Computing Repairs from Active Integrity Constraints

L. Cruz-Filipe[1,2,4] P. Engrácia[1,2] G. Gaspar[3,4] I. Nunes[3,4]

[1]Escola Superior Náutica Infante D. Henrique

[2]Centro de Matemática e Aplicações Fundamentais

[3]Faculty of Sciences, University of Lisbon

[4]Laboratory of Agent Modelling (LabMAg)

TASE 2013
July 3rd, 2013

The Problem

Databases typically pose conditions on data ("integrity constraints")...

The Problem

Databases typically pose conditions on data ("integrity constraints")...

...but because of errors sometimes these conditions no longer hold.

The Problem

Databases typically pose conditions on data ("integrity constraints")...

...but because of errors sometimes these conditions no longer hold.

Question

How can we repair a database that no longer satisfies its integrity constraints?

Outline

1. Integrity constraints

Outline

Outline

Outline

Outline

A database of family relations

Consider a database with information on family relations.

A database of family relations

Consider a database with information on family relations.

Fact

$$\mathrm{siblingOf}(\mathrm{John}, \mathrm{Mary})$$

A database of family relations

Consider a database with information on family relations.

Fact

$$\mathrm{siblingOf}(\mathrm{John}, \mathrm{Mary})$$

This database should also contain

Missing fact

$$\mathrm{siblingOf}(\mathrm{Mary}, \mathrm{John})$$

A database of family relations

Consider a database with information on family relations.

Fact

$$\mathrm{siblingOf}(\mathrm{John}, \mathrm{Mary})$$

This database should also contain

Missing fact

$$\mathrm{siblingOf}(\mathrm{Mary}, \mathrm{John})$$

Integrity constraint (simple)

$$\forall x \forall y.((\mathrm{siblingOf}(x, y) \land \neg \mathrm{siblingOf}(y, x)) \supset \bot)$$

Can fix the problem automatically?

Inconsistency

$$\text{siblingOf}(\text{John}, \text{Mary})$$

$$\forall x \forall y.((\text{siblingOf}(x, y) \wedge \neg \text{siblingOf}(y, x)) \supset \bot)$$

Can fix the problem automatically?

Inconsistency

$$\mathsf{siblingOf}(\mathsf{John}, \mathsf{Mary})$$

$$\forall x \forall y.((\mathsf{siblingOf}(x, y) \wedge \neg \mathsf{siblingOf}(y, x)) \supset \bot)$$

Solution

Add $\mathsf{siblingOf}(\mathsf{Mary}, \mathsf{John})$

Can fix the problem automatically?

Inconsistency

$$\text{siblingOf}(\text{John}, \text{Mary})$$

$$\forall x \forall y.((\text{siblingOf}(x, y) \wedge \neg\text{siblingOf}(y, x)) \supset \bot)$$

Solution

Add siblingOf(Mary, John)

…but is this so automatic?

Can fix the problem automatically?

Inconsistency

$$\mathrm{siblingOf}(\mathrm{John}, \mathrm{Mary})$$

$$\forall x \forall y.((\mathrm{siblingOf}(x, y) \wedge \neg \mathrm{siblingOf}(y, x)) \supset \bot)$$

Solution

Add siblingOf(Mary, John)

... but is this so automatic?

Another solution

Remove siblingOf(John, Mary)

Outline

Active integrity constraints

Motivation

Specify a constraint and propose possible solutions.

Active integrity constraints

Motivation

Specify a constraint and propose possible solutions.

Works both ways:

Active integrity constraints

Motivation

Specify a constraint and propose possible solutions.

Works both ways:

- may express preferences

Active integrity constraints

Motivation

Specify a constraint and propose possible solutions.

Works both ways:

- may express preferences
- may eliminate options

Family relations, revisited

Integrity constraint

$$\forall x \forall y.((\mathsf{siblingOf}(x, y) \wedge \neg \mathsf{siblingOf}(y, x)) \supset \perp)$$

Family relations, revisited

Integrity constraint

$$\forall x \forall y.((\mathrm{siblingOf}(x,y) \wedge \neg\mathrm{siblingOf}(y,x)) \supset \bot)$$

Active integrity constraint

$$\forall x \forall y.((\mathrm{siblingOf}(x,y) \wedge \neg\mathrm{siblingOf}(y,x)) \supset +\mathrm{siblingOf}(y,x))$$

Family relations, revisited

Integrity constraint

$$\forall x \forall y.((\mathsf{siblingOf}(x,y) \wedge \neg\mathsf{siblingOf}(y,x)) \supset \bot)$$

Active integrity constraint

$$\forall x \forall y.((\mathsf{siblingOf}(x,y) \wedge \neg\mathsf{siblingOf}(y,x)) \supset +\mathsf{siblingOf}(y,x))$$

Active integrity constraint

$$\forall x \forall y.((\mathsf{siblingOf}(x,y) \wedge \neg\mathsf{siblingOf}(y,x)) \supset -\mathsf{siblingOf}(x,y))$$

Family relations, revisited

Integrity constraint

$$\forall x \forall y.((\mathsf{siblingOf}(x, y) \wedge \neg \mathsf{siblingOf}(y, x)) \supset \bot)$$

Active integrity constraint

$$\forall x \forall y.((\mathsf{siblingOf}(x, y) \wedge \neg \mathsf{siblingOf}(y, x)) \supset \\ + \mathsf{siblingOf}(y, x) \mid - \mathsf{siblingOf}(x, y))$$

Active integrity constraints

Definition (Flesca2004)

An *Active integrity constraint* is a formula of the form

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

where $\{\alpha_1^D, \ldots, \alpha_k^D\} \subseteq \{L_1, \ldots, L_m\}$.

Active integrity constraints

Definition (Flesca2004)

An *Active integrity constraint* is a formula of the form

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

where $\{\alpha_1^D, \ldots, \alpha_k^D\} \subseteq \{L_1, \ldots, L_m\}$.

A valid AIC

$$\mathsf{siblingOf}(x, y) \wedge \neg\mathsf{siblingOf}(y, x) \supset +\mathsf{siblingOf}(y, x)$$

Active integrity constraints

Definition (Flesca2004)

An *Active integrity constraint* is a formula of the form

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

where $\{\alpha_1^D, \ldots, \alpha_k^D\} \subseteq \{L_1, \ldots, L_m\}$.

A valid AIC

$$\text{siblingOf}(x, y) \land \neg\text{siblingOf}(y, x) \supset +\text{siblingOf}(y, x)$$

An invalid AIC

$$\text{siblingOf}(x, y) \land \neg\text{siblingOf}(y, x) \supset -\text{siblingOf}(x, y) \mid +\text{Parent}(x)$$

Intuitive semantics of AICs

A generic AIC

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

Intuitive semantics of AICs

A generic AIC

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

- conjunction on the left (“body”)

Intuitive semantics of AICs

A generic AIC

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

- conjunction on the left ("body")
- disjunction on the right ("head")

Intuitive semantics of AICs

A generic AIC

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

- conjunction on the left ("body")
- disjunction on the right ("head")
- semantics of (normal) implication

Intuitive semantics of AICs

A generic AIC

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

- conjunction on the left ("body")
- disjunction on the right ("head")
- semantics of (normal) implication
- holds iff one of the L_is fails (but...)

Intuitive semantics of AICs

A generic AIC

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

- conjunction on the left ("body")
- disjunction on the right ("head")
- semantics of (normal) implication
- holds iff one of the L_is fails (but...)
- $\{\alpha_1^D, \ldots, \alpha_k^D\}$ are *updatable* literals

Repairs

Definition (Caroprese et al., 2006)

Let I be a database and η be a set of (A)ICs. A *weak repair* for I and η is a consistent set $\mathcal{U}$ of update actions such that:

Repairs

Definition (Caroprese et al., 2006)

Let I be a database and η be a set of (A)ICs. A *weak repair* for I and η is a consistent set $\mathcal{U}$ of update actions such that:

- $\mathcal{U}$ consists of essential actions only

Repairs

Definition (Caroprese et al., 2006)

Let I be a database and η be a set of (A)ICs. A *weak repair* for I and η is a consistent set $\mathcal{U}$ of update actions such that:

- $\mathcal{U}$ consists of essential actions only
- $\mathcal{I} \circ \mathcal{U} \models \eta$

Repairs

Definition (Caroprese et al., 2006)

Let I be a database and η be a set of (A)ICs. A *weak repair* for I and η is a consistent set $\mathcal{U}$ of update actions such that:

- $\mathcal{U}$ consists of essential actions only
- $\mathcal{I} \circ \mathcal{U} \models \eta$

(Beware of the notation.)

Repairs

Definition (Caroprese et al., 2006)

Let I be a database and η be a set of (A)ICs. A *weak repair* for I and η is a consistent set $\mathcal{U}$ of update actions such that:

- $\mathcal{U}$ consists of essential actions only
- $\mathcal{I} \circ \mathcal{U} \models \eta$

(Beware of the notation.)

Definition

A *repair* is a weak repair that is minimal w.r.t. inclusion.

Family relations, yet again

Inconsistency

$$\text{siblingOf}(\text{John}, \text{Mary})$$

$$\text{siblingOf}(x, y) \wedge \neg\text{siblingOf}(y, x) \supset +\text{siblingOf}(y, x)$$

Family relations, yet again

Inconsistency

$$\mathrm{siblingOf}(\mathrm{John}, \mathrm{Mary})$$

$$\mathrm{siblingOf}(x, y) \wedge \neg\mathrm{siblingOf}(y, x) \supset +\mathrm{siblingOf}(y, x)$$

A repair

$$+\mathrm{siblingOf}(\mathrm{Mary}, \mathrm{John})$$

Family relations, yet again

Inconsistency

$$\mathrm{siblingOf}(\mathrm{John}, \mathrm{Mary})$$

$$\mathrm{siblingOf}(x, y) \wedge \neg\mathrm{siblingOf}(y, x) \supset +\mathrm{siblingOf}(y, x)$$

Another repair

$$-\mathrm{siblingOf}(\mathrm{John}, \mathrm{Mary})$$

Family relations, yet again

Inconsistency

$$\text{siblingOf}(\text{John}, \text{Mary})$$

$$\text{siblingOf}(x, y) \wedge \neg\text{siblingOf}(y, x) \supset +\text{siblingOf}(y, x)$$

A weak repair

$$+\text{siblingOf}(\text{Mary}, \text{John}), +\text{Parent}(\text{John})$$

Family relations, yet again

Inconsistency

$$\text{siblingOf}(\text{John}, \text{Mary})$$

$$\text{siblingOf}(x, y) \wedge \neg\text{siblingOf}(y, x) \supset +\text{siblingOf}(y, x)$$

Not a weak repair

$$+\text{siblingOf}(\text{Mary}, \text{John}), -\text{siblingOf}(\text{John}, \text{Mary})$$

Where does the "active" part come in?

The notion of (weak) repair ignores the head of the AIC.

Where does the "active" part come in?

The notion of (weak) repair ignores the head of the AIC. We will come back to that later.

Where does the "active" part come in?

The notion of (weak) repair ignores the head of the AIC. We will come back to that later.

At this stage, how can we find repairs?

Where does the "active" part come in?

The notion of (weak) repair ignores the head of the AIC. We will come back to that later.

At this stage, how can we find repairs?

Algorithm

1. Choose a set $\mathcal{U}$ of update actions (based on $\mathcal{I}$)
2. Compute $\mathcal{I} \circ \mathcal{U}$
3. Check if all AICs in η hold

Where does the "active" part come in?

The notion of (weak) repair ignores the head of the AIC. We will come back to that later.

At this stage, how can we find repairs?

Algorithm

1. Choose a set $\mathcal{U}$ of update actions (based on $\mathcal{I}$)
2. Compute $\mathcal{I} \circ \mathcal{U}$
3. Check if all AICs in η hold

Each step can be done in polynomial time on $\mathcal{I}$ and η.

Where does the "active" part come in?

The notion of (weak) repair ignores the head of the AIC. We will come back to that later.

At this stage, how can we find repairs?

Algorithm

1. Choose a set $\mathcal{U}$ of update actions (based on $\mathcal{I}$)
2. Compute $\mathcal{I} \circ \mathcal{U}$
3. Check if all AICs in η hold

Each step can be done in polynomial time on $\mathcal{I}$ and η.

Finding weak repairs is NP-complete.

Finding weak repairs is NP-complete, our version

Tree algorithm

Build a tree (the *repair tree for* $\mathcal{I}$ *and* η) as follows.

Finding weak repairs is NP-complete, our version

Tree algorithm

Build a tree (the *repair tree for $\mathcal{I}$ and η*) as follows.

1. The root is $\emptyset$

Finding weak repairs is NP-complete, our version

Tree algorithm

Build a tree (the *repair tree for $\mathcal{I}$ and η*) as follows.

1. The root is $\emptyset$
2. For each consistent node n and rule r, if $\mathcal{I} \circ n \not\models r$, then:

Finding weak repairs is NP-complete, our version

Tree algorithm

Build a tree (the *repair tree for* $\mathcal{I}$ *and* η) as follows.

1. The root is $\emptyset$
2. For each consistent node n and rule r, if $\mathcal{I} \circ n \not\models r$, then:
   - for each action L in the body of r, $n' = n \cup \{L^D\}$ is a child of n

Finding weak repairs is NP-complete, our version

Tree algorithm

Build a tree (the *repair tree for $\mathcal{I}$ and η*) as follows.

1. The root is $\emptyset$
2. For each consistent node n and rule r, if $\mathcal{I} \circ n \not\models r$, then:
   - for each action L in the body of r, $n' = n \cup \{L^D\}$ is a child of n
   - the edge from n to n' is labeled by r

Finding weak repairs is NP-complete, our version

Tree algorithm

Build a tree (the *repair tree for $\mathcal{I}$ and η*) as follows.

1. The root is $\emptyset$
2. For each consistent node n and rule r, if $\mathcal{I} \circ n \not\models r$, then:
    - for each action L in the body of r, $n' = n \cup \{L^D\}$ is a child of n
    - the edge from n to n' is labeled by r

Lemma

- *This tree is finite.*

Finding weak repairs is NP-complete, our version

Tree algorithm

Build a tree (the *repair tree for $\mathcal{I}$ and η*) as follows.

1. The root is $\emptyset$
2. For each consistent node n and rule r, if $\mathcal{I} \circ n \not\models r$, then:
   - for each action L in the body of r, $n' = n \cup \{L^D\}$ is a child of n
   - the edge from n to n' is labeled by r

Lemma

- *This tree is finite.*
- *Every consistent leaf is a weak repair for I and η.*

Finding weak repairs is NP-complete, our version

Tree algorithm

Build a tree (the *repair tree for $\mathcal{I}$ and η*) as follows.

1. The root is $\emptyset$
2. For each consistent node n and rule r, if $\mathcal{I} \circ n \not\models r$, then:
   - for each action L in the body of r, $n' = n \cup \{L^D\}$ is a child of n
   - the edge from n to n' is labeled by r

Lemma

- *This tree is finite.*
- *Every consistent leaf is a weak repair for I and η.*
- *Every repair for I and η corresponds to a leaf in the tree.*

Optimizations

Optimizations

The repair tree can be significantly pruned, e.g. by identifying nodes that correspond to the same set of actions.

Optimizations

The repair tree can be significantly pruned, e.g. by identifying nodes that correspond to the same set of actions.

Inconsistent nodes may also be left out.

Outline

Declarative semantics

The notion of repair ignores the head of the AIC.

Declarative semantics

The notion of repair ignores the head of the AIC.

Caroprese et al., 2006/2011

- *founded* repairs take into account the actions in the head

Declarative semantics

The notion of repair ignores the head of the AIC.

Caroprese et al., 2006/2011

- *founded* repairs take into account the actions in the head
- *justified* repairs avoid justification circles

Declarative semantics

The notion of repair ignores the head of the AIC.

Caroprese et al., 2006/2011

- *founded* repairs take into account the actions in the head
- *justified* repairs avoid justification circles

Intuitively, if $\mathcal{U}$ is founded, then removing an action from $\mathcal{U}$ causes some AIC with that action in the head to be violated.

Computing founded repairs

Example

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, \text{not } b \supset -a \qquad r_3 : a, \text{not } c \supset +c$$
$$r_2 : \text{not } a, b \supset -b \qquad r_4 : b, \text{not } c \supset +c$$

Following the heads of the violated rules, we obtain:

Computing founded repairs

Example

Take $\mathcal{I} = \{a, b\}$ and

$$
\begin{aligned}
r_1 &: a, \text{not } b \supset -a & \qquad r_3 &: a, \text{not } c \supset +c \\
r_2 &: \text{not } a, b \supset -b & \qquad r_4 &: b, \text{not } c \supset +c
\end{aligned}
$$

Following the heads of the violated rules, we obtain:

$$
\begin{tikzcd}
 & \emptyset \arrow[dl, "r_3"'] \arrow[dr, "r_4"] & \\
\{+c\} & & \{+c\}
\end{tikzcd}
$$

Computing founded repairs

Example

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, \text{not } b \supset -a \qquad r_3 : a, \text{not } c \supset +c$$

$$r_2 : \text{not } a, b \supset -b \qquad r_4 : b, \text{not } c \supset +c$$

Following the heads of the violated rules, we obtain:

$$\begin{tikzcd} & \emptyset \arrow[dl, "r_3"'] \arrow[dr, "r_4"] & \\ \{+c\} & & \{+c\} \end{tikzcd}$$

$\{+c\}$ is a founded repair

Circularity of support I

Example

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, \text{not } b \supset -a \qquad r_3 : a, \text{not } c \supset +c$$
$$r_2 : \text{not } a, b \supset -b \qquad r_4 : b, \text{not } c \supset +c$$

But $\{-a, -b\}$ is also a founded repair.

Circularity of support I

Example

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, \text{not } b \supset -a \qquad r_3 : a, \text{not } c \supset +c$$
$$r_2 : \text{not } a, b \supset -b \qquad r_4 : b, \text{not } c \supset +c$$

But $\{-a, -b\}$ is also a founded repair.

The problem is that in $\{-a, -b\}$ we have a *circularity of support*.

Circularity of support I

Example

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, \text{not } b \supset -a \qquad r_3 : a, \text{not } c \supset +c$$
$$r_2 : \text{not } a, b \supset -b \qquad r_4 : b, \text{not } c \supset +c$$

But $\{-a, -b\}$ is also a founded repair.

The problem is that in $\{-a, -b\}$ we have a *circularity of support*.

It is a founded repair that is not justified.

Circularity of support II

Too restrictive?

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, b \supset -a \qquad r_2 : a, \text{not } b \supset -a \qquad r_3 : \text{not } a, b \supset -b$$

Circularity of support II

Too restrictive?

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, b \supset -a \qquad r_2 : a, \text{not } b \supset -a \qquad r_3 : \text{not } a, b \supset -b$$

Again $\{-a, -b\}$ is a founded repair that is not justified.

Justified repair trees I

We can adapt the repair tree as follows.

Justified repair trees I

We can adapt the repair tree as follows.

Modified tree algorithm

- Each node n is a pair of sets of repair actions $\mathcal{U}_n, \mathcal{J}_n$.

Justified repair trees I

We can adapt the repair tree as follows.

Modified tree algorithm

- Each node n is a pair of sets of repair actions $\mathcal{U}_n, \mathcal{J}_n$.
- The root of the tree is $\emptyset, \emptyset$.

Justified repair trees I

We can adapt the repair tree as follows.

Modified tree algorithm

- Each node n is a pair of sets of repair actions $\mathcal{U}_n, \mathcal{J}_n$.
- The root of the tree is $\emptyset, \emptyset$.
- The edges are labeled as before.

Justified repair trees I

We can adapt the repair tree as follows.

Modified tree algorithm

- Each node n is a pair of sets of repair actions $\mathcal{U}_n, \mathcal{J}_n$.
- The root of the tree is $\emptyset, \emptyset$.
- The edges are labeled as before.
- For each node n and rule r, if $\mathcal{I} \circ \mathcal{U}_n \not\models r$, then each α in the head of r yields a descendant n' of n with:

Justified repair trees I

We can adapt the repair tree as follows.

Modified tree algorithm

- Each node n is a pair of sets of repair actions $\mathcal{U}_n, \mathcal{J}_n$.
- The root of the tree is $\emptyset, \emptyset$.
- The edges are labeled as before.
- For each node n and rule r, if $\mathcal{I} \circ \mathcal{U}_n \not\models r$, then each α in the head of r yields a descendant n' of n with:
  - $\mathcal{U}_{n'} = \mathcal{U}_n \cup \{\alpha\}$ and $\mathcal{J}_{n'} = (\mathcal{J}_n \cup \{\mathrm{nup}(r)\}) \setminus \mathcal{U}_n$.

Justified repair trees I

We can adapt the repair tree as follows.

Modified tree algorithm

- Each node n is a pair of sets of repair actions $\mathcal{U}_n, \mathcal{J}_n$.
- The root of the tree is $\emptyset, \emptyset$.
- The edges are labeled as before.
- For each node n and rule r, if $\mathcal{I} \circ \mathcal{U}_n \not\models r$, then each α in the head of r yields a descendant n' of n with:
  - $\mathcal{U}_{n'} = \mathcal{U}_n \cup \{\alpha\}$ and $\mathcal{J}_{n'} = (\mathcal{J}_n \cup \{\mathrm{nup}(r)\}) \setminus \mathcal{U}_n$.
  - If $\mathcal{U}_{n'}$ is inconsistent or $\mathcal{U}_{n'} \cap (\mathcal{J}_{n'})^D \neq \emptyset$, then n' is removed.

Justified repair trees I

We can adapt the repair tree as follows.

Modified tree algorithm

- Each node n is a pair of sets of repair actions $\mathcal{U}_n, \mathcal{J}_n$.
- The root of the tree is $\emptyset, \emptyset$.
- The edges are labeled as before.
- For each node n and rule r, if $\mathcal{I} \circ \mathcal{U}_n \not\models r$, then each α in the head of r yields a descendant n' of n with:
  - $\mathcal{U}_{n'} = \mathcal{U}_n \cup \{\alpha\}$ and $\mathcal{J}_{n'} = (\mathcal{J}_n \cup \{\mathsf{nup}(r)\}) \setminus \mathcal{U}_n$.
  - If $\mathcal{U}_{n'}$ is inconsistent or $\mathcal{U}_{n'} \cap (\mathcal{J}_{n'})^D \neq \emptyset$, then n' is removed.

Motivation

Keep track of the non-updatable atoms in the rules that were used.

Justified repair trees II

Example

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, b \supset -a \qquad r_2 : a, \text{not } b \supset -a \qquad r_3 : \text{not } a, b \supset -b$$

Justified repair trees II

Example

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, b \supset -a \qquad r_2 : a, \text{not } b \supset -a \qquad r_3 : \text{not } a, b \supset -b$$

Corresponding justified repair tree:

Justified repair trees II

Example

Take $\mathcal{I} = \{a, b\}$ and

$$r_1 : a, b \supset -a \qquad r_2 : a, \text{not } b \supset -a \qquad r_3 : \text{not } a, b \supset -b$$

Corresponding justified repair tree:

$$\begin{array}{c} \emptyset, \emptyset \\ \big\downarrow r_1 \\ \{-a\}, \{+b\} \\ \big\downarrow r_3 \\ \{-a, \underline{-b}\}, \{\underline{+b}\} \\ \times \end{array}$$

Justified repair trees III

Theorem

Let $\mathcal{U}$ be a justified repair for $\mathcal{I}$ and η. Then there is a leaf n in the justified repair tree for $\mathcal{I}$ and η such that $\mathcal{U}_n = \mathcal{U}$.

Justified repair trees III

Theorem

Let $\mathcal{U}$ be a justified repair for $\mathcal{I}$ and η. Then there is a leaf n in the justified repair tree for $\mathcal{I}$ and η such that $\mathcal{U}_n = \mathcal{U}$.

Catch

Deciding whether there is a justified repair for $\mathcal{I}$ and η is Σ_P^2-complete.

Justified repair trees III

Theorem

Let $\mathcal{U}$ be a justified repair for $\mathcal{I}$ and η. Then there is a leaf n in the justified repair tree for $\mathcal{I}$ and η such that $\mathcal{U}_n = \mathcal{U}$.

Catch

Deciding whether there is a justified repair for $\mathcal{I}$ and η is Σ_P^2-complete. Therefore sometimes there must be nodes in the tree that do not correspond to justified (weak) repairs.

Justified repair trees III

Theorem

Let $\mathcal{U}$ be a justified repair for $\mathcal{I}$ and η. Then there is a leaf n in the justified repair tree for $\mathcal{I}$ and η such that $\mathcal{U}_n = \mathcal{U}$.

Catch

Deciding whether there is a justified repair for $\mathcal{I}$ and η is Σ_P^2-complete. Therefore sometimes there must be nodes in the tree that do not correspond to justified (weak) repairs.

Checking that a repair is justified is again NP-complete, so our algorithm is (asymptotically) optimal.

Normalized AICs

Definition

An AIC is *normalized* if its head only contains one action.

Normalized AICs

Definition

An AIC is *normalized* if its head only contains one action.

Lemma (Caroprese et al.)

If η is a set of normalized AICs, then existence of justified repairs for $\mathcal{I}$ and η is NP-complete.

Normalized AICs

Definition

An AIC is *normalized* if its head only contains one action.

Lemma (Caroprese et al.)

If η is a set of normalized AICs, then existence of justified repairs for $\mathcal{I}$ and η is NP-complete.

Theorem

If η is a set of normalized AICs, then every leaf of the justified repair tree for $\mathcal{I}$ and η corresponds to a justified repair for $\mathcal{I}$ and η.

Normalized AICs

Definition

An AIC is *normalized* if its head only contains one action.

Lemma (Caroprese et al.)

If η is a set of normalized AICs, then existence of justified repairs for $\mathcal{I}$ and η is NP-complete.

Theorem

If η is a set of normalized AICs, then every leaf of the justified repair tree for $\mathcal{I}$ and η corresponds to a justified repair for $\mathcal{I}$ and η.

We also believe that our algorithm is nicer than the one given by Caroprese et al.

Normalized AICs

Definition

An AIC is *normalized* if its head only contains one action.

Lemma (Caropress et al.)

If η is a set of normalized AICs, then existence of justified repairs for $\mathcal{I}$ and η is NP-complete.

Theorem

If η is a set of normalized AICs, then every leaf of the justified repair tree for $\mathcal{I}$ and η corresponds to a justified repair for $\mathcal{I}$ and η.

We also believe that our algorithm is nicer than the one given by Caropresе et al. It was developed following the intuitive semantics of AICs.

Outline

What we achieved. . .

What we achieved...

- Operational semantics for active integrity constraints

What we achieved. . .

- Operational semantics for active integrity constraints
- Tree algorithms for repairs and justified repairs

What we achieved. . .

- Operational semantics for active integrity constraints
- Tree algorithms for repairs and justified repairs
- More fine-grained distinction between founded and justified repairs

. . . and what we still hope to do

...and what we still hope to do

- Optimization of the tree algorithms, through parallelization

. . . and what we still hope to do

- Optimization of the tree algorithms, through parallelization
- Generalizations outside the database world

Thank you.