Filipe (2018) yields an alternative characterization of all existing AIC semantics (as well as defining some new ones) in terms of fixpoints of a particular operator. Preliminary results suggest that the stratification developed in this work exactly corresponds to the application of the general formalism for approximation fixpoint theory to the AIC operator from (Bogaerts and Cruz-Filipe 2018).

Our original motivation for studying the framework of AICs was to develop a theory of integrity constraints and repairs for general-purpose knowledge bases. The work described in this article is a first step in this direction. AICs have been generalized to heterogenous non-monotonic multi-context systems (Brewka and Eiter 2007; Cruz-Filipe et al. 2016; Cruz-Filipe et al. 2016) following the ideas presented here.

7 Conclusions and Future Work

This paper builds on previous work by Flesca et al. (2004) and Caroprese and Truszczyński (2011), with emphasis on the operational aspects of active integrity constraints. As such, it first introduces independence and precedence relations between AICs, which allow parallelization and sequentialization of the computation of repairs for inconsistent databases, in turn speeding up the process of finding these repairs – a problem that is typically *NP*- or Σ_p^2-complete. In the worst case scenario, there will be no parallelization or sequentialization; however, in typical databases concepts are generally built from more primitive ones, suggesting that this division can play a key role in making the search for repairs much faster. Furthermore, both relations are well behaved w.r.t. the different kinds of repairs considered in the denotational semantics for AICs (Caroprese and Truszczyński 2011).

A second part of the paper concerns an operational semantics for AICs, showing how repairs can be computed by building an adequate tree. Different pruning mechanisms for the branches of the tree yield different types of repairs. Each type of tree is complete, in the sense that it contains as leaves all repairs of a given type, but in some cases these leaves must be checked *a posteriori*. This characteristic is to be expected, since these tree algorithms are all polynomial but some of these problems are Σ_p^2-complete.

The study of these tree algorithms also led to the introduction of well-founded repair trees. They provide a more fine-grained characterization of non-justified repairs, distinguishing essential circularity of support from support that is indeed circular when one considers the final repair, but can be motivated by the heads of violated active integrity constraints.

Acknowledgements

This work was partially supported by Fundação para a Ciência e Tecnologia under contract PEst-OE/EEI/UI0434/2011 and by the Danish Council for Independent Research, Natural Sciences, grants DFF-1323-00247 and DFF-7014-00041.

References

Abiteboul, S. 1988. Updates, a new frontier. In *ICDT*, M. Gyssens, J. Paredaens, and D. van Gucht, Eds. LNCS, vol. 326. Springer–Verlag, 1–18.

Abiteboul, S., Hull, R., and Vianu, V. 1995. *Foundations of Databases.* Addison Wesley.

Antoniou, G., Damasio, C., Grosof, B., Horrocks, I., Kifer, M., Maluszynski, J., and Patel-Schneider, P. 2004. Combining rules and ontologies: A survey. Technical Report IST506779/Linköping/I3-D3/D/PU/a1, Linköping University. available at `http://rewerse.net/publications/`.

Beeri, C. and Vardi, M. 1981. The implication problem for data dependencies. In *Proceedings of the 8th Colloquium on Automata, Languages and Programming.* LNCS, vol. 115. Springer-Verlag, London, UK, 73–85.

Bogaerts, B. and Cruz-Filipe, L. 2018. Fixpoint semantics for active integrity constraints. *Artificial Intelligence 255*, 43–70.

Bohannon, P., Flaster, M., Fan, W., and Rastogi, R. 2005. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD 2005*, F. Özcan, Ed. ACM, 143–154.

Bravo, L., Fan, W., and Ma, S. 2007. Extending dependencies with conditions. In *VLDB 2007*, C. Koch, J. Gehrke, M. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C. Chan, V. Ganti, C. Kanne, W. Klas, and E. Neuhold, Eds. ACM, 243–254.

Brewka, G. and Eiter, T. 2007. Equilibria in heterogeneous nonmonotonic multi-context systems. In *AAAI2007*. AAAI Press, 385–390.

Caroprese, L., Greco, S., Sirangelo, C., and Zumpano, E. 2006. Declarative semantics of production rules for integrity maintenance. In *ICLP*, S. Etalle and M. Truszczynski, Eds. LNCS, vol. 4079. Springer, 26–40.

Caroprese, L., Greco, S., and Zumpano, E. 2009. Active integrity constraints for database consistency maintenance. *IEEE Trans. Knowl. Data Eng. 21,* 7, 1042–1058.

Caroprese, L., Trubitsyna, I., Truszczynski, M., and Zumpano, E. 2012. The view-update problem for indefinite databases. In *JELIA*, L. Fariñas del Cerro, A. Herzig, and J. Mengin, Eds. LNCS, vol. 7519. Springer, 134–146.

Caroprese, L. and Truszczyński, M. 2011. Active integrity constraints and revision programming. *Theory Pract. Log. Program. 11,* 6 (Nov.), 905–952.

Chomicki, J. 2007. Consistent query answering: Five easy pieces. In *ICDT*, T. Schwentick and D. Suciu, Eds. LNCS, vol. 4353. Springer, 1–17.

Cruz-Filipe, L. 2014. Optimizing computation of repairs from active integrity constraints. In *FoIKS 2014*, C. Beierle and C. Meghini, Eds. LNCS, vol. 8367. Springer–Verlag, 361–380.

Cruz-Filipe, L., Engrácia, P., Gaspar, G., and Nunes, I. 2013. Computing repairs from active integrity constraints. In *TASE2013*, H. Wang and R. Banach, Eds. IEEE, 183–190.

Cruz-Filipe, L., Franz, M., Hakhverdyan, A., Ludovico, M., Nunes, I., and Schneider-Kamp, P. 2016. Active integrity constraints: From theory to implementation. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, A. L. Fred, J. L. Dietz, D. Aveiro, K. Liu, and J. Filipe, Eds. CCIS, vol. 631. Springer, 399–420.

Cruz-Filipe, L., Gaspar, G., Nunes, I., and Schneider-Kamp, P. 2016. Active integrity constraints for multi-context systems. In *EKAW 2016*, E. Blomqvist, F. Vitali, P. Ciancarini, and F. Poggi, Eds. LNAI, vol. 10024. Springer, 98–112.

Cruz-Filipe, L., Nunes, I., and Schneider-Kamp, P. 2016. Integrity constraints for general-purpose knowledge bases. In *FoIKS 2016*, M. Gyssens and G. Simari, Eds. LNCS, vol. 9616. Springer, 235–254.

Eiter, T. and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artif. Intell. 57,* 2–3, 227–270.

Fan, W. and Geerts, F. 2012. *Foundations of Data Quality Management.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Fan, W., Ma, S., Tang, N., and Yu, W. 2014. Interaction between record matching and data repairing. *J. Data and Information Quality 4,* 4, 16:1–16:38.

Flesca, S., Greco, S., and Zumpano, E. 2004. Active integrity constraints. In *PPDP*, E. Moggi and D. Scott Warren, Eds. ACM, 98–107.

Geerts, F., Mecca, G., Papotti, P., and Santoro, D. 2013. The LLUNATIC data-cleaning framework. *PVLDB 6,* 9, 625–636.

Geerts, F., Mecca, G., Papotti, P., and Santoro, D. 2014. Mapping and cleaning. In *ICDE 2014*, I. Cruz, E. Ferrari, Y. Tao, E. Bertino, and G. Trajcevski, Eds. IEEE Computer Society, 232–243.

Kakas, A. and Mancarella, P. 1990. Database updates through abduction. In *VLDB 1990*, D. McLeod, R. Sacks-Davis, and H.-J. Schek, Eds. Morgan Kaufmann, 650–661.

Marek, V. and Truszczynski, M. 1995. Revision programming, database updates and integrity constraints. In *ICDT*, G. Gottlob and M. Vardi, Eds. LNCS, vol. 893. Springer–Verlag, 368–382.

Mayol, E. and Teniente, E. 1999a. Addressing efficiency issues during the process of integrity maintenance. In *DEXA*, T. Bench-Capon, G. Soda, and A. Tjoa, Eds. LNCS, vol. 1677. Springer–Verlag, 270–281.

Mayol, E. and Teniente, E. 1999b. A survey of current methods for integrity constraint maintenance and view updating. In *ER (Workshops)*, P. Chen, D. Embley, J. Kouloumdjian, S. Liddle, and J. Roddick, Eds. LNCS, vol. 1727. Springer, 62–73.

Naqvi, S. and Krishnamurthy, R. 1988. Database updates in logic programming. In *PODS 1988*, C. Edmondson-Yurkanan and M. Yannakakis, Eds. ACM, 251–262.

Przymusinski, T. and Turner, H. 1997. Update by means of inference rules. *J. Log. Program. 30,* 2, 125–143.

Teniente, E. and Olivé, A. 1995. Updating knowledge bases while maintaining their consistency. *VLDB J. 4,* 2, 193–241.

Thalheim, B. 1991. *Dependencies in Relational Databases.* Teubner-Texte zur Mathematik. B.G. Teubner.

Vennekens, J., Gilis, D., and Denecker, M. 2006. Splitting an operator: Algebraic modularity results for logics with fixpoint semantics. *ACM Trans. Comput. Log. 7,* 4, 765–797.

Widom, J. and Ceri, S., Eds. 1996. *Active Database Systems: Triggers and Rules For Advanced Database Processing.* Morgan Kaufmann.

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

Wolfson, O. and Silberschatz, A. 1991. Decomposability and its role in parallel logic program evaluation. *J. Log. Program. 11,* 3&4, 345–358.

Appendix A Proofs of results on parallelization

A.1 Proof of Theorem 1

We first consider the case of two independent sets; the main result follows by induction. The first lemma is a technical remark that we use repeatedly in proofs.

Lemma 5 Let $\mathcal{I}$ be a database, η_1, η_2 be independent sets of ICs, and $\mathcal{U}_1$ and $\mathcal{U}_2$ be sets of update actions such that: if $\alpha \in \mathcal{U}_i$, then there is $L \in \mathit{body}\,(r\theta)$ for some instance of a rule $r \in \eta_i$ with $|\mathit{lit}(\alpha)| = |L|$. Take $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$. For every literal L such that $L \in \mathit{body}\,(r\theta)$ for some instance $r\theta$ of a rule $r \in \eta_i$, $\mathcal{I} \circ \mathcal{U} \models L$ iff $\mathcal{I} \circ \mathcal{U}_i \models L$. In particular, for every $r \in \eta_i$, $\mathcal{I} \circ \mathcal{U} \models r$ iff $\mathcal{I} \circ \mathcal{U}_i \models r$.

Proof
Let $L \in \mathit{body}\,(r\theta)$ for some $r \in \eta_1$ and θ. If $\alpha \in \mathcal{U}_2$, then $|\mathit{lit}(\alpha)| \neq |L|$ because $\eta_1 \perp \eta_2$, whence $\mathcal{I} \circ \mathcal{U} \models L$ iff $\mathcal{I} \circ \mathcal{U}_1 \models L$ (as $\mathcal{I} \circ \mathcal{U} = (\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2$). This holds for every θ, so $\mathcal{I} \circ \mathcal{U} \models r$ iff $\mathcal{I} \circ \mathcal{U}_1 \models r$. The argument for $\mathcal{U}_2$ is similar. □

Lemma 6 In the conditions of Lemma 5, if $\mathcal{U}_1$ and $\mathcal{U}_2$ are weak repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I}, \eta_2 \rangle$, respectively, then $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ is a weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$. Furthermore, if $\mathcal{U}_1$ and $\mathcal{U}_2$ are both repairs, then so is $\mathcal{U}$.

Proof
Consistency of $\mathcal{U}$ follows from the disjointness of the sets of atoms underlying the actions in $\mathcal{U}_1$ and $\mathcal{U}_2$, which is a consequence of the hypothesis and $\eta_1 \perp \eta_2$. Hence, if $\{+\alpha, -\alpha\} \subseteq \mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$, then $\{+\alpha, -\alpha\} \subseteq \mathcal{U}_i$ for some i, and $\mathcal{U}_i$ would be inconsistent. Furthermore, if $\alpha \in \mathcal{U}_i$ then α must change $\mathcal{I}$ (since $\mathcal{U}_i$ is a weak repair for $\langle \mathcal{I}, \eta_i \rangle$), so every action in $\mathcal{U}$ changes $\mathcal{I}$.

To see that $\mathcal{U}$ is a weak repair, consider $r \in \eta_1$. Then $\mathcal{I} \circ \mathcal{U}_1 \models r$, since $\mathcal{U}_1$ is a weak repair for $\langle \mathcal{I}, \eta_1 \rangle$, whence $\mathcal{I} \circ \mathcal{U} \models r$ by Lemma 5. The argument for $r \in \eta_2$ is similar.

Now assume that $\mathcal{U}_1$ and $\mathcal{U}_2$ are both repairs. Let $\mathcal{U}' \subsetneq \mathcal{U}$ and define $\mathcal{U}'_i = \mathcal{U}' \cap \mathcal{U}_i$ for $i = 1, 2$. One of the inclusions $\mathcal{U}'_i \subseteq \mathcal{U}_i$ must be strict; without loss of generality, assume that $\mathcal{U}'_1 \subsetneq \mathcal{U}_1$. Since $\mathcal{U}_1$ is a repair, this means that $\mathcal{U}'_1$ cannot be a weak repair, hence there is a rule $r \in \eta_1$ such that $\mathcal{U}'_1 \not\models r$. By Lemma 5 $\mathcal{U}'_1 \cup \mathcal{U}'_2 \not\models r$, hence $\mathcal{U}' = \mathcal{U}'_1 \cup \mathcal{U}'_2$ is not a weak repair for $\langle \mathcal{I}, \eta \rangle$. Therefore $\mathcal{U}$ is a repair for $\langle \mathcal{I}, \eta \rangle$. □

The hypotheses of Lemma 6 imply that the actions in each $\mathcal{U}_i$ are all duals of literals in the body of some rule in η_i. This is an essential requirement: otherwise $\mathcal{U}_1$ could "break" satisfaction of some rule in η_2 or reciprocally, or there might be inconsistencies from joining $\mathcal{U}_1$ and $\mathcal{U}_2$. Although it can be weakened, this hypothesis is a (very) reasonable assumption, since this verification can be done efficiently. See also the discussion in Section 5 of (Caroprese et al. 2009).

The converse result also holds: splitting the actions in a (weak) repair $\mathcal{U}$ according to whether they affect rules in η_1 or η_2 yields (weak) repairs for those sets of ICs.

Lemma 7 Let $\mathcal{I}$ be a database, η_1 and η_2 be independent sets of ICs, $\eta = \eta_1 \cup \eta_2$, and $\mathcal{U}$ be a weak repair for $\langle \mathcal{I}, \eta \rangle$. Then each

$$\mathcal{U}_i = \{\alpha \in \mathcal{U} \mid \mathit{lit}(\alpha)^D \in \mathit{body}\,(r\theta) \text{ for some instance } r\theta \text{ of } r \in \eta_i\}$$

is a weak repairs for $\langle \mathcal{I}, \eta_i \rangle$. Furthermore, if every action in $\mathcal{U}$ is the dual of a literal in the body of an instance of some rule in η, then $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$. In particular, if $\mathcal{U}$ is a repair, then so are $\mathcal{U}_1$ and $\mathcal{U}_2$.

Proof
Assume that $\mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta \rangle$ and let $\mathcal{U}_i$ be as stated. Since $\mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta \rangle$, $\mathcal{I} \circ \mathcal{U} \models r$ for every rule $r \in \eta_i$. By Lemma 5, $\mathcal{I} \circ \mathcal{U}_i \models r$. Therefore $\mathcal{U}_i$ is a weak repair for $\langle \mathcal{I}, \eta_i \rangle$.

The equality $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ holds if every action in $\mathcal{U}$ satisfies the condition defining $\mathcal{U}_i$ for some i, i.e. it is the dual of a literal in the body of an instance of some rule in η. In particular, this holds if $\mathcal{U}$ is a repair. Assume that this is the case, and suppose that $\mathcal{U}'_1 \subsetneq \mathcal{U}_1$ is also a weak repair for $\langle \mathcal{I}, \eta_1 \rangle$. By Lemma 5, $\mathcal{U}' = \mathcal{U}'_1 \cup \mathcal{U}_2$ is a weak repair for $\langle \mathcal{I}, \eta \rangle$ with $\mathcal{U}' \subsetneq \mathcal{U}$, which is absurd. Therefore $\mathcal{U}'_1$ is not a weak repair, hence $\mathcal{U}_1$ is a repair. The case for $\mathcal{U}_2$ is similar. □

The equality $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ can be made to hold in general by adding the actions that do not affect any rule to either $\mathcal{U}_1$ or $\mathcal{U}_2$; however, this is not an interesting situation, and it will not be considered further.

Theorem 1
By induction on n. For $n = 1$, the results are trivial. Assume the result for n; applying the induction hypothesis to $\eta_1, \ldots, \eta_n$ and Lemmas 6 or 7 to $\eta' = \bigcup_{i=1}^{n} \eta_i$ and η_{n+1} yields the result for $\eta_1, \ldots, \eta_{n+1}$, since $\eta' \perp \eta_{n+1}$. □

A.2 Proof of Theorem 4

Again we divide the proof in several lemmas. Throughout this section, we assume $\mathcal{I}$ to be database, η_1 and η_2 to be sets of AICs with $\eta_1 \perp\!\!\!\perp \eta_2$, and $\eta = \eta_1 \cup \eta_2$.

Lemma 8 If $\mathcal{U}_1$ and $\mathcal{U}_2$ are sets of update actions founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I}, \eta_2 \rangle$, respectively, then $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ is founded w.r.t. $\langle \mathcal{I}, \eta \rangle$.

Proof
For $\mathcal{U}$ to be founded w.r.t. $\langle \mathcal{I}, \eta \rangle$, every action in $\mathcal{U}$ must be founded w.r.t. $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$. Without loss of generality, assume $\alpha \in \mathcal{U}_1$. Since $\mathcal{U}_1$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$, there is an instance $r\theta$ of $r \in \eta_1$ such that $\alpha \in \mathit{head}\,(r\theta)$ and $\mathcal{I} \circ \mathcal{U}_1 \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$. By Lemma 5, $I \circ \mathcal{U} \models L$ for every such L. Since $\eta_1 \subseteq \eta$, α is founded w.r.t. $\langle \mathcal{I}, \eta \rangle$ and $\mathcal{U}$. □

Corollary 1 If $\mathcal{U}_1$ and $\mathcal{U}_2$ are founded (weak) repairs, then $\mathcal{U}$ is also a founded (weak) repair.

Proof
Consequence of Lemmas 6 and 8. □

Lemma 9 Let $\mathcal{U}$ be a set of update actions founded w.r.t. $\langle \mathcal{I}, \eta \rangle$. Then $\mathcal{U}_1 = \mathcal{U}_{\eta_1}$ and $\mathcal{U}_2 = \mathcal{U}_{\eta_2}$ are such that $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ and each $\mathcal{U}_i$ is founded w.r.t. $\langle \mathcal{I}, \eta_i \rangle$.

Proof
Without loss of generality, assume $\alpha \in \mathcal{U}_1$. Since $\mathcal{U}$ is founded w.r.t. $\langle \mathcal{I}, \eta \rangle$, there is an instance $r\theta$ of $r \in \eta$ such that $\alpha \in \mathit{head}\,(r\theta)$ and $\mathcal{I} \circ \mathcal{U} \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$. But $\alpha \in \mathit{head}\,(r\theta)$ implies that $r \in \eta_1$, whence $\mathcal{I} \circ \mathcal{U}_1 \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$ by Lemma 5. Therefore α is founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_1$, whence $\mathcal{U}_1$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$.

By definition of founded set, all actions in $\mathcal{U}$ must be in either $\mathcal{U}_1$ or $\mathcal{U}_2$, so $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$. □

Corollary 2 If $\mathcal{U}$ is a (weak) founded repair, then $\mathcal{U}_{\eta_1}$ and $\mathcal{U}_{\eta_2}$ are also (weak) founded repairs.

Proof
Consequence of Lemmas 7 and 9. □

Similar results hold for justified (weak) repairs, although the proofs are a bit more involved.

Lemma 10 If $\mathcal{U}_1$ and $\mathcal{U}_2$ are justified (weak) repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I}, \eta_2 \rangle$, respectively, then $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ is a justified (weak) repair for $\langle \mathcal{I}, \eta \rangle$.

Proof
The following simple facts will be used recurrently throughout the proof.

(a) For $i = 1, 2$, $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) \subseteq ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_i)$, since $\mathcal{U}_i \subseteq \mathcal{U}$.
(b) For $i = 1, 2$, $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_i) \subseteq (ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) \cup \mathcal{U}_{3-i})$: $\mathcal{U}$ can only change literals changed by either $\mathcal{U}_1$ or $\mathcal{U}_2$. In particular, since $\eta_1 \perp\!\!\!\perp \eta_2$, if $\mathit{nup}(r\theta) \subseteq \mathit{lit}(ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_i))$ for some instance $r\theta$ of $r \in \eta_i$, then $\mathit{nup}(r\theta) \subseteq \mathit{lit}(ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$; and if $\alpha \in \mathit{head}\,(r\theta)$ for some instance $r\theta$ of $r \in \eta_i$ and $\alpha \in ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_i)$, then $\alpha \in ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$.
(c) For $i = 1, 2$, if $L \in \mathit{body}\,(r\theta)$ with $r \in \eta_i$ and $L \in \mathit{lit}(\mathcal{U})$, then $L \in \mathit{lit}(\mathcal{U}_i)$: $\eta_1 \perp\!\!\!\perp \eta_2$ and $\mathcal{U}_i$ only contains actions in the heads of rules of η_i.

The first step is to show that $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for η. Without loss of generality, assume $r \in \eta_1$. Suppose that $\mathit{nup}(r\theta) \subseteq \mathit{lit}\,(\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$, and let $L \in \mathit{nup}(r\theta)$. If $L \in \mathit{lit}(\mathcal{U})$, then $L \in \mathit{lit}(\mathcal{U}_1)$ by (c), hence $\mathit{nup}(r\theta) \subseteq \mathit{lit}\,(\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$, whence $\mathit{nup}(r\theta) \subseteq \mathit{lit}(\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1))$ by (a). But $\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ is closed for η_1, so $\mathit{head}\,(r\theta) \cap (\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)) \neq \emptyset$. By $\mathcal{U}_1 \subseteq \mathcal{U}$ and (b), also $\mathit{head}\,(r\theta) \cap (\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$.

For minimality, suppose that $\mathcal{U}' \subsetneq \mathcal{U}$ and $\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for η. Take $\mathcal{U}'_i = \mathcal{U}' \cap \mathcal{U}_i$ for $i = 1, 2$; then one of the inclusions $\mathcal{U}'_i \subseteq \mathcal{U}_i$ must be strict. Without loss of generality, assume this holds for $i = 1$, and take $r \in \eta_1$. If $\mathit{nup}(r\theta) \subseteq \mathit{lit}\,(\mathcal{U}'_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1))$, then $\mathit{nup}(r\theta) \subseteq \mathit{lit}\,(\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$, consequence of $\mathcal{U}'_1 \subseteq \mathcal{U}'$ and (b). Since $\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for η and $\eta_1 \subseteq \eta$, it follows that $\mathit{head}\,(r\theta) \cap (\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$. By definition of $\mathcal{U}_1$ and (a), it follows that $\mathit{head}\,(r\theta) \cap (\mathcal{U}'_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)) \neq \emptyset$. Then $\mathcal{U}'_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ is closed for η_1, contradicting minimality of $\mathcal{U}_1$.

Hence $\mathcal{U}$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$. By Lemma 6, if $\mathcal{U}_1$ and $\mathcal{U}_2$ are both justified repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I}, \eta_2 \rangle$, respectively, then $\mathcal{U}$ is also a justified repair for $\langle \mathcal{I}, \eta \rangle$. □

Lemma 11 If $\mathcal{U}$ is a justified (weak) repair for $\langle \mathcal{I}, \eta \rangle$, then $\mathcal{U}_1 = \mathcal{U}_{\eta_1}$ and $\mathcal{U}_2 = \mathcal{U}_{\eta_2}$ are such that $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ and each $\mathcal{U}_i$ is a justified (weak) repair for $\langle \mathcal{I}, \eta_i \rangle$.

Proof
Properties (a), (b) and (c) from the previous proof still hold. To show that $\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ is closed under η_1, take $r \in \eta_1$ and suppose that $nup(r\theta) \subseteq lit\,(\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1))$. Then $nup(r\theta) \subseteq lit(\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$ by $\mathcal{U}_1 \subseteq \mathcal{U}$ and (b). Since $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for η, it follows that $head\,(r\theta) \cap (\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$. By construction of $\mathcal{U}_1$ and (a), $head\,(r\theta) \cap (\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)) \neq \emptyset$. The case for $\mathcal{U}_2$ is similar.

For minimality, suppose that $\mathcal{U}_1' \subsetneq \mathcal{U}_1$ and $\mathcal{U}_1' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ is closed for η_1; take $\mathcal{U}' = \mathcal{U}_1' \cup \mathcal{U}_2$, let $r \in \eta$, and assume that $nup(r\theta) \subseteq lit(\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$. There are two cases.

- Suppose $r \in \eta_1$ and let $L \in nup(r\theta)$. Then $L \notin lit(\mathcal{U}_2)$, since $\eta_1 \perp\!\!\!\perp \eta_2$, therefore $nup(r\theta) \subseteq lit\,(\mathcal{U}_1' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$, whence by (a) $nup(r\theta) \subseteq lit\,(\mathcal{U}_1' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1))$, and therefore $head\,(r\theta) \cap (\mathcal{U}_1' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)) \neq \emptyset$. From $\mathcal{U}_1' \subseteq \mathcal{U}'$ and (b), also $head\,(r\theta) \cap (\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$.
- Suppose $r \in \eta_2$ and let $L \in nup(r\theta)$. Again $L \notin lit(\mathcal{U}_1')$, whence $L \in \mathcal{U}_2 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$; and since $\mathcal{U}_2 \subseteq \mathcal{U}$ also $nup(r\theta) \subseteq lit(\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$. Since $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for η (which contains η_2), it follows that $head\,(r\theta) \cap (\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$, and since $head\,(r\theta)$ does not contain actions in $\mathcal{U}_1$ necessarily $head\,(r\theta) \cap (\mathcal{U}_2 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$, whence $head\,(r\theta) \cap (\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$.

In either case, if $nup(r\theta) \subseteq (\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$ then $head\,(r\theta) \cap (\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$, whence $\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for η, contradicting minimality of $\mathcal{U}$. This is absurd, so $\mathcal{U}_1$ is a justified weak repair. The case for $\mathcal{U}_2$ is similar.

Since justified weak repairs are founded, Lemma 9 guarantees that $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$. Furthermore, if $\mathcal{U}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$, then each $\mathcal{U}_i$ is a justified repair for $\langle \mathcal{I}, \eta_i \rangle$ by Lemma 7. □

A.3 Proof of Theorem 5

We divide the proof in several lemmas for convenience. Throughout this section, let $\eta_1, \eta_2 \subseteq \eta$ with $\eta_1 \prec \eta_2$, $\mathcal{I}$ be a database and $\mathcal{U}$ be a set of update actions such that all actions in $\mathcal{U}$ occur in the head of some instance of a rule in $\eta_1 \cup \eta_2$, and take $\mathcal{U}_1 = \mathcal{U}_{\eta_1}$ and $\mathcal{U}_2 = \mathcal{U}_{\eta_2}$.

Lemma 12 If $\mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$, then $\mathcal{U}_1$ and $\mathcal{U}_2$ are weak repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, respectively.

Proof
Since $\eta_1 \prec \eta_2$, (a) actions in the head of an instance of a rule in η_2 cannot refer to atoms underlying literals in the body of instances of rules in η_1, and in particular (b) $\mathcal{U}_1$ and $\mathcal{U}_2$ are disjoint. By (a), $\mathcal{I} \circ \mathcal{U}_1 \models r$ iff $\mathcal{I} \circ \mathcal{U} \models r$ for every $r \in \eta_1$, so $\mathcal{U}_1$ is a weak repair for $\langle \mathcal{I}, \eta_1 \rangle$. By (b), $\mathcal{I} \circ \mathcal{U} = \mathcal{I} \circ (\mathcal{U}_1 \cup \mathcal{U}_2) = (\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2$, hence $\mathcal{U}_2$ is a weak repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$. □

Lemma 13 If $\mathcal{U}$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$, then $\mathcal{U}_1$ and $\mathcal{U}_2$ are founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, respectively.

Proof
(i) Let $\alpha \in \mathcal{U}_1$. Since $\mathcal{U}$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$, there is an instance $r\theta$ of a rule $r \in \eta_1 \cup \eta_2$ such that $\alpha \in \mathit{head}\,(r\theta)$ and $\mathcal{I} \circ \mathcal{U} \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$. Since $\eta_1 \prec \eta_2$, $r \in \eta_1$. By (b) from the proof of Lemma 12, $\mathcal{I} \circ \mathcal{U}_1 \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$, whence α is founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_1$. Thus $\mathcal{U}_1$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$.

(ii) Let $\alpha \in \mathcal{U}_2$. Again there must be an instance $r\theta$ of a rule $r \in \eta$ such that $\alpha \in \mathit{head}\,(r\theta)$ and $\mathcal{I} \circ \mathcal{U} \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$, and as before necessarily $r \in \eta_2$. Since $I \circ \mathcal{U} = (\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2$, it follows that α is founded w.r.t. $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$ and $\mathcal{U}_2$, hence $\mathcal{U}_2$ is founded w.r.t. $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$. □

Corollary 3 If $\mathcal{U}$ is a founded weak repair for $\langle \mathcal{I}, \eta \rangle$, then $\mathcal{U}_1$ and $\mathcal{U}_2$ are founded weak repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, respectively.

Proof
Immediate consequence of Lemmas 12 and 13. □

Lemma 14 If $\mathcal{U}$ is a justified weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$, then $\mathcal{U}_1$ and $\mathcal{U}_2$ are justified weak repairs for $\langle \mathcal{I}, \eta_1 \rangle$ and $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, respectively.

Proof
(i) The proof that $\mathcal{U}_1$ is a justified weak repair for $\langle \mathcal{I}, \eta_1 \rangle$ is as in Lemma 11.

(ii) Denote by $\mathcal{N}$ the set $\mathsf{ne}\,(\mathcal{I} \circ \mathcal{U}_1, \mathcal{I} \circ \mathcal{U}_1 \circ \mathcal{U}_2)$. To show that $\mathcal{U}_2$ is a justified weak repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, we need to show that $\mathcal{U}_2 \cup \mathcal{N}$ is closed for η_2 and that it is the minimal such set containing $\mathcal{N}$. The first part is again as in the corresponding step of the proof of Lemma 11.

Now let $\mathcal{U}_2' \subsetneq \mathcal{U}_2$ be such that $\mathcal{U}_2' \cup \mathcal{N}$ is closed for η_2 and take $\mathcal{U}' = \mathcal{U}_1 \cup \mathcal{U}_2'$. Then $\mathcal{U}' \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed under η: let $r \in \eta$ be such that $\mathit{nup}(r\theta) \subseteq \mathit{lit}(\mathcal{U}' \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$; there are two cases to consider.

- $r \in \eta_1$: since $\mathcal{U}' \subseteq \mathcal{U}$, also $\mathit{nup}(r\theta) \subseteq \mathit{lit}(\mathcal{U} \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$, whence $\mathit{head}\,(r\theta) \cap (\mathcal{U} \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$ because $\mathcal{U} \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for η. But actions in $\mathit{head}\,(r\theta)$ may not occur in $\mathcal{U}_2$, hence $\mathit{head}\,(r\theta) \cap (\mathcal{U}' \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$ since $(\mathcal{U} \setminus \mathcal{U}') \subseteq \mathcal{U}_2$.
- $r \in \eta_2$: note that $\mathcal{N} = \mathcal{U}_1 \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$, since $I \circ \mathcal{U}_1$ is "between" $\mathcal{I}$ and $\mathcal{I} \circ \mathcal{U}$ (as $\mathcal{U}_1 \subseteq \mathcal{U}$); therefore, $\mathcal{U}' \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) = \mathcal{U}_1 \cup \mathcal{U}_2' \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) = \mathcal{U}_2' \cup \mathcal{N}$, whence $\mathit{head}\,(r\theta) \cap (\mathcal{U}_2' \cup \mathcal{N}) \neq \emptyset$ because $\mathcal{U}_2' \cup \mathcal{N}$ is closed for η_2, which amounts to saying that that $\mathit{head}\,(r\theta) \cap (\mathcal{U}' \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$.

In either case, $\mathit{head}\,(r\theta) \cap (\mathcal{U}' \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$, so $\mathcal{U}' \cup \mathit{ne}\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed under η, contradicting the fact that $\mathcal{U}$ is a justified weak repair for $\langle \mathcal{I}, \eta \rangle$. Therefore $\mathcal{U}_2$ is a justified weak repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$. □

Theorem 5
Combination of Lemma 12, Corollary 3 and Lemma 14. □

A.4 Proof of Theorem 6

Again we divide the proof in several lemmas. Throughout this section, let $\eta_1, \eta_2 \subseteq \eta$ with $\eta_1 \prec \eta_2$, $\mathcal{I}$ be a database, and $\mathcal{U}_1$ and $\mathcal{U}_2$ be sets of update actions such that all actions in $\mathcal{U}_i$ occur in the head of some instance of a rule in η_i. Define $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

Lemma 15 If $\mathcal{U}_1$ is a weak repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is a weak repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.

Proof
Since $\eta_1 \prec \eta_2$, the hypothesis over $\mathcal{U}_2$ imply that (a) actions in $\mathcal{U}_2$ cannot refer to atoms underlying literals in the body of instances of rules in η_1, and in particular (b) $\mathcal{U}_1$ and $\mathcal{U}_2$ are disjoint. If $r \in \eta_1$, then $\mathcal{I} \circ \mathcal{U}_1 \models r$, whence $\mathcal{I} \circ \mathcal{U} \models r$ by (a). If $r \in \eta_2$, then $(\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2 \models r$, and by (b) $(\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2 = \mathcal{I} \circ \mathcal{U}$. Therefore $\mathcal{U}_1 \circ \mathcal{U}_2$ is a weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$. □

Lemma 16 In the conditions of Lemma 15, if $\mathcal{U}_1$ is a repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is a repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is a repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.

Proof
By Lemma 15, $\mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$. Suppose $\mathcal{U}$ is not a repair; then there is $\mathcal{U}' \subsetneq \mathcal{U}$ such that $\mathcal{U}'$ is also a weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.

Take $\mathcal{U}'_1 = \mathcal{U}' \cap \mathcal{U}_1$ and $\mathcal{U}'_2 = \mathcal{U}' \cap \mathcal{U}_2$; by Lemma 12, $\mathcal{U}'_1$ is a weak repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}'_2$ is a weak repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$. But at least one of the inclusions $\mathcal{U}'_1 \subseteq \mathcal{U}_1$ and $\mathcal{U}'_2 \subseteq \mathcal{U}_2$ must be strict, contradicting the hypothesis that $\mathcal{U}_1$ and $\mathcal{U}_2$ are both repairs. Therefore $\mathcal{U}$ is a repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$. □

The condition that $\mathcal{U}_1$ and $\mathcal{U}_2$ be repairs is sufficient but not necessary, as illustrated by Example 7 – unlike in Lemma 6 earlier.

Lemma 17 If $\mathcal{U}_1$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is founded w.r.t. $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.

Proof
Take $\alpha \in \mathcal{U}_1$. Since $\mathcal{U}_1$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \rangle$, there is an instance $r\theta$ of a rule $r \in \eta_1$ such that $\alpha \in \mathit{head}\,(r\theta)$ and $\mathcal{I} \circ \mathcal{U}_1 \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$. By (b) from the proof of Lemma 15, also $\mathcal{I} \circ \mathcal{U} \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$, whence α is founded w.r.t. $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$ and $\mathcal{U}$.

Take $\alpha \in \mathcal{U}_2$. Since $\mathcal{U}_2$ is founded w.r.t. $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, there is an instance of a rule $r \in \eta_2$ such that $(\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2 \models L$ for every $L \in \mathit{body}\,(r\theta) \setminus \{\mathit{lit}(\alpha)^D\}$, and since $(\mathcal{I} \circ \mathcal{U}_1) \circ \mathcal{U}_2 = \mathcal{I} \circ \mathcal{U}$ this implies that α is founded w.r.t. $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$ and $\mathcal{U}$.

Therefore $\mathcal{U}$ is founded w.r.t. $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$. □

Corollary 4 If $\mathcal{U}_1$ is a founded (weak) repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is a founded (weak) repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is a founded (weak) repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.

Proof
Consequence of Lemmas 15, 16 and 17. □

Lemma 18 In the conditions of Lemma 15, if $\mathcal{U}_1$ is a justified weak repair for $\langle \mathcal{I}, \eta_1 \rangle$ and $\mathcal{U}_2$ is a justified weak repair for $\langle \mathcal{I} \circ \mathcal{U}_1, \eta_2 \rangle$, then $\mathcal{U}$ is a justified weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$.

Proof
Define $\mathcal{N} = \mathsf{ne}\,(\mathcal{I} \circ \mathcal{U}_1, \mathcal{I} \circ \mathcal{U}_1 \circ \mathcal{U}_2)$ as in the proof of Lemma 14, and note the following properties.

(a) $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) \subseteq ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1) \subseteq ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) \cup \mathcal{U}_2$, as in Lemma 10.
(b) $\mathcal{I} \circ \mathcal{U}_1 \models L$ iff $\mathcal{I} \circ \mathcal{U} \models L$ for every literal $L \in body\,(r\theta)$ with $r \in \eta_1$, as in Lemma 12.
(c) $\mathcal{N} = \mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$, as in Lemma 14.

To see that $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for $\langle \mathcal{I}, \eta \rangle$, let $r \in \eta_1 \cup \eta_2$ and θ be such that $nup(r\theta) \subseteq lit(\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$. There are two cases to consider.

- If $r \in \eta_1$, then $nup(r\theta) \subseteq lit(\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1))$ by (a) and (b), and since $\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ is closed for η_1 this implies that $head\,(r\theta) \cap (\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)) \neq \emptyset$, whence also $head\,(r\theta) \cap (\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$ by $\mathcal{U}_1 \subseteq \mathcal{U}$ and (a).
- If $r \in \eta_2$, then by (c) $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) = \mathcal{U}_2 \cup \mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) = \mathcal{U}_2 \cup \mathcal{N}$; then $nup(r\theta) \subseteq lit(\mathcal{U}_2 \cup \mathcal{N})$, whence $head\,(r\theta) \cap (\mathcal{U}_2 \cup \mathcal{N}) \neq \emptyset$ because $\mathcal{U}_2 \cup \mathcal{N}$ is closed for η_2, and the latter condition can be rewritten as $head\,(r\theta) \cap (\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$.

In either case $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for r, whence $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for $\eta_1 \cup \eta_2$.

For minimality, let $\mathcal{U}' \subseteq \mathcal{U}$ be such that $\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for $\eta_1 \cup \eta_2$, and take $\mathcal{U}'_i = \mathcal{U}' \cap \mathcal{U}_i$ for $i = 1, 2$. Then $\mathcal{U}'_1 = \mathcal{U}_1$ and $\mathcal{U}'_2 = \mathcal{U}_2$:

- Let $r \in \eta_1$ and θ be such that $nup(r\theta) \subseteq lit(\mathcal{U}'_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1))$. Since $\mathcal{U}'_1 \subseteq \mathcal{U}'$, from (a) and the fact that $nup(r\theta) \cap lit(\mathcal{U}_2) = \emptyset$ (because $\eta_1 \prec \eta_2$) it follows that $nup(r\theta) \subseteq lit(\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$, whence $head\,(r\theta) \cap (\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$. By (a) and the fact that $head\,(r\theta) \cap \mathcal{U}_2 = \emptyset$, also $head\,(r\theta) \cap (\mathcal{U}'_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)) \neq \emptyset$. Therefore $\mathcal{U}'_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ contains $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ and is closed for η_1; since $\mathcal{U}_1 \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1)$ is the minimal set with this property and $\mathcal{U}_1 \cap ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_1) = \emptyset$, it follows that $\mathcal{U}'_1 = \mathcal{U}_1$.
- Let $r \in \eta_2$ and θ be such that $nup(r\theta) \subseteq lit(\mathcal{U}'_2 \cup \mathcal{N})$. From (c) and the equality $\mathcal{U}'_1 = \mathcal{U}_1$ established above, $nup(r\theta) \subseteq lit(\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}))$, whence $head\,(r\theta) \cap (\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \neq \emptyset$. Again by (c) and $\mathcal{U}'_1 = \mathcal{U}_1$ this amounts to saying that $head\,(r\theta) \cap (\mathcal{U}'_2 \cup \mathcal{N}) \neq \emptyset$. Therefore $\mathcal{U}'_2 \cup \mathcal{N}$ contains $\mathcal{N}$ and is closed for η_2, whence as before necessarily $\mathcal{U}'_2 = \mathcal{U}_2$.

Therefore $\mathcal{U}' = \mathcal{U}$, hence $\mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is the minimal set containing $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ and closed for $\eta_1 \cup \eta_2$. Thus $\mathcal{U}$ is a justified weak repair for $\langle \mathcal{I}, \eta_1 \cup \eta_2 \rangle$. □

Appendix B Proofs of results on repair trees

B.1 Proof of Theorem 8

Theorem 8
Suppose $\mathcal{U}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$ and let $\mathcal{J}$ be

$$\{nup(r\theta) \cap lit(ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) \mid ua(nup(r\theta)) \subseteq \mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})\}$$

The set $\mathcal{J}$ will be used to "guide" the construction of the branch of the tree leading to $\mathcal{U}$. Note that, by construction, $\mathcal{U} \cap \mathcal{J}^D = \emptyset$.

The proof amounts to showing that every node n satisfying $\mathcal{U}_n \subsetneq \mathcal{U}$ and $\mathcal{J}_n \subseteq \mathcal{J}$ has a descendant n' such that $\mathcal{U}_{n'} \subseteq \mathcal{U}$ and $\mathcal{J}_{n'} \subseteq \mathcal{J}$. Since $\emptyset \subseteq \mathcal{U}$ and $\emptyset \subseteq \mathcal{J}$, the root node satisfies this last condition.

Let n be a node of $T^j_{\langle \mathcal{I}, \eta \rangle}$ such that $\mathcal{U}_n \subsetneq \mathcal{U}$ and $\mathcal{J}_n \subseteq \mathcal{J}$. Since $\mathcal{U}_n$ cannot be a repair (it is a proper subset of $\mathcal{U}$), there is some instance $r\theta$ of a rule $r \in \eta$ such that $\mathcal{I} \circ \mathcal{U}_n \not\models r\theta$. Assume that $nup(r\theta) \setminus lit(\mathcal{U}) \not\subseteq \mathcal{J}$; then $\mathcal{U}_n \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for $r\theta$.

Observe that

$$ua(nup(r\theta)) = \underbrace{(ua(nup(r\theta)) \cap \mathcal{U}_n)}_{\subseteq \mathcal{U}_n \subseteq \mathcal{U}} \cup \underbrace{(ua(nup(r\theta)) \setminus \mathcal{U}_n)}_{\subseteq ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_n)}.$$

Since $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_n) = ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}) \cup (\mathcal{U} \setminus \mathcal{U}_n)^D$, actions in $ua(nup(r\theta))$ coming from the second of the above sets must be in either $ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ or $(\mathcal{U} \setminus \mathcal{U}_n)^D$. If the former were the case for every action, then $ua(nup(r\theta)) \subseteq \mathcal{U} \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$, whence $nup(r\theta) \cap lit(ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})) = nup(r\theta) \cap lit\,(ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U}_n)) \subseteq \mathcal{J}$, which contradicts the hypothesis. Therefore, there is an action in $ua(nup(r\theta)) \cap (\mathcal{U} \setminus \mathcal{U}_n)^D$, whence $ua(nup(r\theta)) \not\subseteq \mathcal{U}_n \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ and therefore $\mathcal{U}_n \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for $r\theta$.

But if $\mathcal{U}_n \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for every instance of a rule applicable in node n, then $\mathcal{U}_n \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed for η: $r\theta$ is not applicable in node n if either one of its non-updatable literals is contradicted in $\mathcal{I} \circ \mathcal{U}_n$ or $\mathcal{U}_n$ contains an action in its head. This cannot be the case, since $\mathcal{U}$ is a justified repair for $\langle \mathcal{I}, \eta \rangle$. Therefore, there is some $r\theta$ applicable in node n and such that $nup(r\theta) \setminus lit(\mathcal{U}) \subseteq \mathcal{J}$; yielding a node n' satisfying the required conditions.

Since $\mathcal{U}$ is finite, this construction must terminate at a leaf with update set $\mathcal{U}$. □

B.2 Proof of Theorem 9

We first state an auxiliary lemma.

Lemma 19 Let $\mathcal{I}$ be a database and η be a set of normal AICs. Then every consistent leaf of $T^j_{\langle \mathcal{I}, \eta \rangle}$ contains a founded weak repair.

Proof
Let n be a leaf in $T^j_{\langle \mathcal{I}, \eta \rangle}$. Since $\mathcal{U}_n$ is the label of a leaf $T_{\langle \mathcal{I}, \eta \rangle}$, by Theorem 7 it is a weak repair for $\langle \mathcal{I}, \eta \rangle$. To show that it is founded, let $\alpha \in \mathcal{U}_n$. The rule instance $r\theta$ labeling the edge from n_1 to n_2 in the step where α was introduced provides support for α. Indeed, $\alpha \in head\,(r\theta)$ by construction; furthermore, since every literal $L \neq lit(\alpha)$ in $body\,(r\theta)$ is not updatable (α is the only action in the head of $r\theta$), either $ua(L) \in \mathcal{U}_{n_1}$ or $ua(L) \in \mathcal{J}_{n_2}$. In the first case, $ua(L) \in \mathcal{U}$, since $\mathcal{U}_{n_1} \subseteq \mathcal{U}$; in the second case, $ua(L) \in \mathcal{J}_{n_1}$. Since n was not pruned, this means that $L \in \mathcal{I}$ and that $ua(L)^D \notin \mathcal{U}$. In either case, $\mathcal{I} \circ \mathcal{U} \models L$. Therefore $r\theta$ supports α in $\mathcal{U}$. □

Theorem 9

Lemma 19 establishes that every consistent leaf of the justified repair tree for $\langle \mathcal{I}, \eta \rangle$ is founded for $\langle \mathcal{I}, \eta \rangle$. Let $\mathcal{U}$ be the weak founded repair in a leaf and assume that $\mathcal{U}$ is not a justified repair for $\langle \mathcal{I}, \eta \rangle$; since founded weak repairs are always closed, this means that there is $\mathcal{U}' \subseteq \mathcal{U}$ such that $\mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$ is closed under η.

Let $\alpha \in \mathcal{U} \setminus \mathcal{U}'$. For every instance $r\theta$ of a rule $r \in \eta$ such that $\alpha \in head\,(r\theta)$ and $\mathcal{I} \circ (\mathcal{U} \setminus \{\alpha\})$ satisfies every literal in $body\,(r\theta)$ except for $lit(\alpha)^D$, necessarily $ua(nup(r\theta)) \not\subseteq \mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$: if $ua(nup(r\theta)) \subseteq \mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$, then there is an action $\beta \in head\,(r\theta)$ such that $\beta \in \mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$, which is absurd – β cannot be α (since $\alpha \notin \mathcal{U}'$ and $\alpha \notin ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$), but then $\mathcal{I} \circ (\mathcal{U} \setminus \{\alpha\}) \models lit(\beta)^D$. Since $\mathcal{U}$ is founded, at least one such instance $r\theta$ must exist. Therefore there exists $\beta \in ua(nup(r\theta))$ such that $\beta \notin \mathcal{U}' \cup ne\,(\mathcal{I}, \mathcal{I} \circ \mathcal{U})$, hence $\beta \in \mathcal{U} \setminus \mathcal{U}'$. In other words: for every $\alpha \in \mathcal{U} \setminus \mathcal{U}'$, if a rule supports α in $\mathcal{U}$, then α is not applicable in $\mathcal{U}'$.

Now consider the branch from the root to this particular leaf. The update set at the root is $\emptyset \subseteq \mathcal{U}'$. Each stage extends the update set with an action in either $\mathcal{U}'$ or $\mathcal{U} \setminus \mathcal{U}'$. In the first case, the invariant $\mathcal{U}_n \subseteq \mathcal{U}'$ is maintained; in the second case, an action α is introduced by means of a rule that does not support α in $\mathcal{U}$. But the proof of Lemma 19 shows that the rule introducing an action is always supported in leaves that are not pruned, which is a contradiction. Therefore, $\mathcal{U} \setminus \mathcal{U}' = \emptyset$, whence $\mathcal{U}$ is a justified weak repair. But if η is normal then justified weak repairs are necessarily repairs (Theorem 4 in (Caroprese and Truszczyński 2011)), so $\mathcal{U}$ is a justified repair. □

B.3 Proof of Theorem 10

Theorem 10

1. Under the hypothesis, for $i = 1, 2$ there exist branches $\emptyset = \mathcal{U}_i^0, \mathcal{U}_i^1, \ldots, \mathcal{U}_i^{n_i} = \mathcal{U}_i$ of $T^{wf}_{\langle \mathcal{I}, \eta_i \rangle}$ ending with a consistent leaf and where no rule of η_i is applicable.
   Then $\emptyset = \mathcal{U}_1^0, \mathcal{U}_1^1, \ldots, \mathcal{U}_1^{n_1} = \mathcal{U}_1 \cup \mathcal{U}_2^0, \mathcal{U}_1 \cup \mathcal{U}_2^1, \ldots, \mathcal{U}_1 \cup \mathcal{U}_2^{n_2}$ is a branch $T^{wf}_{\langle \mathcal{I}, \eta \rangle}$. For $i = 0, \ldots, n_1 - 1$ this is immediate, since $\eta_1 \subseteq \eta$; for $j = 0, \ldots, n_2 - 1$, simply observe that $\mathcal{I} \circ \left(\mathcal{U}_1 \cup \mathcal{U}_2^j\right)$ evaluates every instance of every rule in η_2 in the same way as $\mathcal{I} \circ \mathcal{U}_2^j$, since Lemma 5 applies. Therefore $\mathcal{U}_1 \cup \mathcal{U}_2^{j+1}$ is a descendant of $\mathcal{U}_1 \cup \mathcal{U}_2^j$ in $T^{wf}_{\langle \mathcal{I}, \eta \rangle}$. Furthermore, since $\mathcal{U}_1 \cup \mathcal{U}_2^j$ also evaluates every rule in η_1 in the same way as $\mathcal{U}_1$, it follows that $\mathcal{U}_1 \cup \mathcal{U}_2^{n_2} = \mathcal{U}$ is a weak repair for $\langle \mathcal{I}, \eta \rangle$ – consistency being a consequence of Lemma 6. If $\mathcal{U}_1$ and $\mathcal{U}_2$ are both repairs, them Lemma 6 ensures that $\mathcal{U}$ is also a well-founded repair.
2. Since $\mathcal{U}$ is a well-founded weak repair for $\langle \mathcal{I}, \eta \rangle$, there is a branch $\emptyset = \mathcal{U}^0, \mathcal{U}^1, \ldots, \mathcal{U}^n$ in $T^{wf}_{\langle \mathcal{I}, \eta \rangle}$ whose leaf is $\mathcal{U}$. Consider the sets $\mathcal{U}^i_{\eta_1} = \mathcal{U}^i \cap \mathcal{U}_1$. By Lemma 5 $\mathcal{U}^i_{\eta_1}$ always assigns the same semantics to every rule in η_1 as $\mathcal{U}^i$. Then one can show by induction that $\mathcal{U}^i_{\eta_1}$ is a label of some node in $T^{wf}_{\langle \mathcal{I}, \eta_1 \rangle}$. If $i = 0$, this is trivial: $\emptyset$ labels the root. Assume the result holds for i; there are two cases to consider. If the action added from $\mathcal{U}^i$ to $\mathcal{U}^{i+1}$ is in η_2, then $\mathcal{U}^{i+1}_{\eta_1} = \mathcal{U}^i_{\eta_1}$ also labes a node in this tree. Otherwise, this action occurs in the head of an instance of a rule from η_1 applicable in $\mathcal{I} \circ \mathcal{U}^i$; this rule is also applicable in $\mathcal{I} \circ \mathcal{U}^i_{\eta_i}$ by Lemma 5, so $\mathcal{U}^{i+1}_{\eta_1}$ is also a node in $T^{wf}_{\langle \mathcal{I}, \eta_1 \rangle}$.
   Therefore $T^{wf}_{\langle \mathcal{I}, \eta_1 \rangle}$ contains a branch ending in $\mathcal{U}_{\eta_1} = \mathcal{U}_1$. This is a leaf, since no rule in

η_1 is applicable. Therefore $\mathcal{U}_1$ is a well-founded weak repair for $\langle \mathcal{I}, \eta_1 \rangle$. If $\mathcal{U}$ is a repair, then so is $\mathcal{U}_1$ by Lemma 7. The case for η_2 is similar.
Finally, by definition of $T^{wf}_{\langle \mathcal{I}, \eta \rangle}$, all actions in $\mathcal{U}$ must occur in either $\mathcal{U}_1$ or $\mathcal{U}_2$, so $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.
□