Acta Cybernetica 13 (1998) 277-304.

Limitations of Rule Triggering Systems for Integrity Maintenance in the Context of Transition Specifications

Klaus-Dieter Schewe * Bernhard Thalheim †

Abstract

Integrity Maintenance is considered one of the major application fields of rule triggering systems (RTSs). In the case of a given integrity constraint being violated by a database transition these systems trigger repairing actions. Then it is necessary to guarantee the termination of the RTS, its determinacy and the consistency of final states. Transition specifications provide some kind of dynamic semantics requiring certasin effects on database states to occur. In the context of transition specifications integrity maintenance has to cope with the additional problem of effect preservation.

Limitations of RTSs with respect to this extended problems are investigated. It will be shown that for any set of constraints there exist non-repairable transitions, which depend on the closure of the constraint set. This implies that integrity maintenance by RTSs is only possible, if the constraint implication problem is decidable. Even if unrepairable transitions are excluded, this does not prevent the RTS to produce undesired behaviour.

Analyzing the behaviour of RTSs leads to the definition of *critical paths* in associated rule hypergraphs and the requirement of such paths being absent. It will be shown that this requirement can be satisfied if the underlying set of constraints is *stratified*, but this notion turns out to be too strong to be also necessary. A sufficient and necessary condition for the absence of critical paths is obtained, if sets of constraints are required to be *locally stratified*.

Keywords: active databases, integrity maintenance, transition specifications

1 Introduction

Active databases (ADBs) aim at extending relational (or object oriented) DBMS by *rule triggering systems* (RTSs), i.e. by sets of rules which on a given *event* and in

*Technical University of Clausthal, Computer Science Institute, Erzstr. 1, 38678 Clausthal-Zellerfeld, Germany, e-mail: schewe@informatik.tu-clausthal.de

†Cottbus Technical University, Computer Science Institute, Karl-Marx-Str. 17, 03044 Cottbus, Germany, e-mail: thalheim@informatik.tu-cottbus.de

the case of a *condition* being satisfied trigger *actions* on the database (ECA-rules). Events can be external events, time conditions or internal events resulting from operations on the database. Conditions are usually given by boolean queries that have to be evaluated against the database. The action part consists of a sequence of basic operations to insert, delete or update tuples (or objects respectively) in the database.

The current research on ADBs (see e.g. [4]) is dominated by implementational aspects, whilst foundations of RTSs are seldom approached. The work in [2, 3, 5, 10, 11] and partly in [4] considers the problem to enforce database integrity by the use of RTSs. The results concern the generation of repairing ECA-rules and partly the analysis of the resulting RTS. This analysis concentrates on the *termination* of the rule system, the independence of the final database state from the chosen selection order of the rules (*determinacy*) and on *consistency*.

These properties are orthogonal to one another. Therefore, it is reasonable to investigate the consistency requirement alone and to neglect for the moment the other two requirements.

Besides the specification of static semantics by the use of integrity constraints there is an increasing interest in integrating dynamic aspects [1]. In general, not all transitions between consistent states – with respect to the given set of integrity constraints – will be allowed, e.g. allowing transitions from any consistent state to the empty database state are not very useful. As a consequence, given a specification of transitions, the consistency requirement for RTSs occurs to be too weak. In a sense to be made precise in the sequel the final state resulting from an execution of the RTS should be "close" to its inconsistent starting state.

Note that this formulation already assumes that there is a final state and that this state is unique, i.e. the termination and determinacy properties are already tacitly assumed. This formulation could be weakened for non-determinate RTSs requiring the "closeness" for one or all of the possible final states. It could also be weakened for non-terminating RTSs requiring the "closeness" only, if a final state exists. This underlines again the orthogonality of the three basic requirements for integrity maintenance.

In the presence of transition specifications an inconsistent starting state for the RTS always occurs as the result of a specified transition. Therefore, one reasonable additional requirement is to preserve the effect of the transition at hand. In this case not only the static semantics expressed by the integrity constraints is sacrificed, i.e. only consistent states will be reached, but also the dynamic semantics expressed by transitions. This means that the allowed state pairs will always imply the effects of prespecified transitions.

There may be other equally reasonable requirements how to handle transition specifications. E.g., we may want only to undo some or even all effects of a transition or only preserve as many effects as possible depending on some measure on effects.

In this paper we analyze limitations of the rule triggering approach for integrity maintenance under the additional requirement to preserve the effects of transitions. For a given set of constraints in implicational normal form we first investigate the existence of non-repairable transitions. These are determined by the closure of

the constraint set. It turns out that the decidability of the constraint implication problem is necessary for integrity maintenance by RTSs.

Next we analyze, how to obtain RTSs that definitely repair constraint violations by a (repairable) transition without invalidating its intended effect. Given an RTS we first associate with it a *rule hypergraph* which corresponds to the possible sequences of triggered rules. Next we define *critical trigger paths* in these hypergraphs that correspond to the propagation of conditions. Indeed it can be shown that the existence of a single critical trigger path makes the RTS work incorrectly for at least one transition.

Finally, we analyze constraint sets in order to detect, whether it is possible to define an RTS of repairing actions such that the critical trigger paths in its associated hypergraph can only invalidate unrepairable transitions. For this we first introduce *stratified* constraint sets that satisfy this condition. Since the converse is not true, we finally weaken the concept to *locally stratified* constraint sets which gives a necessary and sufficient conditions for the RTS to work correctly.

2 Non-Repairable Transitions

In the following we consider the relational datamodel with *integrity constraints* given by formulae in *implicative normal form*

$$\mathcal{I} \equiv p_1(\vec{x}_1) \wedge \ldots \wedge p_n(\vec{x}_n) \Rightarrow q_1(\vec{y}_1) \vee \ldots \vee q_m(\vec{y}_m) \ , \qquad (1)$$

with predicate symbols p_i, q_j, which correspond either to a relation of the schema or are comparison predicates $(=, \neq, \leq, <)$. Variables on the left hand side are assumed to be universally quantified, those occurring only on the right hand side are assumed to be existentially quantified (with all $\forall$-quantifiers preceding all $\exists$-predicates).

Moreover, we assume that there is at least one relation symbol on the left hand side of each such $\mathcal{I}$. Moreover, $\mathcal{I}$ should contain at least two relation symbols. The first restriction guarantees the empty database to be consistent, i.e. it satisfies all constraints $\mathcal{I}$, and the second one just states that there is no explicit constraint which requires a relation p to be always empty. We may always write $\mathcal{I}$ in clausal form.

For the ECA-*rules* we use the notation ON ⟨event⟩ IF ⟨condition⟩ DO ⟨action⟩ with ⟨event⟩ corresponding to an internal event, i.e. an insert- or delete-operation. ⟨condition⟩ is a formula to be evaluated against the actual database state, written as a negation $\neg\mathcal{I}$ for a constraint $\mathcal{I}$ in implicative normal form (1). ⟨action⟩ is a sequence of basic insert- or delete-operations to be triggered, i.e. to be executed if the event occurred and the condition is satisfied.

In this paper, the assumed execution model for ECA-rules relies on a deferred modus, i.e. the system RTS of rules is started after finishing a transition. Furthermore, we do not assume any order of the rules. Instead of this, the execution model relies on demonic non-determinism, i.e. if the events of several rules $r_1, \ldots, r_n$ oc-

cur and their conditions evaluate to *true*, any of these r_i may be executed unless it is undefined.

Given a single constraint $\mathcal{I}$ in implicative normal form (1) we already get minimum requirements for repairing rules. If a relation symbol p occurs on the left hand side (right hand side) of (1), then each insert- (delete-)operation on p may violate (1), hence give rise to event-parts. The corresponding condition-part is simply $\neg\mathcal{I}$. However, for the action-part there are still several alternatives.

We call a system of ECA-rules *complete* iff for all these cases of events and conditions there exists at least one repairing rule, i.e. whenever the rule is selectable in some database state, the execution of the action part will establish $\mathcal{I}$ as a post-condition. However, we exclude those rules, which simply invalidate the event. For transitions we simply consider sequences of insert- and delete-operations.

Let us first demonstrate the insufficiency of a naive RTS approach by a simple example. In "real" applications the situation of Example 1 will not occur in such an obvious way, but there are always implied and in general not detectable constraints leading to analogous problems as shown in [7].

EXAMPLE 1 Take two unary relations p and q and the constraints $\mathcal{I}_1 \equiv p(x) \Rightarrow q(x)$ and $\mathcal{I}_2 \equiv p(x) \wedge q(x) \Rightarrow \mathit{false}$. This implies p to be always empty, hence insertions into p should be abolished. Then we obtain the following repairing rules:

$$\begin{aligned}
R_1 &: \text{ON insert}_p(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO insert}_q(x)\\
R_2 &: \text{ON delete}_q(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO delete}_p(x)\\
R_3 &: \text{ON insert}_p(x) \text{ IF } \neg\mathcal{I}_2 \text{ DO delete}_q(x)\\
R_4 &: \text{ON insert}_q(x) \text{ IF } \neg\mathcal{I}_2 \text{ DO delete}_p(x)
\end{aligned}$$

If we try to execute a transition $\text{insert}_p(a)$ on a database state satisfying $q(a)$, then we successively trigger the rules R_3 and R_2 with the effect of only deleting a in q. This contradicts the original intention of the transition. □

In order to analyze the unintended behaviour in Example 1 consider a set Σ of constraints in implicational normal form. Let Σ^* denote the (semantic) *closure*, i.e. $\Sigma^* = \{\mathcal{I} \mid \Sigma \models \mathcal{I}\}$. Now let $\mathcal{I} \in \Sigma^*$ be non-trivial, i.e. it does not hold in all database states. Write $\mathcal{I}$ in implicational normal form

$$\mathcal{I} \equiv p_1(\vec{x}_1) \wedge \ldots \wedge p_n(\vec{x}_n) \Rightarrow q_1(\vec{y}_1) \vee \ldots \vee q_m(\vec{y}_m)$$

and let $p_{i_1}, \ldots, p_{i_k}$ and $q_{j_1}, \ldots, p_{j_\ell}$ denote the relation symbols on the left and right hand sides of $\mathcal{I}$ respectively. We may define a transition T by

$$\text{delete}_{q_{j_1}}(\vec{y}_{j_1}); \ldots; \text{delete}_{q_{j_\ell}}(\vec{y}_{j_\ell}); \text{insert}_{p_{i_1}}(\vec{x}_{i_1}); \ldots; \text{insert}_{p_{i_k}}(\vec{x}_{i_k}) \ .$$

If we start T with values for the $\vec{x}_i$ and $\vec{y}_j$ such that the additional conditions on the left hand side of $\mathcal{I}$ are satisfied, whilst the additional conditions on the right hand side are not, T will always reach a database state satisfying $\neg\mathcal{I}$. This effect of

T is intentional and hence the only reasonable approach to integrity maintenance in this case is to disallow such transitions.

More formally, the *effect* of a transition T in a state σ is given by the strongest (with respect to $\Rightarrow$) formula $\mathbf{Eff}_\sigma(T) = \psi$ such that $\models_\sigma wp(T)(\psi)$ holds. Here $wp(T)(\psi)$ denotes the weakest precondition of ψ under the transition T, i.e. starting T in initial state σ will reach a final state τ satisfying ψ.

Since we only consider sequences of insertions and deletions, $\mathbf{Eff}_\sigma(T)$ can always be written as a conjunction of literals, i.e. in negated implicational normal form, with the positive literals corresponding to insertions and the negative ones to deletions. In addition, we may consider the effect of a sequence $T; RTS$, where T is a transition and RTS a system of rules. We say that RTS *invalidates the effect* of T iff $\not\models \mathbf{Eff}_\sigma(T) \wedge \mathbf{Eff}_\sigma(T; RTS)$ holds for some state σ.

Then it is justified to call a transition T *repairable* with respect to the constraint set Σ iff $\neg\mathbf{Eff}_\sigma(T) \notin \Sigma^*$ holds for at least one state σ. Then a complete terminating system RTS of ECA-rules always invalidates the effect of a non-repairable transition T. Hence the problem is to detect (and exclude) non-repairable transitions. In order to decide whether a given transition T is repairable or not, we must be able to decide, whether $\neg\mathbf{Eff}_\sigma(T)$ is in the closure Σ^*. Hence the implication problem for constraints must be decidable.

Proposition 1 Let Σ be a set of constraints. The problem to decide, whether a transition T is repairable with respect to Σ is equivalent to the constraint implication problem for Σ, i.e. the problem to decide, whether a given constraint $\mathcal{I}$ is a member of Σ^* or not. □

Proposition 1 defines the first limit on integrity maintenance by rule triggering systems. In the following sections we shall concentrate on repairable transitions.

Note that our treatment ignores the termination problem. Non-terminating transitions have to be excluded as well, but this problem is independent from the repairability problem, since non-termination of RTSs occurs as an orthogonal problem.

3 Critical Paths

Let us ask, whether we can always find a complete set of repair rules for all repairable transitions. For this we introduce the notions of *associated hypergraphs* and *critical trigger paths*.

Definition 2 Let $\mathcal{S} = \{p_1, \ldots, p_n\}$ be a relational database schema and $RTS = \{R_1, \ldots, R_m\}$ a system of ECA-rules on $\mathcal{S}$. Then the *associated rule hypergraph* (V, E) is constructed as follows:

- V is the disjoint union of $\mathcal{S}$ and RTS. We then talk of $\mathcal{S}$-vertices and RTS-vertices respectively.

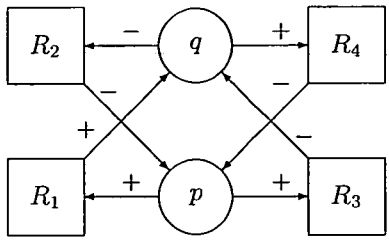

Figure 1: Associated Rule Hypergraph

- If $R \in RTS$ has event-part Ev on $p \in \mathcal{S}$ and actions on $p_1, \ldots, p_k$, then we have a hyperedge from p to $\{R\}$ labelled by $+$ or $-$ depending on Ev being an insert or delete, and a hyperedge from $\{R\}$ to $\{p_1, \ldots, p_k\}$ analogously labelled by k values $+$ or $-$. □

Figure 1 shows the associated rule hypergraph of Example 1 in which case we have a simple graph.

Definition 2 ignores the condition part of the rules. These come into play if we consider critical trigger paths in associated hypergraphs. These are defined in several steps starting from paths in the associated hypergraph which correspond to possible sequences of ECA-rules with respect only to their event- and action-parts. Secondly we attach formulae to the $\mathcal{S}$-vertices in the path in such a way that pre- and postconditions of the involved rules are expressed. Then we talk of *trigger paths*.

A maximal trigger path with contradicting initial and final condition will then be called *critical*. Then imagine a transition with an effect implied by the initial formula, i.e. that there is an initial state such that running the transition in this state results in a state which satisfies the initial condition of the trigger path. If we execute this transition followed by the rule triggering system along the critical trigger path will then turn the effect of the transition into its opposite. This means that the RTS invalidates the effect of at least one transition.

Definition 3 Let $G = (V, E)$ be the rule hypergraph associated with a system RTS of rules. A *trigger path* in G is a sequence $v_0, e_1, v'_1, e'_1, \ldots, e'_\ell, v_\ell$ of vertices and hyperedges with the following conditions:

- $v_i \in \mathcal{S}$ holds for all $i = 0, \ldots, \ell$,
- $v'_i \in RTS$ holds for all $i = 1, \ldots, \ell$,
- e_i is a hyperedge from v_{i-1} to v'_i and
- e'_i is a hyperedge from v'_i to V_i with $v_i \in V_i$ and the same label as e_{i+1}.

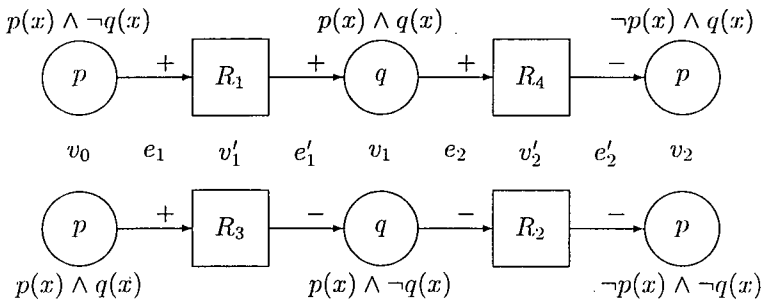

Figure 2: Critical Trigger Paths

We call ℓ the *length* of the trigger path.

In addition we associate with each vertex $v_i \in \mathcal{S}$ $(i = 0, \ldots, \ell)$ a formula φ_i in negated implication normal form such that $\models \varphi_i \Rightarrow cond(v'_{i+1})$ holds for the condition part $cond(v'_{i+1})$ of rule $v'_{i+1} \in RTS$ and $\models \varphi_i \Rightarrow wp(A_{i+1})(\varphi_{i+1})$ holds for the action-part A_{i+1} of rule v'_{i+1} $(i = 0, \ldots, \ell - 1)$. Furthermore, there is no $e_{\ell+1} \in E$ from v_ℓ to $v'_{\ell+1}$ with the same label as e'_ℓ such that $\models \varphi_\ell \Rightarrow cond(v'_{\ell+1})$ holds.

Then a trigger path is *critical* iff $\models \neg(\varphi_0 \wedge \varphi_\ell)$ holds. Such a critical trigger path is called *admissible* iff there is a consistent state σ and a repairable transition T such that $\mathbf{Eff}_\sigma(T) \Leftrightarrow \varphi_0$ holds. □

Critical trigger paths for the associated rule hypergraph in Figure 1 are sketched in Figure 2. Note that in this case both critical trigger paths are not admissible. If a critical trigger path is not admissible, then only a non-repairable transition can be invalidated by running the rules in the trigger path. Since we exclude non-repairable transitions, we only have to consider admissible trigger paths. After these remarks we are able to prove our first result.

Proposition 4 Let RTS be a complete set of rules associated with a set Σ of constraints and let $G = (V, E)$ be the associated rule hypergraph. Then G contains an admissible critical trigger path iff there exists a consistent database state σ and a repairable transition T such that executing T in σ and consecutively running RTS invalidates the effect of T without leaving the database unchanged.

Proof. Let us first assume that G contains an admissible critical trigger path. Let $\varphi_0, \ldots, \varphi_\ell$ denote the formulae associated with the $\mathcal{S}$-vertices in this trigger path.

Case 1. Assume that e_1 is labelled by $+$. Then φ_0 contains at least one positive literal $p(\vec{x})$. Let σ be a consistent state and T a repairable transition such that $\mathbf{Eff}_\sigma(T)$ is given by φ_0. We may assume that $\models_\sigma \neg p(\vec{x})$ holds and that the final

action in T in an insertion into p. If we start T in the initial state σ, then the resulting state satisfies φ_0.

T followed by the RTS may then result in a state τ satisfying φ_ℓ. Hence the effect of $T; RTS$ in σ is given by φ_ℓ. Since $\models \neg(\varphi_0 \wedge \varphi_\ell)$ holds by the definition of critical trigger paths, this implies that RTS invalidates the effect of T. Furthermore, τ is consistent with respect to all constraints in Σ, since RTS is complete and there is no hyperedge $e_{\ell+1}$ from v_ℓ to some $v'_{\ell+1} \in RTS$ with the same label as e'_ℓ such that $\models \varphi_\ell \Rightarrow cond(v'_{\ell+1})$ holds.

It remains to show $\tau \neq \sigma$. If this does not hold, we get $\models_\sigma \varphi_\ell$ and consequently there exists some ψ such that $\varphi_\ell \Leftrightarrow \neg p(\vec{x}) \wedge \psi$ and $\varphi_0 \Leftrightarrow p(\vec{x}) \wedge \psi$ hold. This implies $\ell > 1$, because otherwise the rule v'_1 would have the form ON $\text{insert}_p(\vec{x})$ IF $\neg\mathcal{I}$ DO $\text{delete}_p(\vec{x})$, which we excluded.

If $\ell > 1$ holds, there is at least one other literal $q(\vec{y})$ (or $\neg q(\vec{y})$) in φ_0 such that $\text{delete}_q(\vec{y})$ (or $\text{insert}_q(\vec{y})$ respectively) occurs in the action-part of v'_1. Then we may consider the admissible critical trigger path $v_1, e_2, \ldots, v_\ell$ of length $\ell - 1$ instead. Following the argumentation above, we may choose σ and T in such a way that $\models_\sigma \neg q(\vec{y})$ (or $\models_\sigma q(\vec{y})$ respectively) holds. This implies $\tau \neq \sigma$ as required.

Case 2. If e_1 is labelled by $-$, then φ_0 contains a literal $\neg p(\vec{x})$. Thus, we have to consider a transition T containing $delete_p(\vec{x})$ as its final action and a consistent state σ with $\models_\sigma p(\vec{x})$ and $\mathbf{Eff}_\sigma(T) \Leftrightarrow \varphi_0$. Then we may apply the same arguments as for case 1.

Conversely, assume that there is no admissible critical trigger path. Let T be a repairable transition and σ a database state which is consistent with respect to Σ. Now start T in σ and assume that the resulting state σ' is not consistent. Then consider a trigger path of finite length such that $\models_{\sigma'} \varphi_0$ holds. The consecutive execution of the rules in this trigger path will result in a state τ satisfying φ_ℓ. Thus, we have $\mathbf{Eff}_\sigma(T) \Leftrightarrow \varphi_0$ and $\mathbf{Eff}_\sigma(T; RTS) \Leftrightarrow \varphi_\ell$.

According to our assumption, the used trigger path cannot be critical, i.e. $\varphi_\ell \wedge \varphi_0$ is satisfiable. Hence RTS does not invalidate the effect of T. □

4 Stratified Constraint Sets

According to the result in Proposition 4 we may ask for constraint sets that allow to define complete RTSs which exclude admissible critical trigger paths in their associated hypergraphs. Let us start with a simple example.

EXAMPLE 2 Take again two unary relations p and q and the constraints $\mathcal{I}_1 \equiv p(x) \Rightarrow q(x)$ and $\mathcal{I}_2 \equiv q(x) \Rightarrow p(x)$ which implies p to be always equal to q. Then

we obtain the following repairing rules:

$$\begin{aligned} R_1 &: \text{ON insert}_p(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO insert}_q(x) \\ R_2 &: \text{ON delete}_q(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO delete}_p(x) \\ R_3 &: \text{ON insert}_q(x) \text{ IF } \neg\mathcal{I}_2 \text{ DO insert}_p(x) \\ R_4 &: \text{ON delete}_p(x) \text{ IF } \neg\mathcal{I}_2 \text{ DO delete}_q(x) \end{aligned}$$

In this case there are no admissible critical paths in the associated rule hypergraph. We omit further details. □

Let us now investigate the reason for the absence of admissible critical trigger paths in Example 2. This leads us to the notion of a *stratified* set of constraints.

The motivation behind this is as follows: In Example 2 insertions (deletions) on a relation p only trigger insertions (deletions) on q and vice versa. This should be sufficient for not invalidating a once established effect. The corresponding constraints can therefore be grouped together.

Definition 5 Let Σ be a set of constraints in implicative normal form (1) on a schema $\mathcal{S}$. The Σ is called *stratified* iff we have a partition $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_n$ with pairwise disjoint constraint sets Σ_i called *strata* such that the following conditions are satisfied:

(i) If L is a literal on the left hand side (right hand side) of some constraint $\mathcal{I} \in \Sigma_i$, then all constraints $\mathcal{J} \in \Sigma$ containing a literal L' on the right hand side (left hand side) such that L and L' are unifiable also lie in stratum Σ_i.

(ii) All constraints $\mathcal{I}$, $\mathcal{J}$ containing unifiable literals L and L' either on the left or the right hand side must lie in different strata Σ_i and Σ_j. □

Now we can prove in general that stratified constraint sets always give rise to RTSs without admissible critical trigger paths in the associated rule hypergraph.

Proposition 6 Let Σ be a stratified constraint set on a schema $\mathcal{S}$. Then there exists a complete RTS such that for any repairable transition T on $\mathcal{S}$ the RTS does not invalidate the effect of T.

Proof. Given a constraint $\mathcal{I}$ in implicative normal form (1), then each relation symbol p_i on the left hand side gives rise to rules

$$\begin{aligned} &\text{ON } \mathit{insert}_{p_i}(\vec{x}_i) \text{ IF } \neg\mathcal{I} \text{ DO } \mathit{insert}_{q_j}(\vec{y}_j) \quad , \\ &\text{ON } \mathit{insert}_{p_i}(\vec{x}_i) \text{ IF } \neg\mathcal{I} \text{ DO } \mathit{delete}_{p_j}(\vec{y}_j) \end{aligned}$$

with relation symbols q_j occurring on the right hand side and p_j $(j \neq i)$ on the left hand side of $\mathcal{I}$. Similarly, each predicate symbol q_j on the right hand side gives rise to rules

$$\begin{aligned} &\text{ON } \mathit{delete}_{q_j}(\vec{y}_j) \text{ IF } \neg\mathcal{I} \text{ DO } \mathit{insert}_{q_i}(\vec{y}_j) \quad (i \neq j) \ , \\ &\text{ON } \mathit{delete}_{q_j}(\vec{y}_j) \text{ IF } \neg\mathcal{I} \text{ DO } \mathit{delete}_{p_i}(\vec{y}_j) \end{aligned}$$

This defines a complete set RTS of rules. Now assume there exists a critical trigger path $v_0, e_1, v_1', e_1', \ldots, e_\ell', v_\ell$ in the associated rule hypergraph. Each RTS-vertex v_i' corresponds to a constraint $\mathcal{I}_i \in \Sigma$. Since e_i' and e_{i+1} are equally labelled corresponding to the action- or event-part respectively, the construction of the rules above implies $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$ to lie in the same stratum ($i = 0, \ldots, \ell - 1$).

However, the condition $\models \neg(\varphi_0 \wedge \varphi_\ell)$ implies that φ_0 contains a literal L, φ_ℓ its negation, hence the construction of rules implies $\mathcal{I}_1$ and $\mathcal{I}_\ell$ to lie in different strata. Hence, there are only critical trigger paths of length $\ell = 1$.

According to our construction of RTS this implies $\models \varphi_0 \Rightarrow \neg\mathcal{I}$ to hold for some $\mathcal{I} \in \Sigma$. Thus, $\neg\varphi_0 \in \Sigma^*$ holds. Due to the definition of admissible critical trigger paths and the definition of repairable transitions, we conclude that the trigger paths of length $\ell = 1$ cannot be admissible. Then the proposition follows from Proposition 4. □

Finally, we may ask for cases, where stratified constraint sets occur. Recall from [6] that a relational database schema $\mathcal{S}$ with constraint set Σ is in *Entity-Relationship normal form* (ERNF) – and hence is equivalent to an ER-schema – iff

- all inclusion constraints in Σ are key-based and non-redundant,
- there is no cycle of inclusion constraints in Σ,
- each relation schema $R \in \mathcal{S}$ is in BCNF with respect to the functional dependencies in Σ^* and
- there are only inclusion and functional dependencies in Σ^*.

If a relational database schema $\mathcal{S}$ with constraint set Σ is in ERNF, then it is easy to see that Σ is stratified.

Corollary 7 Let $\mathcal{S}$ be a database schema in ERNF with respect to the constraint set Σ. Then Σ is stratified. □

Hence, following the design approach of Mannila and Räihä in [6] – if this is sufficient for the application – leads to schemata without any problems concerning consistency enforcement by RTSs.

EXAMPLE 3 Let us look at the following constraints

$$\begin{aligned} \mathcal{I}_1 &: \quad p(x,y) \Rightarrow q(x,z) \qquad \text{and} \\ \mathcal{I}_2 &: \quad q(x,z) \wedge q(y,z) \Rightarrow x = y \end{aligned}$$

Then this set of constraints corresponds to the Entity-Relationship diagram [9] in Figure 3. Obviously, the constraint set is stratified. □

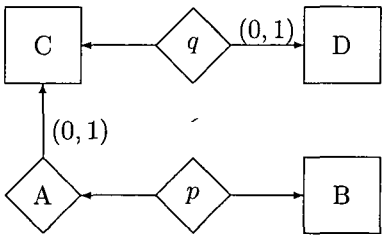

Figure 3: Entity-Relationship constraints

5 An Algorithm for Checking Stratification

Before we analyze the converse of Proposition 6 and present the weaker notion of locally stratified constraint sets, let us first concentrate on an algorithm for checking stratification and its complexity. For this we consider the set

$$BW \quad = \quad \{\top, \bot\} \cup (\mathbb{N} - \{0\}) \cup \{\{j_1, \ldots, j_n\} \mid n \geq 1, j_k \in \mathbb{N} - \{0\}\} \quad .$$

In the algorithm we successively add labels from BW to constraints. A label $i \in \mathbb{N}$ for a constraint $\mathcal{I}$ is used to indicate that $\mathcal{I}$ must lie in the stratum Σ_i. A label $\{j_1, \ldots, j_n\}$ indicates that $\mathcal{I}$ must not lie in Σ_{j_k} for $k = 1, \ldots, n$. $\bot$ represents no information and $\top$ an inconsistent assignment of stratum numbers.

For a more convenient terminology we call an element of BW *black*, if it is in $(\mathbb{N} - \{0\}) \cup \{\top\}$, otherwise *white*. Furthermore, we use a commutative, associative binary operation $\oplus$ on BW defined by

$$\begin{aligned}
x \oplus \bot &= x \quad , \\
x \oplus \top &= \top \quad , \\
i \oplus j &= \begin{cases} i & \text{if } i = j \\ \top & \text{otherwise} \end{cases} \quad , \\
\{j_1, \ldots, j_n\} \oplus \{k_1, \ldots, k_m\} &= \{j_1, \ldots, j_n\} \cup \{k_1, \ldots, k_m\} \quad \text{and} \\
i \oplus \{j_1, \ldots, j_n\} &= \begin{cases} \top & \text{if } i = j_k \text{ for some } k \in \{1, \ldots, n\} \\ i & \text{otherwise} \end{cases}
\end{aligned}$$

Algorithm 8 ((Stratification Check))

Input: a set $\Sigma = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ of constraints
in clausal form $\mathcal{I}_i = L_{i,1} \vee \ldots \vee L_{i,n_i}$

Output: a boolean value b

Method:

VAR *gather* : ARRAY $1 \ldots n$ OF BW ,

```
    mb, mb' : BW ;
BEGIN
    FOR i = 1 TO n DO
        gather(i) := ⊥
    ENDFOR ;
    b := true ;
    mb := 1 ;
    WHILE Σ ≠ ∅ DO
        CHOOSE i_0 ∈ {1,...,n} WITH I_{i_0} ∈ Σ AND gather(i_0) is maximal ;
        Σ := Σ − {I_{i_0}} ;
        IF gather(i_0) is white
        THEN gather(i_0) := mb ;
            mb := mb + 1
        ENDIF ;
        mb' := gather(i_0) ;
        FOR j = 1 TO n_{i_0} DO
            FOR ALL I_k ∈ Σ DO
                FOR ℓ = 1 TO n_{i_k} DO
                    IF L_{i_0,j} and L_{k,ℓ} are unifiable AND gather(i_0) ≠ ⊤
                    THEN gather(k) := gather(k) ⊕ {gather(i_0)}
                    ELSIF L_{i_0,j} and ~ L_{k,ℓ} are unifiable
                    THEN gather(k) := gather(k) ⊕ gather(i_0)
                    ENDIF
                ENDFOR
            ENDFOR
        ENDFOR
    ENDDO ;
    FOR i = 1 TO n DO
        IF gather(i) = ⊤
        THEN b := false
        ENDIF
    ENDFOR ;
    RETURN(b)
END
```

□

We have to check that the algorithm is correct. Then we analyze its time complexity. Before we do this let us first look at a simple example.

EXAMPLE 4 Consider the following constraints:

$$\begin{aligned}
\mathcal{I}_1 &= \neg p(x) \vee \neg q(x) \vee r(x) \vee s(x) \quad , \\
\mathcal{I}_2 &= \neg q(x) \vee r(x) \vee \neg t(x) \quad , \\
\mathcal{I}_3 &= p(x) \vee \neg r(x) \quad , \\
\mathcal{I}_4 &= \neg s(x) \vee t(x) \quad \text{and} \\
\mathcal{I}_5 &= q(x) \vee \neg t(x) \quad .
\end{aligned}$$

Table 1: Stratification Check

	$L_{1,1}$	$L_{1,2}$	$L_{1,3}$	$L_{1,4}$	$L_{3,2}$	$L_{3,1}$	$L_{2,1}$	$L_{2,2}$	$L_{2,3}$	$L_{4,1}$	$L_{4,2}$	$L_{5,2}$	$L_{5,1}$	*gather*
1	1	1	1	1										1
3	1		1		1	1								1
2		{1}	{1}		1		$\top$	$\top$	$\top$					$\top$
4				1					$\top$	$\top$	$\top$			$\top$
5		1					$\top$				$\top$	$\top$	$\top$	$\top$
		$\mathcal{I}_1$			$\mathcal{I}_3$			$\mathcal{I}_2$			$\mathcal{I}_4$		$\mathcal{I}_5$	$b = false$

Then consider Table 1.

Each row corresponds to a constraint $\mathcal{I}_i$ and lists the values added to $gather(i)$ during the excution of the algorithm. The chosen order of the constraints in the algorithm is $\mathcal{I}_1$, $\mathcal{I}_3$, $\mathcal{I}_2$, $\mathcal{I}_4$, $\mathcal{I}_5$. Then b will become $false$ and hence Σ is not stratifiable. □

Let us now address the correctness of Algorithm 8.

Proposition 9 Let Σ be a set of constraints. Then Σ is stratifiable iff Algorithm 8 applied to the input Σ computes the output $b = true$.

Proof. Let us first assume that Σ is stratified. Let $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_n$ be a decomposition into strata and assume that the Σ_i are taken minimal with the required properties. We use induction on n.

For $n = 1$ there are no unifiable literals L and L' in different constraints $\mathcal{I}, \mathcal{J} \in \Sigma$. Hence $gather(i)$ will become 1 for alle i and we obtain $b = true$.

For $n > 1$ we may assume without loss of generality that some constraint in Σ_1 will be chosen first. Then, due to our minimality assumption, we get $gather(i) = 1$ for all $\mathcal{I}_i \in \Sigma_1$, whereas $gather(j)$ will be white for all $\mathcal{I}_j \notin \Sigma_1$. Thus, all constraints in Σ_1 will be chosen first.

Since $gather(j)$ was white for $\mathcal{I}_j \notin \Sigma_1$ and $gather(i) = 1$ for $\mathcal{I}_i \in \Sigma_1$ before chosing the first constraint in $\Sigma_2 \cup \ldots \cup \Sigma_n$, we may apply the induction hypothesis to $\Sigma_2 \cup \ldots \cup \Sigma_n$, which gives $gather(j) \neq \top$ for all $\mathcal{I}_j \notin \sigma_1$. This implies $b = true$ as claimed in the proposition.

Conversely, assume that the algorithm produces the result $b = true$. Then we must have $gather(i) \in \mathbb{N} - \{0\}$. Define $\Sigma_k = \{\mathcal{I}_i \in \Sigma \mid gather(i) = k\}$. Assume that the partition $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_n$ does not satisfy the conditions for strata in Definition 5. Then there are two possible cases:

(i) There are literals L and L' in constraints $\mathcal{I}_i \in \Sigma_k$ and $\mathcal{I}_j \in \Sigma_\ell$ with $k \neq \ell$ such that L and $\sim L'$ are unifiable. Suppose that $\mathcal{I}_i$ is chosen first by the algorithm. Then k will be added to $gather(j)$, which gives $gather(j) = \top$ contradicting our assumption.

(ii) There are unifiable literals L and L' in constraints $\mathcal{I}_i, \mathcal{I}_j \in \Sigma_k$. If $\mathcal{I}_i$ is chosen first by the algorithm, $\{k\}$ will be added to $gather(j)$, which also gives $gather(j) = \top$ contradicting our assumption.

Thus $\Sigma_1 \cup \ldots \cup \Sigma_n$ is a partition into strata, which completes the proof. □

Proposition 10 Let Σ be a set of constraints in clausal form, $n = \#\Sigma$, k the maximal arity of predicate symbols occurring in constraints $\mathcal{I} \in \Sigma$ and let ℓ be the maximum number of literals in these constraints. Then the time complexity of Algorithm 8 for checking, whether Σ is stratified is in $O(k \cdot \ell^2 \cdot n^2)$.

Proof. The initialization and the final computation of b can both be done in $O(n)$ steps.

In the inner FOR-loop the test for unifiability can be done in $O(k)$ steps, since there are no function symbols. All other operations have a complexity in $O(1)$. Hence the inner FOR-loop has a total complexity in $O(k)$. This loop is executed $\ell' \cdot \ell''$ times, where ℓ' is the number of literals in the chosen constraint $\mathcal{I}_{i_0}$ and ℓ'' is the total number of literals in the remaining constraints. If $\mathcal{I}_{i_0}$ is the i'th literal chosen by the algorithm, this can be estimated by $\ell^2 \cdot (n - i)$.

Since each $\mathcal{I} \in \Sigma$ will be chosen by the algorithm, the outer WHILE-loop will be executed n times. This gives the total complexity in

$$O(n) + O(\ell^2 \cdot \sum_{i=1}^{n}(n-i)) \cdot O(k) + O(n) \quad = \quad O(k \cdot \ell^2 \cdot n^2)$$

as claimed in the proposition. □

It is easy to see that $n \cdot \ell$ can be replaced by the total number $u = \sum_{i=1}^{n} n_i$ of literals in Σ with $u < n \cdot \ell$. Thus, the time complexity of the stratification checking algorithm 8 is in $O(k \cdot u^2)$.

From Proposition 6 we know that active mechanisms can be effectively applied, if the constraint set is stratified. In particular, this holds for schemata in ERNF [6], which are equivalent to Entity-Relationship schemata. From Proposition 10 we know that a stratification check can be done efficiently.

6 Locally Stratified Constraint Sets

Unfortunately, the converse of Proposition 6 does not hold, as seen in the next example. The reason for this is that in the proof of Proposition 6 we considered all repairing rules for a given constraint, whereas the constraint set in Example 5 allows to select only a subset thus gaining the required result without loosing the completeness of the RTS.

EXAMPLE 5 Take three unary relations p and q and the constraints $\mathcal{I}_1 \equiv p(x) \wedge r(x) \Rightarrow q(x)$, $\mathcal{I}_2 \equiv q(x) \Rightarrow p(x)$ and $\mathcal{I}_3 \equiv p(x) \Rightarrow r(x)$. It is easy to see that this constraint set is not stratified.

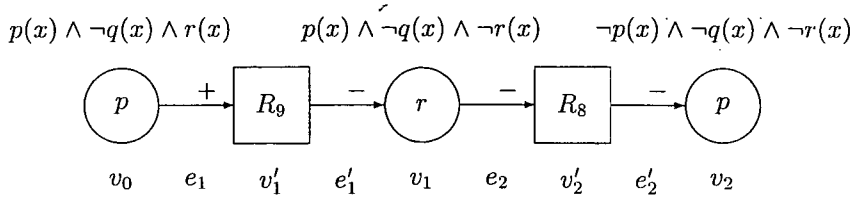

Figure 4: An Admissible Critical Trigger Path

However, we may consider the following system of ECA-rules:

$$
\begin{aligned}
R_1 &: \text{ON insert}_p(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO insert}_q(x)\\
R_2 &: \text{ON delete}_q(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO delete}_p(x)\\
R_3 &: \text{ON insert}_r(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO insert}_q(x)\\
R_4 &: \text{ON delete}_q(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO delete}_r(x)\\
R_5 &: \text{ON insert}_q(x) \text{ IF } \neg\mathcal{I}_2 \text{ DO insert}_p(x)\\
R_6 &: \text{ON delete}_p(x) \text{ IF } \neg\mathcal{I}_2 \text{ DO delete}_q(x)\\
R_7 &: \text{ON insert}_p(x) \text{ IF } \neg\mathcal{I}_3 \text{ DO insert}_r(x)\\
R_8 &: \text{ON delete}_r(x) \text{ IF } \neg\mathcal{I}_3 \text{ DO delete}_p(x)
\end{aligned}
$$

We dispense with showing that there are no admissible critical trigger paths in the associated rule hypergraph.

Note that the construction in the proof of Proposition 6 would result in two more rules corresponding to insertions:

$$
\begin{aligned}
R_9 &: \text{ON insert}_p(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO delete}_r(x)\\
R_{10} &: \text{ON insert}_r(x) \text{ IF } \neg\mathcal{I}_1 \text{ DO delete}_p(x)
\end{aligned}
$$

These give rise to admissible critical trigger paths. The one shown in Figure 4 allows to invalidate the effect of the repairable transition $\text{insert}_p(x)$. □

The constraint set in Example 5 is not stratified, but nevertheless the associated RTS does not invalidate the effect of repairable transitions. This shows that a constraint set need not be stratified to allow a reasonable rule behaviour. Indeed, replacing $\mathcal{I}_1$ in the example by $\mathcal{I}_1' \equiv p(x) \Rightarrow q(x)$ gives an equivalent constraint set, which is stratified. However, equivalence of constraint sets is undecidable in general. Therefore, we introduce the weaker notion of being *locally stratified*. In this case we shall construct RTSs which only contain a subset of the set of rules constructed in the proof of Proposition 6.

Definition 11 Let Σ be a set of constraints in implicative normal form on a schema $\mathcal{S}$.

A *labelled subsystem* consists of a subset $\Sigma' = \{\mathcal{I} \in \Sigma \mid \rho_L(\mathcal{I}) \text{ is defined } \}$ together with a set of clauses $\Sigma'' = \{\rho_L(\mathcal{I}) \mid \mathcal{I} \in \Sigma'\}$ and a literal L (the *label*) such that each constraint $\mathcal{I} \in \Sigma'$ can be written as the disjunction $\rho_L(\mathcal{I}) \vee \mathcal{I}'$ with $\models \mathcal{I}' \Rightarrow L$.

Here $\rho_L(\mathcal{I})$ is defined iff the negation $\sim L$ does not occur in $\mathcal{I}$ (written as a clause). Then $\rho_L(\mathcal{I})$ results from $\mathcal{I}$ by omission of the literal L if the result contains at least two literals. Otherwise $\rho_L(\mathcal{I})$ is simply $\mathcal{I}$. We call $\mathcal{I}'$ the *label part* and $\rho_L(\mathcal{I})$ the *label-free part* of the constraint $\mathcal{I}$. If L is understood from the context, we drop the subscript and write ρ instead of ρ_L.

A labelled subsystem (Σ', Σ'', L) is called *stratified* iff the set Σ'' is stratified in the sense of Definition 5 or locally stratified as defined below.

The constraint set Σ is called *locally stratified* iff $\Sigma = \Sigma'_1 \cup \ldots \cup \Sigma'_n$ with stratified labelled subsystems $(\Sigma'_i, \Sigma''_i, L_i)$ $(i = 1, \ldots, n)$ such that for each constraint $\mathcal{I} \in \Sigma'_i$ and each literal L occurring in its label part with respect to Σ_i there exists another j with $\mathcal{I} \in \Sigma'_j$ and L occurring in its label-free part of $\mathcal{I}$ with respect to Σ_j. □

EXAMPLE 6 For the constraint set Σ in Example 5 we obtain the partition into $\Sigma'_1 = \{\mathcal{I}_1, \mathcal{I}_3\}$ and $\Sigma'_2 = \{\mathcal{I}_1, \mathcal{I}_2\}$.

For the first of these we have the label $L_1 \equiv \neg p(x)$ and the label-free parts defined by $\rho_{L_1}(\mathcal{I}_1) \equiv q(x) \vee \neg r(x)$ and $\rho_{L_1}(\mathcal{I}_3) \equiv \mathcal{I}_3$.

For Σ'_2 we get the label $L_2 \equiv \neg r(x)$ and the label-free parts $\rho_{L_2}(\mathcal{I}_1) \equiv \neg p(x) \vee q(x)$ and $\rho_{L_2}(\mathcal{I}_2) \equiv \mathcal{I}_2$.

This shows that the constraint set in Example 5 is indeed locally stratified. □

Note that each stratified constraint set Σ is also locally stratified. In this case we define $depth(\Sigma) = 0$. If Σ is locally stratified by a partition $\Sigma = \Sigma'_1 \cup \ldots \cup \Sigma'_n$, we define $depth(\Sigma) = \max_{i=1}^{n} depth(\Sigma''_i) + 1$. We call $depth(\Sigma)$ the *depth* of the locally stratified constraint set Σ.

Finally, we can strengthen Proposition 6 now dealing with locally stratified constraint sets. This condition turns out to be sufficient and also necessary for the absence of admissible critical trigger paths.

Theorem 12 Let Σ be a constraint set on a schema $\mathcal{S}$. Then Σ is locally stratified iff there exists a complete RTS such that for any repairable transition T the RTS does not invalidate the effect of T.

Proof. First assume that Σ is locally stratified. Let the labelled subsystems in the partition be $(\Sigma'_i, \Sigma''_i, L_i)$ for $i = 1, \ldots, n$. We shall use induction on the depth of Σ. For $depth(\Sigma) = 0$ we are done by Proposition 6.

Let us now consider the case $depth(\Sigma) = 1$. As in the proof of Proposition 6 we construct an RTS for Σ. Since each Σ''_i is stratified in the sense of Definition 5, we first construct a rule system RTS'_i with respect to Σ''_i as in the proof of Proposition 6. The condition parts in these rules have the form $\neg\rho_{L_i}(\mathcal{I})$ for $\mathcal{I} \in \Sigma'_i$. Then let RTS_i result from RTS'_i by changing all condition parts replacing $\neg\rho_{L_i}(\mathcal{I})$ by $\neg\mathcal{I}$. Finally, take $RTS = \bigcup_{i=1}^{n} RTS_i$.

Due to the last property in the definition of locally stratified constraint sets in Definition 11 we conclude that RTS is complete.

Now consider a critical trigger path $v_0, e_1, v_1', e_1', v_1, \ldots, e_\ell', v_\ell$ in the rule hypergraph associated with RTS. Without loss of generality assume $v_1' \in RTS_1$. According to Proposition 4 we have to show that this trigger path is not admissible.

We use induction on the length ℓ of this critical trigger path. For $\ell = 1$ we may use the same argument as in the proof of Proposition 4. Therefore, assume $\ell > 1$ and take a state σ with $\models_\sigma \Sigma$ and a transition T with $\models \mathbf{Eff}_\sigma(T) \Leftrightarrow \varphi_0$. Then we have to show that T is not repairable.

Assume that T is repairable. Then there exists a state τ with $\models_\tau \Sigma$ such that $\neg\mathbf{Eff}_\tau(T) \notin \Sigma^*$. We shall derive a contradiction from this.

For this regard the critical trigger path $v_1, e_2, v_2', e_2', v_2, \ldots, e_\ell', v_\ell$ of length $\ell - 1$. By induction it is not admissible. If A_1 is the action in the rule v_1', we get $\models \mathbf{Eff}_\sigma(T; A_1) \Leftrightarrow \varphi_1$ and $T; A_1$ cannot be repairable. In particular, this implies $\neg\mathbf{Eff}_\tau(T; A_1) \in \Sigma^*$.

Since A_1 is a simple insertion or deletion, we get $\models \neg\mathbf{Eff}_\tau(T) \Leftrightarrow \varphi \wedge L$ and $\models \neg\mathbf{Eff}_\tau(T; A_1) \Leftrightarrow \varphi \wedge \sim L$ for some literal L and its negation $\sim L$. From this we conclude $\varphi \in \Sigma^*$ and $\sim L \in \Sigma^*$.

Then there must exist a resolution refutation for L from input Σ. Any literal L' (except L) in this refutation must be selected at least once for building the resolvent. Therefore, due to our construction of $\rho_{L_1}(\mathcal{I})$ we may cancel all clauses $\mathcal{I} \in \Sigma$ containing the literal $\sim L_1$ and simultaneously the literal L_1 in all clauses. Thus, there must also exist a resolution refutation for L from input Σ_1''.

On the other hand, each clause in Σ_1'' contains at least two literals. Therefore, any resolvent will also contain at least two literals unless we have some $\mathcal{I}_1 \in \Sigma_1''$ with literals L_1 and L_2 and another $\mathcal{I}_2 \in \Sigma_1''$ with literals L_1' and $\sim L_2'$ such that L_1, L_1' (and L_2, L_2' respectively) are unifiable.

This property, however, means that Σ_1'' is not stratified contradicting our assumptions. Hence T cannot be repairable and we are done.

Next let $depth(\Sigma) > 1$. We proceed analogously. By induction, since Σ_i'' is (locally) stratified, there exists a rule system RTS_i' for Σ_i'' with the required property. The condition parts in these rules have the form $\neg\rho_{L_i}(\mathcal{I})$ for $\mathcal{I} \in \Sigma_i'$. Then let RTS_i result from RTS_i' by changing all condition parts from $\neg\rho_{L_i}(\mathcal{I})$ to $\neg\mathcal{I}$. Finally, take $RTS = \bigcup_{i=1}^n RTS_i$.

Again due to the last property in the definition of locally stratified constraint sets (cf. Definition 11) RTS must be complete.

Now consider a critical trigger path $v_0, e_1, v_1', e_1', v_1, \ldots, e_\ell', v_\ell$ in the rule hypergraph associated with RTS. According to Proposition 4 we have to show that this trigger path is not admissible. Without loss of generality assume $v_1' \in RTS_1$. Then take a maximal k such that $v_1', \ldots, v_k' \in RTS_1$ holds. Then for $i = 0, \ldots, k$ we may write φ_i as a conjunction $\psi_i \wedge \mathcal{J}$ with $\models \psi_i \Rightarrow \neg\rho_{L_1}(\mathcal{I}_i)$ for some $\mathcal{I}_i \in \Sigma_1'$. Hence, if we replace v_i' by the corresponding rule in RTS_1', we obtain a critical trigger path for RTS_1'.

Now take a state σ with $\models_\sigma \Sigma$ and a transition T with $\models \mathbf{Eff}_\sigma(T) \Leftrightarrow \varphi_0$. We have to show that T is not repairable. Assume the contrary. Then there exists a state τ with $\models_\tau \Sigma$ and $\neg\mathbf{Eff}_\tau \notin \Sigma^*$.

Assume $\models_\sigma \neg L_1$. Since $\models_\sigma \Sigma$ holds and each constraint $\mathcal{I} \in \Sigma_1'$ can be written as a disjunction $\mathcal{I}' \vee \rho_{L_1}(\mathcal{I})$ with $\models \mathcal{I}' \Rightarrow L_1$, we conclude $\models_\sigma \Sigma_1''$.

Since $v_0, e_1, v_1', e_1', v_1, \ldots, e_k', v_k$ is a critical trigger path for RTS_1' and $\models \mathbf{Eff}_\sigma \Leftrightarrow \varphi_0$ holds, we may apply the induction hypothesis to Σ_1'' with $depth(\Sigma_1'') < depth(\Sigma)$. Therefore, T cannot be repairable, i.e. for any state ζ with $\models_\zeta \Sigma_1''$ we get $\neg\mathbf{Eff}_\zeta(T) \in (\Sigma_1'')^*$.

In particular, take $\zeta = \sigma$. Then $\neg\mathbf{Eff}_\sigma(T) \in (\Sigma_1'')^*$ implies $\models_\sigma \neg\rho_{L_1}(\mathcal{I})$ for some $\mathcal{I} \in \Sigma_1'$ and further $\not\models_\sigma \Sigma^*$ contradicting our assumption on σ. Thus, we must have $\models_\sigma L_1$.

Assume $\models_\tau \neg L_1$. Then we must have $\models_\tau \Sigma_1''$ and consequently $\neg\mathbf{Eff}_\tau(T) \in (\Sigma_1'')^*$. As above this implies $\models_\tau \neg\rho_{L_1}(\mathcal{I})$ for some $\mathcal{I} \in \Sigma_1'$ and hence $\not\models_\tau \Sigma^*$ contradicting our assumption on τ. Hence, we must have $\models_\tau L_1$.

Now let $\mathcal{I}_1 \in \Sigma$ correspond to the rule v_1'. Without loss of generality we may assume $\models \varphi_0 \Rightarrow \neg L_1$. Otherwise, we must have $\models \neg\mathcal{I}_1'$ and $\rho_{L_1}(\mathcal{I}_1)$ must not contain L_1. This implies L_1 to occur in $\mathcal{J}$, in which case we may change it to $\neg L_1$ without affecting the trigger path being critical.

Since $\models_\sigma L_1$ holds, T must involve an insertion (deletion) corresponding to a negative (positive) literal L_1. Hence, $\models \mathbf{Eff}_\tau(T) \Leftrightarrow \neg L_1 \wedge \neg\psi_\tau$ holds. Due to the independence of $\mathcal{J}$ from Σ_1'' we may choose τ in such a way that $\psi_\tau \in (\Sigma_1'')^*$ holds.

However, this implies $\models \neg\mathbf{Eff}_\tau(T) \Leftrightarrow L_1 \vee \psi_\tau \in \Sigma^*$ contradicting the non-repairability of T with respect to RTS_1'. This completes the sufficiency proof.

Conversely, assume that we are given a complete RTS for Σ which for any repairable transition T does not invalidate its effect. According to Proposition 4 this implies that all critical trigger paths in the associated rule hypergraph are not admissible. From this we have to construct a partition of Σ into stratified labelled subsystems.

First consider a single rule R corresponding to a constraint $\mathcal{I} \in \Sigma$. In particular, $\mathcal{I}$ is the condition part of this rule. Since RTS is complete, the event part of R gives rise to a negative (positive) literal L_{ev} in $\mathcal{I}$ for the case of an insertion (deletion). Similarly, an insertion (deletion) in the action part of R gives rise to a positive (negative) literal L_a in $\mathcal{I}$.

Let $\rho(\mathcal{I}) = L_{ev} \vee L_a$. If $\mathcal{I}$ contains $n \geq 1$ more literals $L_1, \ldots, L_n$, let $\rho_i(\mathcal{I}) = \rho(\mathcal{I}) \vee L_1 \vee \ldots \vee \underbrace{L_i}_{\text{omit}} \vee \ldots \vee L_n$. Then define $\Sigma_i'(R) = \{\mathcal{J} \in \Sigma \mid \rho_{L_i}(\mathcal{J}) \text{ is defined}\}$ and $\Sigma_i''(R) = \{\rho_{L_i}(\mathcal{J}) \mid \mathcal{J} \in \Sigma_i'(R)\}$. (For $\mathcal{I} \Leftrightarrow \rho(\mathcal{I})$ let $L_1 = L_{ev}$ and $L_2 = L_a$ and define $\Sigma_i'(R)$ and $\Sigma_i''(R)$ analogously.)

Define $\Sigma(R) = \{(\Sigma_i'(R), \Sigma_i''(R), L_i) \mid \Sigma_i''(R) \text{ is locally stratified}\}$, if this satisfies the last condition of Definition 11. Otherwise let $\Sigma(R) = \emptyset$. Then the elements of $\Sigma(R)$ define stratified labelled subsystems of Σ.

In order to check the local stratification for $\Sigma_i''(R)$ first check, whether it is stratified. If not, define for each literal L in $\rho_i(\mathcal{I})$ the sets $\Sigma_{i,L}'(R) = \{\mathcal{J} \in \Sigma_i''(R) \mid \rho_L(\mathcal{J}) \text{ is defined}\}$ and $\Sigma_{i,L}''(R) = \{\rho_L(\mathcal{J}) \mid \mathcal{J} \in \Sigma_{i,L}'(R)\}$. Consider

$\{(\Sigma'_{i,L}(R), \Sigma''_{i,L}(R), L) \mid \Sigma''_{i,L}(R)$ is locally stratified$\}$ and check the last condition of Definition 11.

Now take $LSS = \bigcup_{R \in RTS} \Sigma(R)$. If $\Sigma(R) \neq \emptyset$ holds for all $R \in RTS$, this satisfies the last condition of Definition 11 and we obtain a partition of Σ into stratified labelled subsystems. Then LSS is the required partition.

It remains to show $\Sigma(R) \neq \emptyset$ in the construction above. Assume $\Sigma(R) = \emptyset$. Then there exists a sequence $L_1, L_2, \ldots, L_k$ of literals in $\mathcal{I}$ and a sequence $(\Sigma'_1, \Sigma''_1, L_1), \ldots, (\Sigma'_k, \Sigma''_k, L_k)$ of non-stratified labelled subsystems such that $\Sigma'_{i+1} = \{\mathcal{J} \in \Sigma''_i \mid \rho_{L_{i+1}}(\mathcal{J})$ is defined$\}$ and Σ''_k contains two clauses $\mathcal{I}_1^k$ and $\mathcal{I}_2^k$ with literals L^1, $L^{1\prime}$ and L^2, $L^{2\prime}$ respectively such that L^1, L^2 and $L^{1\prime}$, $L^{2\prime}$ are unifiable.

$\mathcal{I}_1^k$ and $\mathcal{I}_2^k$ correspond to rules with respect to Σ''_k that define an admissible trigger path in the associated rule hypergraph. Since for $i = 1, 2$ $\mathcal{I}_1^k$ is $\neg\rho_{L_k}(\mathcal{I}_1^{k-1})$, we may successively replace these rules by rules corresponding to $\Sigma''_{k-1}, \ldots, \Sigma''_1, \Sigma$ and simultaneously replace the formulae φ_i^k by $\varphi_i^{k-1} = \varphi_i^k \wedge \neg L_k, \ldots, \varphi_i^0 = \varphi_i^1 \wedge \neg L_1$. The resulting trigger path is still critical and due to our construction it is also admissible with respect to Σ contradicting our assumption. This completes the necessity proof. □

EXAMPLE 7 Let us extend Example 3 and add a third constraint

$$\mathcal{I}_3 \quad \equiv \quad p(x,z) \wedge q(y,z) \Rightarrow \mathit{false} \quad .$$

In terms of the Entity-Relationship diagram in Figure 3 $\mathcal{I}_3$ corresponds to an exclusion constraint B||D. It is easy to see that the new set $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$ of constraints is not stratified.

In particular, any local stratification must contain a labelled subsystem with label $\neg q(x,z)$ with the reduced constraints $\mathcal{I}'_2 \equiv \neg q(y,z) \vee x = y$ and $\mathcal{I}'_3 \equiv \mathcal{I}_3$. However, $\neg q(x,z)$ cannot occur in the label-free part of some $\mathcal{I}'_2$, since this always defines the same labelled subsystem. Hence, the given constraint set is also not locally stratified. This shows that adding a single exclusion constraint to an Entity-relationship schema may already destroy a reasonable rule behaviour. □

7 Complexity of Local Stratification

Let us now look at the check, whether a given set of constraints Σ is locally stratified. In the second part of the proof of Theorem 12 we have seen that this check can be done by direct construction of the desired partition into maximal stratified labelled subsystems. The first part of that proof then indicates how to construct the corresponding RTS. In [8] we gave an explicit algorithm which also produces for each constraint the set of "reduced" constraints used in the RTS construction. However, the time complexity of that algorithm was beyond any practicality, since we could proof the following result.

Proposition 13 Let Σ be a set of constraints in clausal form, $n = \#\Sigma$, ℓ the maximum number of literals in constraints $\mathcal{I} \in \Sigma$ and k the maximal arity of

predicate symbols occurring in these constraints. Then checking Σ to be locally stratified can be done with a time complexity in $O(k \cdot \ell^2 \cdot n^{2n^2 \cdot \ell})$. □

We now want to show that this complexity result is not accidentally. For this we first show a technical lemma.

Lemma 14 Let Σ be a set of clauses containing only propositional atoms. Let L be a literal, such that $\sim L$ does not occur in any of the clauses in Σ. Assume $\Sigma = \Sigma_1 \cup \Sigma_2$ such that L does not occur in any of the clauses in Σ_1, but in all clauses of Σ_2. Moreover, Σ_2 contains only clauses with exactly two literals. If Σ_1 is locally stratified and Σ_2 is stratified, then Σ is locally stratified.

Proof. First assume that Σ_2 contains a single clause $C = L \vee L'$. If Σ_1 is not stratified, there is a partition $\Sigma_1 = \Sigma'_{11} \cup \cdots \cup \Sigma'_{1n}$ $(n > 2)$ with stratified labelled subsystems $(\Sigma'_{1i}, \Sigma''_{1i}, L_i)$. Then at most one L_k can be $\sim L'$ and we may define

$$\Sigma'_i \quad = \quad \begin{cases} \Sigma'_{1i} & \text{if } L_i = \sim L' \\ \Sigma'_{1i} \cup \{C\} & \text{otherwise} \end{cases}$$

By induction $(\Sigma'_i, \Sigma''_i, L_i)$ is a stratified labelled subsystem. Thus, $\Sigma = \Sigma'_1 \cup \cdots \cup \Sigma'_n$ defines the required partition.

Now assume that Σ_1 is stratified. Let $\Sigma_1 = \Sigma_{11} \cup \cdots \cup \Sigma_{1n}$ be a partition into pairwise disjoint strata. If Σ_1 contains just one clause C' with $\sim L'$ and no clause with L', we are done, since C may be added to the stratum of C'. Analogously, C may define its own stratum, if such a C' does not exist at all. Therefore, we are reduced to the following two cases:

- There is more than one clause in Σ_1 containing $\sim L'$ (and hence none containing L') and these clauses belong to different strata.
- There are exactly two clauses C_1 and C_2 containing $\sim L'$ or L' respectively. Inparticular, C_1 and C_2 belong to the same stratum Σ_{1i}.

In both cases we choose the literals $L_1 = \sim L$ and $L_2 = L'$ to define labelled subsystems

$$(\Sigma_1, \Sigma_1, L_1) \quad \text{and} \quad (\underbrace{\{C\} \cup \Sigma_1 - \{C'' \mid C'' \text{ contains } \sim L'\}}_{\Sigma'_2}, \Sigma''_2, L_2)$$

where Σ'_2 (and hence also Σ''_2) are stratified by the previous remarks.

In the first case choose C' containing $\sim L'$ and another literal L'' to define a labelled subsystem

$$(\underbrace{\Sigma'_1 \cup \{C\}}_{\Sigma'_3}, \Sigma''_3, L_3)$$

with $L_3 = \sim L''$, where Σ_1' is a proper subset of Σ_1 not containing C'. By induction Σ_3'' must be locally stratified.

In the second case choose $C_2 = L' \vee C_2'$, a literal L'' in C_2' and $L_3 = \sim L''$, which defines a labelled subsystem $(\Sigma_3', \Sigma_3'', L_3)$ as before with $\Sigma_3' = \Sigma_1' \cup \{C\}$ with a proper subset $\Sigma_1' \subsetneq \Sigma_1$ containing C_1, but not C_2. Thus, Σ_3' and Σ_3'' are stratified.

In both cases we have obtained a partition $\Sigma = \Sigma_1 \cup \Sigma_2' \cup \Sigma_3'$ with stratified labelled subsystems $(\Sigma_1, \Sigma_1, L_1)$, $(\Sigma_2', \Sigma_2'', L_2)$ and $(\Sigma_3', \Sigma_3'', L_3)$. Since the additional condition for local stratification is easily verified, we conclude that Σ is locally stratified.

For the general case we may assume that $\Sigma_0 = \Sigma_1 \cup (\Sigma_2 - \{C\})$ is locally stratified by successive application of the constructions in the first part of this proof. Then we observe that in the case of non-stratified Σ_0 we do not change labels, when we add C. However, it may happen that one of these labels now is $\sim L$. This label results (as label L_1) from adding C' to some stratified constraint set. From the construction of this local stratification and the fact that Σ_2 is stratified we conclude that the other labels L_2 and L_3 are different from $\sim L$, which guarantees the local stratification condition to hold also in the general case.

For the case of Σ_0 being stratified the arguments are the same as before except for the case that Σ_0 contains exactly one clause C' with $\sim L'$ and none with L'. Then the corresponding stratum may also contain clauses C_i with literals $\sim L_i$ and L_{i+1} $(i = 1, \ldots, m)$, where L_1 occurs in C' and $L_{m+1} = L$.

In particular, we have $C_m \in \Sigma_2$ and adding C to this stratum is no longer possible. Since Σ_2 is stratified, we must have $m > 0$, but then the literals L', L_1 and $\sim L$ define a local stratification with associated constraint sets $\Sigma_0 - \{C'\} \cup \{C\}$, $\Sigma_0 - \{C_m\} \cup \{C\}$ and Σ_0 respectively. □

We shall use Lemma 14 in the proof of NP-hardness to shrink propositional constraint sets. Another way to reduce the technical complexity of that proof is to drop the restriction on Σ to contain only clauses with at least one negative literal. If Σ is a set of propositional clauses containing neither the atom q nor its negation, we add $\neg q$ to each clause to form the set Σ^{ext} of clauses.

Lemma 15 Let Σ be a set of propositional clauses each with at least two literals. Then Σ^{ext} is locally stratified iff Σ is satisfiable and locally stratified.

Proof. First let Σ be locally stratified and satisfiable. If Σ is not stratified, we may choose the same labels to obtain a local stratification for Σ^{ext}.

Thus, assume Σ to be stratified. Then $(\Sigma^{\text{ext}}, \Sigma, \neg q)$ is a stratified labelled subsystem. Since all clauses in all other labelled subsystems contain the literal $\neg q$, we have to isolate these clauses. Therefore, take a model for Σ which is given by a set $\{L_1, \ldots, L_n\}$ of literals occurring in Σ which must be interpreted as *true*. Taking $\sim L_i$ as a label and the corresponding labelled subsystem $(\Sigma_i', \Sigma_i'', \sim L_i)$, we obtain a proper subset $\Sigma_i' \subsetneq \Sigma^{\text{ext}}$. For $\#\Sigma_i'' > 1$ we may proceed with the other literals $\sim L_j$. The last step results in unary sets $\{\neg q \vee L_k\}$ which are obviously stratified.

Conversely, given a local stratification for Σ^{ext} we can remove $\neg q$ to obtain a local stratification for Σ. It remains to show that Σ is satisfiable. If Σ^{ext} is

stratified, this is obvious, because a literal L with $\sim L$ occurring in some clause in Σ cannot occur in any clause of Σ_0.

If Σ^{ext} is not stratified, there is at least one stratified labelled subsystem (Σ', Σ'', L) such that $\neg q$ occurs in all clauses in Σ'', i.e. $\Sigma'' = \Sigma_0^{\text{ext}}$ and Σ_0 is satisfiable. This still holds if we put back the literal L and extend our interprete L as *false* to satisfy clauses in $\Sigma - \Sigma_0$. □

Theorem 16 Let Σ be a set of constraints. Then checking that Σ is locally stratified is NP-hard.

Proof. We show that the *disjoint cover problem* (DCP) – which is known to be NP-complete – can be reduced in polynomial time to the local stratification problem. For this, let $(X, \mathcal{S})$ be an instance of DCP, i.e. X is a finite set, say $X = \{x_1, \ldots, x_n\}$ and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a subset of the power set $\mathfrak{P}(\mathfrak{X})$. The problem is to decide, whether a subset $\mathcal{S}' \subseteq \mathcal{S}$ exists such that X is the disjoint union of the sets in $\mathcal{S}'$. Such a $\mathcal{S}'$ is called a *solution* for $(X, \mathcal{S})$.

Without loss of generality we may always assume that $X = \bigcup_{S_i \in \mathcal{S}} S_i$ holds. Moreover, we may allow $\mathcal{S}$ to be a multiset.

We now associate with $(X, \mathcal{S})$ a set of constraints Σ. For this let p_{ij} be a propositional atom for all $x_i \in S_j$. For $S_i = \{x_{j_1}, \ldots, x_{j_i}\} \in \mathcal{S}$ we define clauses $\neg p_{j_k i} \vee p_{j_\ell i}$ and $\neg p_{j_\ell i} \vee p_{j_k i}$ for $k, \ell \in \{1, \ldots, i\}$, $k \neq \ell$. We refer to these clauses as *connection clauses* with respect to S_i. For $x_i \in S_j \cap S_k$ $(j \neq k)$ we define an *exclusion clause* $\neg p_{ij} \vee \neg p_{ik}$. Finally, for each x_i we define a *cover clause* $p_{ij_1} \vee \cdots \vee p_{ij_m}$ for the sets $S_{j_1}, \ldots, S_{j_m} \in \mathcal{S}$ containing x_i provided $m \geq 2$. Σ contains all these connection, exclusion and cover clauses.

Then we have to show that $(X, \mathcal{S})$ has a solution iff Σ is locally stratified and satisfiable. For this we introduce a partial order $\leq$ on DCP-instances letting $(X_1, \mathcal{S}_1) < (X_2, \mathcal{S}_2)$ iff

$$\sum_{S \in \mathcal{S}_1} |\, S \,| < \sum_{S \in \mathcal{S}_2} \quad \text{or} \quad \left(\sum_{S \in \mathcal{S}_1} = \sum_{S \in \mathcal{S}_2} \quad \text{and} \quad |\, \mathcal{S}_1 \,| > |\, \mathcal{S}_2 \,| \right)$$

holds.

First let $\mathcal{S}' = \{S_{i_1}, \ldots, S_{i_k}\}$ be a solution for $(X, \mathcal{S})$. Then Σ is obviously satisfiable. In order to use induction with respect to $\leq$ we consider the following two operations:

- Replace $S_j \in \mathcal{S}'$ by $S_j - \{x_\ell\}$ and add $S_{m+1} = \{x_\ell\}$ for some $x_\ell \in S_j$.
- Replace $S_j \notin \mathcal{S}'$ by $S_j - \{x_\ell\}$ for some $x_\ell \in S_j$.

In both cases we obtain a smaller DCP-instance which has a solution. By induction the corresponding constraint set Σ_1' is locally stratified.

In the first case we remove all clauses with literals $p_{i,m+1}$ from Σ_1'. The resulting subset Σ_1'' is still locally stratified. Now build the labelled subsystem (Σ', Σ'', L) with the label $L = \neg p_{\ell j}$.

The clauses in Σ' (and hence in Σ'') do not contain $p_{\ell j}$, i.e. we omit the cover clause with respect to x_ℓ and connection clauses containing $p_{\ell j}$ with respect to $x_\ell \in S_j$. Clauses in Σ'' containing $\neg p_{\ell j}$ arise from the restriction to keep at least two literals, hence must also lie in Σ'. Therefore, we obtain $\Sigma'' = \Sigma_1 \cup \Sigma_2$, where Σ_2 is stratified and contains only clauses with two literals, one of them is $\neg p_{\ell j}$, whereas clauses in Σ_1 do not contain $\neg p_{\ell j}$.

Thus, the remaining connection clauses with respect to $x_\ell \in S_j$ and the exclusion clauses with respect to $x_\ell \in S_j$ occur in Σ_2. This implies $\Sigma_1 = \Sigma_1''$. From Lemma 14 we conclude that Σ'' is locally stratified.

In the second case we build the labelled subsystem (Σ', Σ'', L) with the label $L = p_{\ell j}$. The clauses in Σ' (and hence in Σ'') do not contain $\neg p_{\ell j}$, i.e. we omit exclusion clauses and connection clauses containing $\neg p_{\ell j}$ with respect to $x_\ell \in S_j$. Again, the clauses in Σ'' containing $p_{\ell j}$ only arise from the restriction to keep at least two literals. Hence, these clauses define a stratified subset Σ_2 of Σ'' (and of Σ') containing only clauses with two literals.

The remaining clauses form a subset Σ_1 and clauses in Σ_1 do not contain $p_{\ell j}$, i.e. the remaining connection clauses with respect to $x_\ell \in S_j$ and the cover clause with respect to x_ℓ (if it contains just two literals) occur in Σ_2, which implies $\Sigma_1 = \Sigma_1'$. From Lemma 14 we conclude that Σ'' is locally stratified.

Since in the first case ($x_\ell \in S_j \in \mathcal{S}'$) only the cover clause with respect to x_ℓ and connection clauses containing $p_{\ell j}$ and in the second case ($x_\ell \in S_j \notin \mathcal{S}'$) only exclusion clauses with respect to $x_\ell \in S_j$ and connection clauses containing $\neg p_{\ell j}$ are omitted in Σ', the additional condition for local stratification is easily verified, if all such choices are taken provided there are at least three such possibilities. The only critical case arises, if there are only three choices of the second kind, all with the same x_ℓ. In this case we must have another $S_j = \{x_\ell\} \in \mathcal{S}'$ and we simply add the labelled subsystem $(\Sigma', \Sigma'', \neg p_{\ell j})$ to satisfy the additional local stratification condition.

If there are at most two choices, then either

- $\mathcal{S} = \mathcal{S}'$ and there is exactly one $S_j = \{x_k, x_\ell\}$ or
- $\mathcal{S}'$ contains only unary sets and these are exactly $S_j = \{x_j\} \notin \mathcal{S}'$ and $S_k = \{x_k\} \notin \mathcal{S}'$ or
- $\mathcal{S}'$ contains only unary sets and there is exactly one $S_j = \{x_k, x_\ell\} \notin \mathcal{S}'$.

In the first case Σ contains only two connection clauses with respect to S_j and hence is obviously stratified. In the second case Σ contains only four clauses

$$\neg p_{kk} \vee \neg p_{kk'},\ p_{kk} \vee p_{kk'},\ \neg p_{jj} \vee \neg p_{jj'} \quad \text{and} \quad p_{jj} \vee p_{jj'}$$

for $S_{j'} = \{x_j\} \in \mathcal{S}'$ and $S_{k'} = \{x_k\} \in \mathcal{S}'$, hence Σ is stratified.

In the third case we obtain six clauses

$$\neg p_{kj} \vee p_{\ell j},\ \neg p_{\ell j} \vee p_{kj},\ \neg p_{kj} \vee p_{kk'},\ \neg p_{\ell j} \vee p_{\ell\ell'},\ p_{kj} \vee p_{kk'} \quad \text{and} \quad p_{\ell j} \vee p_{\ell\ell'}$$

for $S_{k'} = \{x_k\} \in \mathcal{S}'$ and $S_{\ell'} = \{x_\ell\} \in \mathcal{S}'$. Using Lemma 14 it is easily verified that the labels p_{kj}, $p_{\ell j}$, $\neg p_{kj}$ and $\neg p_{\ell j}$ define a partition into stratified labelled subsystems.

For the converse let us first assume that Σ is stratified, i.e. there cannot exist three clauses with literals L, L and $\sim L$ respectively. In connection clauses we may have $L = p_{\ell j}$ (or $L = \neg p_{\ell j}$) and it follows that Σ does not contain exclusion or cover clauses for $x_\ell \in S_j$. This implies $x_\ell \notin S_k$ for all $k \neq j$. If we have an exclusion clause for $x_\ell \in S_j$, say $\neg p_{\ell j} \vee \neg p_{\ell k}$, then we also have a cover clause $p_{\ell j} \vee p_{\ell k} \vee C'$ and vice versa, but there cannot be further exclusion clauses nor connection clauses for $x_\ell \in S_j$, i.e. $C' \equiv \mathit{false}$ and $S_j = \{x_\ell\}$.

To summarize, if x_ℓ occurs in more than one S_j, then $\#S_j = 1$ and there are just two such sets. Therefore, for a solution $\mathcal{S}'$ we take all S_j with $\#S_j \geq 2$ and select a singleton set $\{x_\ell\}$ for the remaining elements.

Next assume that Σ is locally stratified, i.e. there is a local stratification with labels $L_1, \ldots, L_n$ ($n \geq 3$). Again, we proceed by induction on DCP-instances.

For $L_1 = \neg p_{\ell j}$ and the stratified labelled subsystem $(\Sigma_1', \Sigma_1'', L_1)$ the cover clause for x_ℓ and connection clauses for $x_\ell \in S_j$ containing $p_{\ell j}$ have been removed from Σ to give Σ_1', hence must occur in two other labelled subsystems such that for a label $\neg p_{ki}$ we must have $k\neg\ell$ and for a label p_{ki} we must have $i \neq j$.

Analogously, for $L_1 = p_{\ell j}$ exclusion and connection clauses for $x_\ell \in S_j$, the latter ones containing $\neg p_{\ell j}$ have been removed omitted in Σ_1' and must occur in two other labelled subsystems such that for another positive label p_{ki} we must have $k\neg\ell$ and for a negative label $\neg p_{ki}$ we must have $i \neq j$. Hence, for the minimum number of three labels L_1, L_2 and L_3 we obtain the following four cases:

$$\begin{aligned}
L_1 = \neg p_{\ell j},\ L_2 = \neg p_{k_1 i_1},\ L_3 = \neg p_{k_2 i_2} &\quad \text{with pairwise different } \ell, k_1, k_2 \quad , \\
L_1 = \neg p_{\ell j},\ L_2 = \neg p_{k_1 i_1},\ L_3 = p_{k_2 i_2} &\quad \text{with } \ell \neq k_1 \text{ and } j \neq i_2 \neq i_1 \quad , \\
L_1 = \neg p_{\ell j},\ L_2 = p_{k_1 i_1},\ L_3 = p_{k_2 i_2} &\quad \text{with } k_1 \neq k_2 \text{ and } i_1 \neq j \neq i_2 \quad \text{or} \\
L_1 = p_{\ell j},\ L_2 = p_{k_1 i_1},\ L_3 = p_{k_2 i_2} &\quad \text{with pairwise different } \ell, k_1, k_2 \quad .
\end{aligned}$$

For a negative literal $L_i = \neg p_{\ell j}$ or a positive literal $L_i = p_{\ell j}$ it follows from Lemma 14 that replacing S_j by $S_j - \{x_\ell\}$ and $\{x_\ell\}$ defines a locally stratified constraint set. Therefore, by induction in all four cases (with the restrictions for indices) we obtain solutions for smaller DCP-instances with

$$\begin{aligned}
\mathcal{S}_1 &= \{S_1, \ldots, S_j - \{x_\ell\}, \ldots, S_m, \{x_\ell\}\} \quad , \\
\mathcal{S}_2 &= \{S_1, \ldots, S_{i_1} - \{x_{k_1}\}, \ldots, S_m, \{x_{k_1}\}\} \quad \text{and} \\
\mathcal{S}_3 &= \{S_1, \ldots, S_{i_2} - \{x_{k_2}\}, \ldots, S_m, \{x_{k_2}\}\}
\end{aligned}$$

respectively. If any of these solutions contains both (or none) of the splitted components, e.g. $S_j - \{x_\ell\}$ and $\{x_\ell\}$, we also have a solution for the original problem.

Therefore, assume that all solutions for $(X, \mathcal{S}_i)$ must contain exactly one of the splitted components denoted as $\overline{S_1}$, $\overline{S_2}$ and $\overline{S_3}$. Let $\mathcal{S}_i' = \{S_1^i, \ldots, S_{n_i}^i, \overline{S_i}\}$ be a solution for $(X, \mathcal{S}_i)$. For $i \neq j$ we proceed in the following way:

Start with $\mathcal{T}_i = \mathcal{S}'_i - \mathcal{S}'_j$, $\mathcal{T}_j = \mathcal{S}'_j - \mathcal{S}'_i$ and $\mathcal{T} = \{\overline{S_j}\}$ and execute the following steps until there are no more changes:

- Remove all sets from $\mathcal{T}_i$ intersecting some set in $\mathcal{T}$ and let these define a new $\mathcal{T}$.
- Remove all sets from $\mathcal{T}_j$ intersecting some set in $\mathcal{T}$ and let these define a new $\mathcal{T}$.

Finally, if $\mathcal{T}_i$ (and then also $\mathcal{T}_j$) are non-empty, this means that we may replace $\mathcal{T}_j \subseteq \mathcal{S}'_j$ by $\mathcal{T}_i$ or $\mathcal{S}'_j - \mathcal{T}_j$ by $\mathcal{S}'_i - \mathcal{T}_i$. According to our assumption on solutions we always keep either $\overline{S_i}$ or $\overline{S_j}$. Consequently, the procedure above defines a chain

$$\overline{S_i} - S^j_{i_1} - S^i_{i_1} - S^j_{i_2} - S^i_{i_2} - \cdots - S^j_{i_k} - S^i_{i_k} - \overline{S_j} \quad ,$$

where neighbouring sets have a common element. This is still true, if we replace $\overline{S_i}$ by the original S_j. Taking together all three choices for (i,j) we obtain an odd-length cycle

$$S_{i_1} - S_{i_2} - S_{i_3} - \cdots - S_{i_m} - S_{i_1}$$

with intersecting neighbouring sets $S_{i_j} \in \mathcal{S}$. Let Σ' be the set of constraints corresponding to $\{S_{i_1}, \ldots, S_{i_m}\}$. Then Σ' differs from a subset $\Sigma_0 \subseteq \Sigma$ only by the fact that cover clauses may have been shortened. Since omitted (positive) literals in these cover clauses do not occur in any other clauses in Σ', this must be locally stratified iff Σ_0 is locally stratified. Therefore, the proof is completed, if we can show that cycles as above always define constraint sets that are not locally stratified or not satisfiable.

With each neighbouring pair $(S_{i_j}, S_{i_{j+1}})$ we may associate a witness $x \in S_{i_j} \cap S_{i_{j+1}}$. Then without loss of generality (just rename indices) we can always assume a cycle

$$S_1 \overset{x_1}{-} S_2 \overset{x_2}{-} S_3 - \cdots - S_m \overset{x_m}{-} S_{m+1} = S_1$$

and show that the following conditions can be achieved:

- m is odd,
- the x_i are pairwise different,
- the S_i are pairwise different and
- the cover clause in Σ' for x_ℓ has the form $p_{\ell\ell} \vee p_{\ell\ell+1} \vee C'_\ell$, where literals in C'_ℓ do not occur in any other clause in Σ'.

The last condition will allow us to assume without loss of generality that cover clauses in Σ' only contain two literals.

In order to achieve such a cycle recall that our original cycle is composed of three subpaths (called *flanks*) corresponding to a solution of a smaller DCP-instance and each pair of flanks has a common set (called *corner*). If $\overline{S_i} \subsetneq S_j$ is such a corner, then the following cases may arise:

- The two nieghbours S_i and S_k coincide which allows to remove the corner S_j and to identify S_i with S_k.
- If S_i, S_j and S_k are pairwise different, we either obtain a simple cycle of length 3 or let the cycle unchanged.
- If one of the neighbours equals S_j, say $S_k = S_j$, then S_k is not common in the solutions for the flank with S_j and S_k, i.e. there must be some $S_{j'}$ in the same solution as $\overline{S_i}$ with $S_j \cap S_{j'} \neq \emptyset$. In this case we may replace the even number of edges between S_j and $S_{j'}$ by a single edge. By the same argument the even number of edges between the opposite edge S_ℓ (in the same flank) and some $S_{\ell'}$ by a single edge.

In all these cases the cycle length remains odd.

If x_i occurs twice, say between Si_1 and S_{i_2} and between Si_3 and S_{i_4} respectively, we may assume paths from S_{i_1} to S_{i_4} and from S_{i_2} to S_{i_3} of length n_1 and n_2 respectively. Then there are cycles with S_{i_2}, S_{i_3} and S_{i_1}, S_{i_4} connected by x_i respectively and one of the corresponding lengths $n_1 + 1$ or $n_2 + 1$ must be odd. The only critical cases occur for $S_{i_2} = S_{i_4}$ or $S_{i_1} = S_{i_3}$, but these correspond to corners that have already been removed.

Finally, in order to achieve the condition on cover clauses consider $S_i \cap S_j \neq \emptyset$.

- If S_i and S_j belong to different flanks, but to the same solution, then we have $S_i = S_j$ and we may identify them and remove the even number of edges between them.
- If S_i and S_j belong to different flanks and different solutions, then for $S_i \neq S_j$ we may replace the odd number of edges between them by a single new edge, whereas for $S_i = S_j$ we may consider the odd number of edges between them as our new cycle.
- If S_i and S_j belong to the same flank, then the number of edges between them is even iff $S_i = S_j$, thus may be removed or replaced by a single new edge.

The conditions on our cycle now allows clauses to be arranged in such a way that we have

$$\Sigma' \quad = \quad \{\sim L_1 \vee L_2,\ \sim L_2 \vee L_3, \ldots, \sim L_{p-1} \vee L_p,\ \sim L_p \vee L_1\}$$

for an even number p with $L_{p/2+i} = \sim L_i$ for $i = 1, \ldots, p/2$. Such a Σ', however, is not satisfiable. □

8 Conclusion

In this article we investigated the limits of rule triggering systems (RTSs) for maintaining database integrity under the additional requirement to preserve the effects

of transitions. The first result assures the existence of non-repairable transitions. In order to disallow such transitions the constraint implication problem must be decidable.

Secondly, we analyzed critical trigger paths in rule hypergraphs associated with RTSs. We could show that the existence of critical trigger paths leads to RTSs which may invalidate the effect of some transitions, even if these are repairable. Such a behaviour can only be excluded for *locally stratified* constraint sets. In this case the needed RTS can be computed effectively, but checking local stratification is NP-hard.

To summarize, both results limit the applicability of RTSs for integrity maintenance under the assumption that the intended effects of user-defined transitions should be preserved. Fortunately, there is a stronger condition on a constraint set to be *stratified*, which is only sufficient for reasonable rule behaviour, but not necessary. Stratified constraint sets occur, if we have a relational database schema in Entity-Relationship normal form, which means that it is equivalent to an ER-schema without exclusion constraints. Checking stratification is not only effective, but also efficient.

On the other hand, the RTS approach to integrity maintenance completely ignores user-defined transitions. Thus, a second conclusion from our studies is that these should be taken into consideration.

References

[1] S. Abiteboul, V. Vianu: *Equivalence and Optimization of Relational Transactions*, Journal of the ACM, vol. 35(1), 1988, 70-120

[2] S. Ceri, J. Widom: *Deriving Production Rules for Constraint Maintenance*, Proc. 16th Conf. on VLDB, Brisbane (Australia), August 1990, 566-577

[3] S. Ceri, P. Fraternali, S. Paraboschi, L. Tanca: *Automatic Generation of Production Rules for Integrity Maintenance*. ACM ToDS, vol. 19(3), 1994, 367-422.

[4] S. Chakravarty, J. Widom (Eds.): *Research Issues in Data Engineering — Active Databases*, Proc., Houston, Februar 1994

[5] M. Gertz, U. W. Lipeck: *Deriving Integrity Maintaining Triggers from Transition Graphs*, in Proc. 9th ICDE, IEEE Computer Society Press, 1993, 22-29

[6] H. Mannila, K.-J. Räihä: *The Design of Relational Databases*, Addison-Wesley 1992

[7] K.-D. Schewe, B. Thalheim: *Consistency Enforcement in Active Databases*, in S. Chakravarty, J. Widom (Eds.): *Research Issues in Data Engineering — Active Databases*, Proc., Houston, Februar 1994

[8] K.-D. Schewe, B. Thalheim: *Active Consistency Enforcement for Repairable Database Transitions*, in S.Conrad, H.-J. Klein, K.-D. Schewe (Eds.): Integrity in Databases, Proc. 6th Int. Workskop on Foundations of Models and Languages for Data and Objects, Schloß Dagstuhl, 1996, 87-102, available via `http://wwwiti.cs.uni-magdeburg.de/~conrad/IDB96/Proceedings.html`

[9] B. Thalheim: *Foundations of entity-relationship modeling*, Annals of Mathematics and Artificial Intelligence, vol. 7, 1993, 197-256

[10] S. D. Urban, L. Delcambre: *Constraint Analysis: a Design Process for Specifying Operations on Objects*, IEEE Trans. on Knowledge and Data Engineering, vol. 2 (4), December 1990

[11] J. Widom, S. J. Finkelstein: *Set-oriented Production Rules in Relational Database Systems*, in Proc. SIGMOD 1990, 259-270

Received, April, 1997