

Towards a Tailored Theory of Consistency Enforcement in Databases

Sebastian Link, Massey University, Palmerston North, New Zealand

1. Related Work

2. Axiomatic Program Semantics

3. The Theory of Greatest Consistent Specializations

4. Tailoring the Theory

5. Summary and Questions

1.1. Overview of Related Work in Logic Programming

- integrity constraints defined as normal clauses with empty head
- find translations of update request against program (insertion, deletion, update)
- translations are sets of updates implying the given request without violating any invariant
 - **Mayol/Teniente**^{FoIKS 2000} : computes all minimal translations (set inclusion), detailed comparison
 - **Wüthrich**^{ICDE 93} : does not compute all translations (some may not be minimal)

- **Console/Sapino/Theseider**^{Journal of Intelligent Information Systems 95} : class of invariants rather restricted
- **Lobo/Trajcevski**^{Journal of Applied Non-Classical Logics 97} : restricted classes, some translations do not imply request
- **Decker**^{Dynamics 97} : cannot handle all update requests
- GCS/MCE approach is not a run-time approach and is more ambitious than **Mayol/Teniente** and **Dekhtyar/Dikovsky/Dudakov/Spyratus**^{FoIKS 2000}
- program specifications of arbitrary complexity and not just sequences of assignments (update requests)
- maximal expansion and minimal translations are MCEs computed at run-time

1.2. Rule Triggering Systems

- consistency is tried to be achieved by event-condition-action rules
- rules are derived from invariants as in **Ceri/Fraternali/Paraboschi/Tanka**^{ACM TODS 94} or are assumed to be programmed by user as in **Gertz**^{RIDE 94}
- analysis of reasonable rule behaviour is missing in these approaches
- **Schewe/Thalheim**^{Acta Cybernetica 98} : reveals limitations with respect to preservation of effects due to update programs
 - non-repairable sequences of insertions and deletions can be found for every set of invariants (checking repairability equivalent to implication problem of constraint class)
 - triggering repairable program executions may still result in undoing some of the program's effects, but nevertheless change the final state

1.3. The Failure of Triggers—An Example

- consider relation schema:

$AUTHOR_SESSION = \{AName, ATitle, ATopic, RName\}$
 $RESPONDER_SESSION = \{RName, RTitle, RTopic\}$

- consider in addition the following constraints:

$ID \equiv RESPONDER_SESSION[RTopic] \subseteq AUTHOR_SESSION[ATopic]$
 $FD \equiv AName \rightarrow ATitle, ATopic, RName$
 $ED \equiv AUTHOR_SESSION[AName] \parallel RESPONDER_SESSION[RName]$

- the following relations define an instance on the schema

AS

AName	ATitle	ATopic	RName
Thalheim	Foundations of ER-Modeling	Design Theory	Schewe
Vianu	Databases and Finite Model Theory	Query Languages	Turull-Torres

RS

RName	RTitle	RTopic
Schewe	Limitations of Rule Triggering Systems	Design Theory
Turull-Torres	Homogeneity in Logics with Counting	Query Languages

- to enforce *ID* insertions into *RS* must be replaced by

$$\begin{aligned}
 &RS := RS \cup \{t\}; \\
 &\text{IF } t \mid_{\text{RTopic}} \notin AS[\text{ATopic}] \\
 &\text{THEN } AS := AS \cup \{(\text{?}, \text{?}, \text{?}, t \mid_{\text{RTopic}})\}
 \end{aligned}$$

- to enforce *ID* deletions from *AS* must be replaced by

$$\begin{aligned}
 &AS := AS - \{t\}; \\
 &\text{IF } t \mid_{\text{ATopic}} \in RS[\text{RTopic}] - AS[\text{ATopic}] \\
 &\text{THEN } RS := RS - \{t' : t' \mid_{\text{RTopic}} = t \mid_{\text{ATopic}}\} \\
 &\text{ENDIF}
 \end{aligned}$$

- to enforce *ED* insertions must be replaced by

$$\begin{aligned}
 &RS := RS \cup \{t\}; \\
 &AS := AS - \{t' : t' \mid_{\text{AName}} = t \mid_{\text{RName}}\}
 \end{aligned}$$

and

$$\begin{aligned}
 &AS := AS \cup \{t\}; \\
 &RS := RS - \{t' : t' \mid_{\text{RName}} = t \mid_{\text{AName}}\}
 \end{aligned}$$

- now execute

$\text{insert}_{RS}((\text{Thalheim, Foundations of ER-Modeling, Design Theory}))$

- in order to enforce ED we delete the tuple

$(\text{Thalheim, Foundations of ER-Modeling, Design Theory, Schewe})$

from AS

- then

$(\text{Schewe, Limitations of Rule Triggering Systems, Design Theory})$

and

$(\text{Thalheim, Foundations of ER-Modeling, Design Theory})$

must be deleted from RS to enforce ID

- the resulting instance became

AS

AName	ATitle	ATopic	RName
Vianu	Databases and Finite Model Theory	Query Languages	Turull-Torres

RS

RName	RTitle	RTopic
Turull-Torres	Homogeneity in Logics with Counting	Query Languages

- the insertion of a tuple in *RS* by the original operation is completely lost and the new effect is a deletion in *AS* and *RS*

2.1. Logic, State Concept and Specifications

- $\mathcal{L}_{\infty\omega}^\omega$
- **state space**: finite set X of variables with associated types $\#x$
- **state**: type-compatible assignment $x \mapsto \sigma(x) \in \#x$
- **static invariants** on X : formulae \mathcal{I} with $fr(\mathcal{I}) \subseteq X$
- **guarded commands on X** :
 - skip, fail, loop and $\mathbf{x}_{i_1} := \mathbf{t}_{i_1} \parallel \dots \parallel \mathbf{x}_{i_k} := \mathbf{t}_{i_k}$
 - $\mathbf{S}; \mathbf{T}$, $\mathbf{S} \square \mathbf{T}$, $\mathbf{S} \boxtimes \mathbf{T}$, $\mathcal{P} \rightarrow \mathbf{S}$, $@\mathbf{y} \bullet \mathbf{S}$ and $\mu\mathbf{S}.f(\mathbf{S})$

2.2. Predicate Transformers

- **relational semantics**:
states $\Sigma = \Sigma(X)$ and transitions $\Delta(S) \subseteq \Sigma \times (\Sigma \cup \{\infty\})$
- **wlp(S)(φ)** characterizes those initial states σ such that each terminating execution of S starting in σ results in a state τ satisfying φ
- **wp(S)(φ)** characterizes those initial states σ such that each execution of S starting in σ terminates and results in a state τ satisfying φ
- predicate transformers exists

2.3 Inversion

- **dual predicate transformers**: $w(l)p(S)^*(\varphi) \Leftrightarrow \neg w(l)p(S)(\neg\varphi)$
- **pairing**: $wp(S)(\varphi) \Leftrightarrow wlp(S)(\varphi) \wedge wp(S)(true)$
- **universal conjunctivity**: $wlp(S)(\bigwedge_{i \in I} \varphi_i) \Leftrightarrow \bigwedge_{i \in I} wlp(S)(\varphi_i)$
- **expressiveness**:

$$\Delta(S) = \{(\sigma, \infty) \models_{\sigma} wp(S)^*(false)\} \cup \{(\sigma, \tau) \models_{\sigma} wlp(S)^*(\varphi_{\tau})\}$$

2.4. Guarded Commands - Semantics

$$\begin{aligned}
w(l)p(\text{skip})(\varphi) &\Leftrightarrow \varphi, \quad w(l)p(\text{fail})(\varphi) \Leftrightarrow \text{true}, \\
w(l)p(\text{loop})(\varphi) &\Leftrightarrow \text{false}(\vee \text{true}), \\
w(l)p(x_{i_1} := t_{i_1} \parallel \dots \parallel x_{i_k} := t_{i_k})(\varphi) &\Leftrightarrow \{x_{i_1}/t_{i_1}, \dots, x_{i_k}/t_{i_k}\}.\varphi, \\
w(l)p(S; T)(\varphi) &\Leftrightarrow w(l)p(S)(w(l)p(T)(\varphi)), \\
w(l)p(\mathcal{P} \rightarrow S)(\varphi) &\Leftrightarrow \mathcal{P} \Rightarrow w(l)p(S)(\varphi), \\
w(l)p(S \square T)(\varphi) &\Leftrightarrow w(l)p(S)(\varphi) \wedge w(l)p(T)(\varphi), \\
w(l)p(S \boxtimes T)(\varphi) &\Leftrightarrow w(l)p(S)(\varphi) \wedge (wp(S)^*(\text{true}) \vee w(l)p(T)(\varphi)), \\
w(l)p(@y \bullet S)(\varphi) &\Leftrightarrow \forall y. w(l)p(S)(\varphi) \\
wlp(\mu S. f(S))(\varphi) &\Leftrightarrow \bigwedge_{\alpha \in \text{Ord}} wlp(f^\alpha(\text{loop}))(\varphi) \\
wp(\mu S. f(S))(\varphi) &\Leftrightarrow \bigvee_{\alpha \in \text{Ord}} wp(f^\alpha(\text{loop}))(\varphi)
\end{aligned}$$

3.1. Consistency and Greatest Consistent Specializations

- proof obligation for consistency: $\mathcal{I} \Rightarrow \mathbf{wlp}(\mathbf{S})(\mathcal{I})$
- S on X , T on Y with $X \subseteq Y$:
 $\mathbf{T} \sqsubseteq \mathbf{S}$ iff $w(l)p(S)(\varphi) \Rightarrow w(l)p(T)(\varphi)$ holds for all X -formulae φ
- \mathcal{I} on X , S on $Y \subseteq X$:
 $S_{\mathcal{I}}$ on X is **GCS** of S wrt \mathcal{I} iff $S_{\mathcal{I}} \sqsubseteq S$, $S_{\mathcal{I}}$ is \mathcal{I} -consistent and \mathcal{I} -consistent T on X with $T \sqsubseteq S$ satisfy $T \sqsubseteq S_{\mathcal{I}}$
- $S_{\mathcal{I}}$ **always exists** and is **uniquely determined** by S and \mathcal{I}

3.2. Commutativity, Compositionality and Effectivity

- **Commutativity:** $\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow (S_{\mathcal{I}_1})_{\mathcal{I}_2} \equiv \mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S_{\mathcal{I}_1 \wedge \mathcal{I}_2}$
- **Compositionality:**
 - within complex specification S , replace each basic command by its GCS wrt. \mathcal{I} and call result $S'_{\mathcal{I}}$
 - under mild restrictions $S_{\mathcal{I}} \sqsubseteq S'_{\mathcal{I}}$ (**upper bound theorem**) and $S_{\mathcal{I}}$ can be obtained by adding a precondition $\mathcal{P}(S, \mathcal{I})$ to $S'_{\mathcal{I}}$
- **Effectivity:** switch to \mathcal{L}_{ar}
 - $(S, \mathcal{I}) \mapsto S_{\mathcal{I}}$ becomes computable, if recursive guarded commands are restricted to bounded loops
 - effective computation possible in form $S_{\mathcal{I}} = @y_1 \bullet \dots \bullet @y_n \bullet T_{\mathcal{I}}$ with all preconditions $\mathcal{P}(T', \mathcal{I}, \cdot)$ being decidable

3.3. Example—Formal Specification

- state space $X = \{x_1, x_2\}$ with values being sets of pairs
- consider the following three X -constraints:

$$\mathcal{I}_1 \equiv \pi_1(x_1) \subseteq \pi_1(x_2)$$

$$\mathcal{I}_2 \equiv \forall x, y. x \in x_2 \wedge y \in x_2 \wedge \pi_2(x) = \pi_2(y) \Rightarrow \pi_1(x) = \pi_1(y)$$

$$\mathcal{I}_3 \equiv \pi_2(x_1) \cap \pi_2(x_2) = \emptyset$$

- consider database transition S on $\{x_1\}$ defined by

$$x_1 := x_1 \cup \{(a, b)\}$$

3.3. Example—Enforcement

- replace S by a branch of its GCS with respect to \mathcal{I}_1 and obtain

$$x_1 := x_1 \cup \{(a, b)\} ; (a \notin \pi_1(x_2) \rightarrow @c \bullet x_2 := x_2 \cup \{(a, c)\} \boxtimes skip)$$

- replace assignment to x_2 by deterministic GCS branch $c \notin \pi_2(x_2) \rightarrow x_2 := x_2 \cup \{(a, c)\}$ with respect to \mathcal{I}_2 , rearranging gives following GCS branch of S with respect to $\mathcal{I}_1 \wedge \mathcal{I}_2$:

$$x_1 := x_1 \cup \{(a, b)\} ; ((a \notin \pi_1(x_2) \rightarrow @c \bullet c \notin \pi_2(x_2) \rightarrow x_2 := x_2 \cup \{(a, c)\}) \\ \square a \in \pi_1(x_2) \rightarrow skip)$$

- replace herein assignment to x_1 in S_2 by deterministic GCS branch

$$x_1 := x_1 \cup \{(a, b)\} ; x_2 := x_2 - \{x \in x_2 \mid \pi_2(x) = b\}$$

and replace x_2 by deterministic GCS branch

$$x_2 := x_2 \cup \{(a, c)\} ; x_1 := x_1 - \{x \in x_1 \mid \pi_2(x) = c\}$$

and compute

$$\mathcal{P}(S_2, \mathcal{I}_3) \Leftrightarrow b \notin \pi_2(x_2) \wedge (a \notin \pi_1(x_2) \Rightarrow \forall c. (c \notin \pi_2(x_2) \Rightarrow c \notin \pi_2(x_1) \cup \{b\}))$$

- final result is

$$b \notin \pi_2(x_2) \rightarrow x_1 := x_1 \cup \{(a, b)\} ; ((a \notin \pi_1(x_2) \rightarrow @c \bullet \\ c \notin \pi_2(x_2) \wedge c \notin \pi_2(x_1) \rightarrow x_2 := x_2 \cup \{(a, c)\}) \square a \in \pi_1(x_2) \rightarrow skip)$$

3.4. Major Problems with the Specialization Order

- specialization order \sqsubseteq is too **coarse**, i.e., the set of program specifications within such a chain is too small
- suppose we have $T \sqsubseteq S$ with S on X and T on Y with $X \subseteq Y$
- the GCS approach is **too liberal on $Y - X$**
- the GCS approach is **too restrictive on X**
- maximal consistent effect preservers are intended to overcome both weaknesses

4.1. Example—A different Policy

- GCS branch of inclusion constraint \mathcal{I}_1 was chosen that insertion into x_1 was followed by insertion into x_2
- alternative restricts insertions by adding a precondition

$$a \in \pi_1(x_2) \rightarrow x_1 := x_1 \cup \{(a, b)\} \boxtimes skip$$

and enforces consistency with respect to \mathcal{I}_1

- there is nothing to do to enforce \mathcal{I}_2
- replace $b \notin \pi_2(x_2) \rightarrow x_1 := x_1 \cup \{(a, b)\} \boxtimes skip$ for the assignment $x_1 := x_1 \cup \{(a, b)\}$ to enforce \mathcal{I}_3

$$a \in \pi_1(x_2) \wedge b \notin \pi_2(x_2) \rightarrow x_1 := x_1 \cup \{(a, b)\} \boxtimes skip$$

4.2. Characterizing the Specialization Order

- **transition constraint** on X : formulae \mathcal{J} with $fr(\mathcal{J}) \subseteq X \cup X'$
- each \mathcal{J} gives rise to specification **$S(\mathcal{J})$**

$$S(\mathcal{J}) = (@x'_1, \dots, x'_n \bullet \mathcal{J} \rightarrow x_1 := x'_1 \parallel \dots \parallel x_n := x'_n) \square loop$$

- an X -transition constraint \mathcal{J} is called **δ -constraint** for S on X iff $\{\mathbf{x}'/\mathbf{x}\}.wlp(S')(\mathcal{J})$ holds
- T on X , S on $Y \subseteq X$:
 $T \sqsubseteq S$ iff each δ -constraint \mathcal{J} for S with $fr(\mathcal{J}) \subseteq Y \cup Y'$ is δ -constraint for T and $wp(S)(true) \Rightarrow wp(T)(true)$

4.3. Compatibility and Lowness of δ -constraints

- build GCS of some $S(\mathcal{J})$ and exclude δ -constraints from \mathcal{J} that never lead to \mathcal{I} -consistent state
- consider $S'(\mathcal{J}) = @\mathbf{x}' \bullet \mathcal{J} \rightarrow \mathbf{x} := \mathbf{x}'$, building $S'(\mathcal{J})_{\mathcal{I}}$ shouldn't increase partiality
- \mathcal{J} is **compatible** wrt. \mathcal{I} iff $wp(\mathcal{I} \rightarrow S'(\mathcal{J})_{\mathcal{I}})(false) \Rightarrow wp(S'(\mathcal{J}))(false)$ holds
- S δ -constraint \mathcal{J} is **low** wrt. \mathcal{I} iff \mathcal{I} -compatible and there is no strictly stronger \mathcal{I} -compatible δ -constraint for S

4.4. Definition of Maximal Consistent Effect Preservers

- \mathcal{I} on X , \mathcal{J} is \mathcal{I} -compatible δ -constraint for S : $S^{\mathcal{I},\mathcal{J}}$ on X is called **consistent effect preserver** (CE) of S wrt. \mathcal{I} and \mathcal{J} if and only if
 - (i) \mathcal{J} is a δ -constraint for $S^{\mathcal{I},\mathcal{J}}$
 - (ii) $wp(S)(false) \Rightarrow wp(S^{\mathcal{I},\mathcal{J}})(false)$
 - (iii) $S^{\mathcal{I},\mathcal{J}}$ is \mathcal{I} -consistent
 - (iv) any other T with these properties satisfies $T \sqsubseteq S^{\mathcal{I},\mathcal{J}}$.
- if \mathcal{J} is low for S wrt. \mathcal{I} , then $S^{\mathcal{I},\mathcal{J}}$ is called **MCE**

4.5. Normal Form and Commutativity

- the MCE $S^{\mathcal{I}, \mathcal{J}}$ wrt \mathcal{I} and \mathcal{J} is $wp(S)^*(true) \rightarrow S(\mathcal{J})_{\mathcal{I}}$
- $S^{\mathcal{I}, \mathcal{J}}$ **always exists** and is **uniquely determined** by S , \mathcal{I} and \mathcal{J}
- $\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow (S^{\mathcal{I}_1, \mathcal{J}_1})_{\mathcal{I}_2, \mathcal{J}_2}$ and $\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S^{\mathcal{I}_1 \wedge \mathcal{I}_2, \mathcal{J}_{12}}$ differ
- S on Y , $\mathcal{I}_1, \mathcal{I}_2$ on X with $Y \subseteq X$, \mathcal{J} low δ -constraint for S with respect to $\mathcal{I}_1 \wedge \mathcal{I}_2$: $\mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow (S^{\mathcal{I}_1, \mathcal{J}})_{\mathcal{I}_2} \equiv \mathcal{I}_1 \wedge \mathcal{I}_2 \rightarrow S^{\mathcal{I}_1 \wedge \mathcal{I}_2, \mathcal{J}}$

5.1. Summary

- ambitious approach to Consistency Enforcement is well-founded by GCS
- operational specialization is simple model for effect preservation
- characterizing \sqsubseteq by the preservation of δ -constraints leads to a new approach to Consistency Enforcement based on MCEs
- close relationship between MCEs and GCSs
- adequate version of Commutativity Result in MCE framework

5.2. Questions

- Which kind of logics allow an axiomatic program semantics to be defined? Are there tight connections to Dynamic Logics?
- How do GCS for assignments with respect to various dependency classes look like?
- Do compositionality and effective computation carry over to MCE's?
- Is there any better order?