

Reflections on Some Fundamental Issues of Rule-based Incremental Update Propagation*

Rainer Manthey

Department of Computer Science
University of Bonn
Römerstraße 164, D-53117 Bonn, Germany
rainer@informatik.uni-bonn.de

Abstract

This paper addresses two fundamental aspects of update propagation in deductive databases:

1. How to incrementally compute the sets of updates induced by a deductive rule from the changes of the data sets on which the rule depends? and
2. How to express such incremental definitions by means of deductive rules defining induced updates?

Both problems have already been addressed by many authors in various contexts. The motivation for coming up with yet another such paper comes from the impression that some of the very basic assumptions and justifications of the approaches proposed in literature have not been understood well enough. In this paper we therefore investigate the basic algebraic laws of incremental change computation as well as their direct encoding in form of deductive rules from a new perspective. In addition we address the problems arising when the entire process of update propagation is driven by a (semi-)naive fixpoint procedure applied to the rules encoding the incremental definitions of induced updates.

1 Introduction

Update propagation is one of the fundamental inference capabilities which a deductive DBMS should be able to offer. Triggered by the update of some base data, rules depending on the changed base facts have to be activated in order to determine which derived data will change in consequence. There are several reasons why it is necessary to be informed about such *induced* changes. The most important and well-known motivations are the need to check integrity constraints involving rule-defined data and the need to refresh materialized views. Both problems have been addressed in the research literature at length during the last ten years. Many different individual methods have been proposed, so that it already becomes hard to keep track of the state-of-the-art. Also for active database systems condition monitoring by means of update propagation has been proposed [RCB89].

*This work has been supported in part by the Commission of the European Community under ESPRIT project number 6333 (IDEA)

1.1 Incremental Approaches to Update Propagation

All the approaches introduced so far share a common goal, in that they try to reduce the overhead arising from a “naive” computation of induced updates based on a comparison of old and new state (before and after an update). Every method to be found in the literature aims at reaching some degree of incrementality by at least avoiding to activate rules that are definitely not affected by the change under consideration. There are several methods which are aiming at an exact computation of the difference between old and new state (e.g., [Qian91, Kue91, VBK91, Dec86, GMS93, Ull88, UO92, BDM88]). Most of them are dealing with some variation of the concept of a *delta set* (also called delta relation, or increment, or differential, or derivative, or internal event, just to mention some of the most common terms referring to more or less the same concept). A delta set of a base concept contains all the individual facts in the extension of that base concept which have been explicitly changed. Delta sets for derived concepts contain all facts which are newly derivable after the change, or cease to be derivable in the new state. The goal of the methods cited is to define derived deltas as far as possible in terms of the deltas of those data sets on which the derived concept depends. Update propagation in such a context means a stepwise computation of interdependent delta sets, related in the same way as the corresponding derived data sets are.

If comparing the essential techniques and relationships appearing in the different approaches regardless of the differences in terminology and formalism, one soon discovers that the problems arising and the solutions proposed are very close to each other. Some of the most general and concise presentations of the basic laws of incremental change computation have been given in the context of relational algebra. In Ullman’s textbook [Ull88], derivatives for monotone relational algebra have been characterized and proved, but the important (and complicated) case of the difference operator has been omitted (for sake of brevity of presentation). In a recent article by Qian and Wiederhold [Qian91] a characterization of Boolean increments has appeared which covers difference as well. However, the laws and proofs given there are so general that it is very hard (at least for this author) to apply their results to individual and concrete situations.

Other non-algebraic attempts of defining deltas are working with similar basic axioms of change, but either do not prove them, or somehow “hide” the basic laws by some other formalism. Despite all the efforts towards properly defining deltas incrementally, we did not yet encounter a characterization that we regarded as both, easily accessible and directly applicable to concrete examples, and which have been proved in a clear and appropriately formal manner. A first result of this paper is a new proposal for characterizing deltas in an algebraic style and for proving these basic laws of change correct. Although we did not discover new axioms (but confirm those which can be found behind many existing approaches), we regard the way how these axioms have been formulated and how they are justified based on just a few simple equations from basic set-theory more satisfying than what we saw before. This impression is definitely partially subjective, but will hopefully turn out to be shared by others too. Section 2 of this paper will be devoted to establishing our proposal of an algebraic framework for change computation.

1.2 Rule-based Specifications of Differentials

Apart from the goal of reaching an incremental characterization of deltas there is a second goal shared by a large number of methods proposed. Many authors have aimed at expressing the

definitions of delta sets by means of deductive rules as well, thus reducing update propagation to rule-based query evaluation. Various names have been proposed for this kind of internal rules derived by means of a compiler from source rules and used by the DBMS in order to implement update propagation. In [VBK91] such rules have been called propagation rules, [Oli91] calls them internal events rules, [Kue91] speaks about delta rules, [CGMD94] uses the term logic metaprograms, [GMS93] call them Δ -rules, and so on. The basic idea underlying all these approaches is always the same, and it is a rather straightforward one which has been discovered independently by many during the past decade. In a sense the technique of implementing update propagation by internal, declarative rules is very similar to the well-known approach of implementing deductive query answering by means of internal rules (such as the “magic” rules of the magic set method).

Rule-based specifications of deltas are closely related to the basic laws of incremental change computation addressed above, in that rule specifications are nothing else but alternative formulations of such laws. A question which has not sufficiently clearly been addressed up till now is how the two styles of specification relate. It is quite obvious that algebraic axioms like those of Qian or Ullman are closely related to rule-based formulations like those of [Kue91] or [UO92], for example. But there hasn't been any attempt to explicitly and systematically relate the two up till now. In this paper we try to establish such a relationship in a rather direct and straightforward manner.

A different stream of research in rule-based update propagation has been established in a series of papers by Ceri and Widom ([CW90], [CW91]). Here “rules” are active rules (or triggers), rather than deductive (passive) rules. Deductive rules are compiled into update propagation triggers performing computations which are exactly equivalent to those computations performed when materializing delta sets defined by means of deductive propagation rules. In fact, the problems to be handled by such active rule methods are exactly the same as those to be mastered by deductive methods (and vice versa). A companion paper in this workshop [GM94] presents another approach in this line. It is based on the same laws of change that will be discussed in this paper, but addresses update propagation in an object-oriented data model implemented by means of active propagation rules. Section 3.1 of this paper is devoted to rule-based specifications of deltas.

1.3 Update Propagation and Fixpoint Computation

Another aspect of the rule-based approach which has not received sufficient attention up till now is the question of overall organization of an update propagation process. Most of the rule-based methods propose to activate rules during integrity checking by means of a general-purpose query evaluator, be it a QSQ-like procedure as in [VBK91] or SLDNF-resolution as, e.g., used by [UO92]. Others - like [GMS93] - propose their own algorithms for “driving” the rules defining delta sets. We are not convinced that such in a sense “indirect” approaches are most promising and appropriate. Instead we believe that induced updates - if specified by means of rules themselves - should simply be computed by means of (semi-)naive fixpoint iteration, the basic procedure contained in any bottom-up query evaluator. Fixpoint iteration as such, however, is not bound to query evaluation, but can be regarded as a general-purpose materialization procedure, able to materialize all data derivable by means of a given rule set from a given set of base facts. Fixpoint iteration does not care about the meaning of the rules it is applied to: Whether these rules specify answers to queries or induced updates (or quite different things, like explanations or statistical values) is irrelevant for efficiently applying

rules within an iterative process aimed at reaching a fixpoint.

It is surprising that up till now none of the sources mentioned has explicitly pointed out the fact that update propagation can be interpreted as and implemented by a fixpoint computation as well (in very much the same way as bottom-up query evaluation). This insight is not a fundamentally new one, because it is more or less clearly implied by the techniques used for driving propagation rules in the sources mentioned. For non-recursive rule sets, it is possible to directly process the rules defining derived deltas by means of fixpoint iteration. In the recursive case, however, this is not possible, because the resulting delta rules will not be stratifiable anymore! This happens even if the source rules (defining the data sets as such) have been stratifiable. Intuitively, this phenomenon can be explained by the fact that in case of recursion induced insertions can be definitely identified only if all induced deletions have been completely determined before, and vice versa. Both requirements together cannot be satisfied. There are at least two ways out, which have been used by the methods proposed up till now in different ways. Either evaluation over the changed state of the database is "simulated" by means of an additional top-down evaluation using meta-rules, or a superset of the set of induced updates is determined using delta sets and old state only, and erroneously computed deltas are removed in a second phase based on evaluation over the new state again. It is an important result of this paper that one of these extra features *has* to be used in addition to ordinary fixpoint iteration in order to compute delta sets correctly!

In this paper, the aspect of rule-based specifications of deltas and the problem of fixpoint computations of delta sets will be addressed only rather briefly and superficially, although they deserve a much more thorough and formal treatment. However, such a systematic account of the problem of rule-based update propagation is on the one hand beyond the scope of this paper, and on the other hand not yet fully mastered. As there is no way, however, to present the algebraic laws of change and their rule-based encoding without addressing the problem of how to apply these rules, we decided to treat the latter topic in an informal and sketchy way only, instead of omitting it altogether. In this sense, the second part of our paper (section 3.2) is more of an informal overview of basic ideas and problems related to the rule-based approach in general, rather than a sound and polished treatment of the matter.

2 The Algebra of Change

Independent of the particular data model considered, and independent of the particular style in which changes are expressed, the fundamental issue of update propagation can be outlined in set-theoretic terms as follows:

- A database state can be viewed as a finite collection of named, finite sets of data elements, called base sets.
- Database changes affect one or more of these base sets by adding or deleting one or more elements from each affected set.
- The problem is to determine if and how changes to base sets give rise to changes of Boolean derivatives of such sets, i.e., to data sets derived from base sets by applying unary or binary Boolean operators (such as, e.g., projection or union).

In this section, we will introduce formal characterizations of the changes induced on Boolean combinations of base sets in terms of the changes of these base sets.

2.1 Motivation

As a motivating example illustrating the spirit of this characterization, consider two base sets

$$\mathbf{R} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\} \quad \mathbf{S} = \{\mathbf{a}, \mathbf{d}, \mathbf{e}\}.$$

As Boolean derivative consider the difference of the two sets, currently consisting of two elements:

$$\mathbf{R} \setminus \mathbf{S} = \{\mathbf{b}, \mathbf{c}\}.$$

Now assume a transaction changes both sets, \mathbf{R} and \mathbf{S} , as specified by means of the following sets of added or deleted elements (which will be called *delta sets* in the following):

$$\begin{aligned} \mathbf{R}^+ &= \{\mathbf{d}\} & \mathbf{S}^+ &= \{\mathbf{f}\} \\ \mathbf{R}^- &= \{\mathbf{b}\} & \mathbf{S}^- &= \{\mathbf{a}\} \end{aligned}$$

Applying these changes to the two base sets results in the changed sets

$$\mathbf{R}^{\text{new}} = \{\mathbf{a}, \mathbf{c}, \mathbf{d}\} \quad \mathbf{S}^{\text{new}} = \{\mathbf{d}, \mathbf{e}, \mathbf{f}\}$$

Applying the difference operator to the changed sets results in the changed derivative

$$(\mathbf{R} \setminus \mathbf{S})^{\text{new}} = \mathbf{R}^{\text{new}} \setminus \mathbf{S}^{\text{new}} = \{\mathbf{a}, \mathbf{c}\}.$$

By comparing the new and the old extension of the difference, the induced changes can be determined and expressed by means of delta sets as well:

$$(\mathbf{R} \setminus \mathbf{S})^+ = \{\mathbf{a}\} \quad (\mathbf{R} \setminus \mathbf{S})^- = \{\mathbf{b}\}.$$

Instead of computing such delta sets from the difference of old and new state (resp., new and old state), however, one would like to be able to directly compute the delta set of a Boolean derivative from the delta sets of its operands. It is the purpose of this section to introduce and prove such incremental derivations. As an example of the kind of characterizations we have in mind consider the following general formula for computing changes of set differences (to be formally introduced and proved later in this section):

$$(\mathbf{R} \setminus \mathbf{S})^+ = (\mathbf{R}^+ \setminus \mathbf{S}^{\text{new}}) \cup (\mathbf{R}^{\text{new}} \cap \mathbf{S}^-).$$

Applying this incremental formula to the example sets above yields exactly the same result as obtained by non-incremental "hand computation" before, namely

$$(\mathbf{R} \setminus \mathbf{S})^+ = (\{\mathbf{d}\} \setminus \{\mathbf{d}, \mathbf{e}, \mathbf{f}\}) \cup (\{\mathbf{a}, \mathbf{c}, \mathbf{d}\} \cap \{\mathbf{a}\}) = \{\mathbf{a}\}.$$

The incremental characterization still refers to the updated base sets \mathbf{R}^{new} and \mathbf{S}^{new} , rather than only to the unchanged base sets and their increments, respectively. It is possible, however, to express each changed set \mathbf{P}^{new} systematically in terms of the old version \mathbf{P} and of the increments \mathbf{P}^+ and \mathbf{P}^- according to the equation

$$\mathbf{P}^{\text{new}} = (\mathbf{P} \setminus \mathbf{P}^-) \cup \mathbf{P}^+.$$

Thus a truly incremental characterization of the derived delta set has been obtained. It was the purpose of this example to introduce the problem of computing induced changes of Boolean derivatives and to outline the kind of solutions we have in mind. In the following paragraphs we will formally address the problem of incrementally computing delta sets.

2.2 A Formal Basis of Update Propagation

In order to address the set-theoretic formulation of the update propagation problem systematically, we need a precise formalization of the assumptions we want to make about the problem domain. We assume that there is a database state Σ , consisting of a finite collection of finite sets of elements. We do not care about the kind of data elements that may be contained in these sets, but just state that they come from some universal domain and that for the purpose of this section they are considered n-tuples of atomic elements (which may be interpreted as relational tuples, or as records representing the state of an object, etc.).

Furthermore, we assume that a simple or complex change updates Σ . This change is formally represented by so-called *delta sets* associated with each data set in the database. For each $S \in \Sigma$, there are finite, possibly empty sets S^+ and S^- , consisting of those data elements that will be added to, or removed from S , resp., when performing the update under consideration.

The following assumptions about delta sets are important and will influence the algebraic laws to be discussed in this section. For each $S \in \Sigma$ the respective delta sets are related in the following way. First, only new elements, not in S before the change, are inserted, i.e.,

$$S^+ \cap S = \emptyset. \quad (1)$$

Second, only elements already contained in S before the change may be deleted, i.e.,

$$S^- \subseteq S. \quad (2)$$

Thirdly, an element may not be simultaneously inserted in and deleted from one and the same set, i.e.,

$$S^+ \cap S^- = \emptyset. \quad (3)$$

At least one of the component sets of Σ is supposed to change, i.e., for at least one $S \in \Sigma$

$$S^+ \neq \emptyset \vee S^- \neq \emptyset. \quad (4)$$

After having performed the change(s) expressed by the delta sets, each data set S is assumed to be in a new state, denoted S^{new} . In the following, we will denote the state before the change by S^{old} as well. The delta sets, the components of S^{old} and those of S^{new} are related in the following way:

$$S^{\text{new}} = (S^{\text{old}} \cup S^+) \setminus S^-, \quad (5)$$

or alternatively

$$S^{\text{new}} = (S^{\text{old}} \setminus S^-) \cup S^+ \quad (6)$$

Due to our above assumptions, the two characterizations of the new state are equivalent and will be used in the one or the other way according to needs. Another - obvious - way of reading this relationship can be regarded as a "definition" of the delta sets in terms of old and new state:

$$S^+ = S^{\text{new}} \setminus (S^{\text{old}} \cup S^-)$$

$$S^- = S^{\text{old}} \setminus (S^{\text{new}} \setminus S^+)$$

Because of (1) and (2), these equations can be simplified into a form that will be frequently used within proofs:

$$S^+ = S^{\text{new}} \setminus S^{\text{old}} \quad (7)$$

and

$$S^- = S^{\text{old}} \setminus S^{\text{new}} \quad (8)$$

Finally, we introduce a notation for abbreviating one of the more frequently appearing subexpressions:

$$S^0 = S^{\text{old}} \setminus S^- \quad (9)$$

denoting those elements of S^{old} , that are not deleted, or equivalently those elements of S^{new} that have already been in S^{old} before the change. We will call the elements of these sets "preserved" elements. Using this additional notation, the new state may alternatively be characterized incrementally

$$S^{\text{new}} = S^0 \cup S^+. \quad (10)$$

Similarly, the old state may be characterized as

$$S^{\text{old}} = S^0 \cup S^-. \quad (11)$$

2.3 Some Useful Transformations from Set-theory

Our main task while proving incremental characterizations of derived delta sets correct will consist in applying equivalence preserving transformations to instances of equations defining delta sets of derived sets in terms of new and old state (in the spirit of (7) and (8)). In this paragraph we will list and prove those transformations that are required for deriving incremental characterizations of delta sets.

For each of the binary Boolean operators $\cup, \cap, \setminus, \times, \bowtie$, we will have to perform more or less the same task, namely to distribute the difference operator over the respective Boolean operator under consideration appearing in both arguments of the difference. This is due to the fact that delta sets are defined by means of the difference between new and old (old and new, resp.) states. As an example consider the set of those elements that are implicitly inserted into the union of two sets A and B . According to (7) the delta set of the union of A and B is

$$(A \cup B)^+ = (A^{\text{new}} \cup B^{\text{new}}) \setminus (A^{\text{old}} \cup B^{\text{old}}).$$

In order to arrive at a characterization of the derived delta set in terms of the delta sets of A and B , we will have to distribute the main operator \setminus over the two occurrences of the \cup operator. In a similar way, all the other Boolean operators have to be treated. From the following collection of *distribution lemmata* it can be seen that in each case there are some significant differences to be observed, which are due to the nature of the particular operator appearing in the operands of a difference expression.

In the following, let A_1, A_2, B_1 and B_2 denote sets such that A_i and B_i are union compatible. (In the subsequent proofs making use of the lemmata, the A_i will denote new versions of the two operands, the B_i will denote old versions, or vice versa.)

Lemma 1 *Difference distributes over union by turning one of the union operators into difference.*

$$(A_1 \cup A_2) \setminus (B_1 \cup B_2) = \quad (12)$$

$$[(A_1 \setminus B_1) \setminus B_2] \cup [(A_2 \setminus B_2) \setminus B_1]$$

Proof:

$$\begin{aligned} x \in (A_1 \cup A_2) \setminus (B_1 \cup B_2) &\Leftrightarrow \\ (x \in A_1 \vee x \in A_2) \wedge \neg(x \in B_1 \vee x \in B_2) &\Leftrightarrow \\ [x \in A_1 \wedge \neg(x \in B_1 \vee x \in B_2)] \vee [x \in A_2 \wedge \neg(x \in B_1 \vee x \in B_2)] &\Leftrightarrow \\ [(x \in A_1 \wedge \neg x \in B_1) \wedge \neg x \in B_2] \vee [(x \in A_2 \wedge \neg x \in B_2) \wedge \neg x \in B_1] &\Leftrightarrow \\ x \in [(A_1 \setminus B_1) \setminus B_2] \cup [(A_2 \setminus B_2) \setminus B_1] &\quad \square \end{aligned}$$

In one of the cases we will have to discuss, a different kind of distribution law for difference and union is required, which refers to just one occurrence of the union operator.

Lemma 2 *Let B denote another data set. Then the following holds:*

$$(A_1 \cup A_2) \setminus B = [(A_1 \setminus B) \cup (A_2 \setminus B)] \quad (13)$$

This lemma can be regarded as a special case of the first one.

In a similar way, lemmata for the other Boolean operators can be proved. We will omit the proofs, as they don't add new insight, but merely apply the standard technique of reducing set equations to equivalences in predicate calculus. The lemmata as such, however, are important for the proofs of the incremental equations to be introduced in the following and will thus be given in this paragraph.

Lemma 3 *Difference distributes over intersection without changing any of the operators. However, the direction of distribution is opposite to the direction in the case of union, i.e., the A_i distribute, not the B_i .*

$$(A_1 \cap A_2) \setminus (B_1 \cap B_2) = \quad (14)$$

$$[(A_1 \setminus B_1) \cap A_2] \cup [(A_2 \setminus B_2) \cap A_1]$$

The Boolean operators product and join are closely related to intersection (in that both join and intersection can be viewed as special cases of product). It is therefore not surprising that product and join distribute in very much the same way as intersection (although the proofs for these more general operators are slightly more elaborate). We omit the lemmata stating distribution laws of difference over product and join, resp., which differ from the lemma for intersection only in the operator considered.

The following lemma about the distribution of an outermost difference operator over two inner occurrences of \setminus concludes this paragraph and can be proved in a similar way as the other lemmata.

Lemma 4 *Difference distributes over difference by changing one of the operators as well as the order of the arguments in one of the two operands.*

$$(A_1 \setminus A_2) \setminus (B_1 \setminus B_2) = \quad (15)$$

$$[(A_1 \setminus B_1) \setminus A_2] \cup [(B_2 \setminus A_2) \cap A_1]$$

2.4 Propagating Updates into Boolean Expressions

Now we are in a position to introduce and prove equations incrementally characterizing the delta sets corresponding to a Boolean expression in terms of the delta sets of its operands before the change. For each Boolean operator $\Phi \in \{\cup, \cap, \setminus, \times, \bowtie\}$, the corresponding derived delta set is defined by simply applying the defining equation for delta sets given above to the operands of the respective Boolean operator. This means for operand sets **A** and **B**:

$$(\mathbf{A} \Phi \mathbf{B})^+ =_{\text{def}} [\mathbf{A}^{\text{new}} \Phi \mathbf{B}^{\text{new}}] \setminus [\mathbf{A}^{\text{old}} \Phi \mathbf{B}^{\text{old}}] \quad (16)$$

$$(\mathbf{A} \Phi \mathbf{B})^- =_{\text{def}} [\mathbf{A}^{\text{old}} \Phi \mathbf{B}^{\text{old}}] \setminus [\mathbf{A}^{\text{new}} \Phi \mathbf{B}^{\text{new}}] \quad (17)$$

This definition of deltas for Boolean derivatives, however, is not an incremental one, in that it refers to the entire new state of the operands, rather to the increments (delta sets). An intuitive expectation one might have on first thought is that increment operators and Boolean operators might simply distribute, i.e. that:

$$(\mathbf{A} \Phi \mathbf{B})^+ = \mathbf{A}^+ \Phi \mathbf{B}^+$$

$$(\mathbf{A} \Phi \mathbf{B})^- = \mathbf{A}^- \Phi \mathbf{B}^-.$$

However, this assumption is wrong! Instead the general structure of the incremental characterizations looks as follows

$$(\mathbf{A} \Phi \mathbf{B})^+ = (\mathbf{A}^{\delta_1} \sigma_1 \mathbf{B}^{\tau_1}) \cup (\mathbf{B}^{\delta_2} \sigma_2 \mathbf{A}^{\tau_2})$$

$$(\mathbf{A} \Phi \mathbf{B})^- = (\mathbf{A}^{\delta_3} \sigma_3 \mathbf{B}^{\tau_3}) \cup (\mathbf{B}^{\delta_4} \sigma_4 \mathbf{A}^{\tau_4})$$

where $\delta_i \in \{+, -\}$, σ_i are Boolean operators, and $\tau_i \in \{\text{new}, \text{old}\}$, depending on the particular Φ under consideration. This general format can be explained in a rather straightforward manner. The fact that the delta set of a derivative is always composed of two components (the operands of the outermost union operator) is explained by the fact that each of these two components contains one of the delta sets of **A** and **B**, respectively: Each argument's increment contributes individually to the derived increment. Each of the delta sets of the two arguments, however, is combined with a *residue*, i.e., with either the new, or the old state of the other argument. Intuitively this can be explained by the fact that the contribution of one of the arguments' increments to the overall increment might be invalidated by a change simultaneously affecting the other argument.

In order to identify the exact way how the individual Boolean operators "behave" with respect to incremental change propagation, we have to consider each of them in turn. We will first look at the classical set-operators union, intersection, difference and Cartesian product. In the following, **Q** and **R** denote arbitrary data sets (which may be either base or derived sets).

2.4.1 Propagation into Unions

The first operator we are going to discuss is the union operator. The particular influence of this operator on the general structure of the delta set characterizations outlined above is reflected in the following

Proposition 1 *The delta sets of $P = Q \cup R$ satisfy the incremental equations*

$$P^+ = [Q^+ \setminus R^{\text{old}}] \cup [R^+ \setminus Q^{\text{old}}] \quad (18)$$

$$P^- = [Q^- \setminus R^{\text{new}}] \cup [R^- \setminus Q^{\text{new}}] \quad (19)$$

Proof:

$$\begin{aligned} (Q \cup R)^+ &= \\ [Q^{\text{new}} \cup R^{\text{new}}] \setminus [Q^{\text{old}} \cup R^{\text{old}}] &= \\ [(Q^{\text{new}} \setminus Q^{\text{old}}) \setminus R^{\text{old}}] \cup [(R^{\text{new}} \setminus R^{\text{old}}) \setminus Q^{\text{old}}] &= \\ [Q^+ \setminus R^{\text{old}}] \cup [R^+ \setminus Q^{\text{old}}] & \end{aligned}$$

The first equality follows from the definition of the delta sets of a Boolean derivative (16), the second follows from the distribution lemma for the union operator (12), whereas the third equality follows from the definition of the delta sets for Q and R , resp., i.e., from (7). The characterization of P^- can be proved in an analogous way:

$$\begin{aligned} (Q \cup R)^- &= \\ [Q^{\text{old}} \cup R^{\text{old}}] \setminus [Q^{\text{new}} \cup R^{\text{new}}] &= \\ [(Q^{\text{old}} \setminus Q^{\text{new}}) \setminus R^{\text{new}}] \cup [(R^{\text{old}} \setminus R^{\text{new}}) \setminus Q^{\text{new}}] &= \\ [Q^- \setminus R^{\text{new}}] \cup [R^- \setminus Q^{\text{new}}] & \quad \square \end{aligned}$$

Intuitively, this proposition can be understood as follows:

1. A new element appears in the union of two sets, if it is inserted into (at least) *one* of the operands of the union, and if it was *not* present in the other operand before the change.
2. An element disappears from a union, if it disappears from (at least) *one* of the operands, and if it will *not* be contained in the new state of the other operand.

The case where an element is simultaneously inserted into *both* operands is covered by the above condition, because being inserted means not being present before according to the assumption about $^+$ -sets stated above. Similarly the case that an element is deleted from both operands at a time is explained.

2.4.2 Propagation into Intersections, Products and Joins

Next we consider intersections of Q and R . The equations for union and those for intersection differ in two respects:

1. The increments are not connected with their respective residues by means of the difference operator (as for unions), but by means of intersection.
2. The superscripts of the residues themselves are exchanged according to the duality principle: **new**-sets are replaced by **old**-sets, and vice versa.

Proposition 2 *The delta sets of $P = Q \cap R$ satisfy the incremental equations*

$$P^+ = [Q^+ \cap R^{\text{new}}] \cup [R^+ \cap Q^{\text{new}}] \quad (20)$$

$$P^- = [Q^- \cap R^{\text{old}}] \cup [R^- \cap Q^{\text{old}}] \quad (21)$$

Proof:

$$\begin{aligned} (Q \cap R)^+ &= \\ [Q^{\text{new}} \cap R^{\text{new}}] \setminus [Q^{\text{old}} \cap R^{\text{old}}] &= \\ [(Q^{\text{new}} \setminus Q^{\text{old}}) \cap R^{\text{new}}] \cup [(R^{\text{new}} \setminus R^{\text{old}}) \cap Q^{\text{new}}] &= \\ [Q^+ \cap R^{\text{new}}] \cup [R^+ \cap Q^{\text{new}}] &\quad \square \end{aligned}$$

The first equality again follows from the definition of the delta sets of a Boolean derivative (16), the second follows from the distribution lemma for the intersection operator (14), and again the third equality follows from the definition of the delta sets for Q and R , resp., i.e., from (7). As for unions, the characterization of the negative delta set can be proved in an analogous way, exploiting the duality of **new** and **old**.

Due to the characterization of new and old state in terms of preserved elements and delta sets (10) and (11) we can eliminate a common subterm occurring when evaluating the expressions characterizing the delta sets for intersections given by the previous proposition. The optimized version is:

Lemma 5 *The delta sets of $P = Q \cap R$ can alternatively be characterized by*

$$P^+ = [Q^+ \cap R^0] \cup [R^+ \cap Q^+] \cup [R^+ \cap Q^0] \quad (22)$$

$$P^- = [Q^- \cap R^0] \cup [R^- \cap Q^-] \cup [R^- \cap Q^0] \quad (23)$$

Again we can rephrase these equations in natural language in order to make their essence more comprehensible:

1. A new element appears in the intersection of two sets, if it is inserted into the one and has "survived" the change in the other operand (in which it already was contained before), or if it has been inserted into both component sets.
2. An element disappears from an intersection, if it disappears from at least one of the operands, and if it has been present in the other operand before the change as well.

As the lemma stating the way how difference can be distributed over intersection applies to the related operators product and join as well, the last proposition can be formulated and proved analogously for each of these operators, too. We omit an explicit proposition and proof for \times and \bowtie .

2.4.3 Propagation into Differences

The difference operator itself is interesting again, as its distribution behaviour is significantly different from that of the other operators.

Proposition 3 *The delta sets of $P = Q \setminus R$ satisfy the incremental equations*

$$P^+ = [Q^+ \setminus R^{\text{new}}] \cup [R^- \cap Q^{\text{new}}] \quad (24)$$

$$P^- = [Q^- \setminus R^{\text{old}}] \cup [R^+ \cap Q^{\text{old}}] \quad (25)$$

Proof:

$$\begin{aligned} (Q \setminus R)^+ &= \\ [Q^{\text{new}} \setminus R^{\text{new}}] \setminus [Q^{\text{old}} \setminus R^{\text{old}}] &= \\ [(Q^{\text{new}} \setminus Q^{\text{old}}) \setminus R^{\text{new}}] \cup [(R^{\text{old}} \setminus R^{\text{new}}) \cap Q^{\text{new}}] &= \\ [Q^+ \setminus R^{\text{new}}] \cup [R^+ \cap Q^{\text{new}}] & \end{aligned}$$

and

$$\begin{aligned} (Q \setminus R)^- &= \\ [Q^{\text{old}} \setminus R^{\text{old}}] \setminus [Q^{\text{new}} \setminus R^{\text{new}}] &= \\ [(Q^{\text{old}} \setminus Q^{\text{new}}) \setminus R^{\text{old}}] \cup [(R^{\text{new}} \setminus R^{\text{old}}) \cap Q^{\text{old}}] &= \\ [Q^+ \setminus R^{\text{old}}] \cup [R^- \cap Q^{\text{old}}] & \quad \square \end{aligned}$$

While the residues have to be “evaluated” in the same states as for intersections, the connecting operators in between increments and residues are different from both, the union and the intersection case. In addition the effect of the difference operator becomes visible within the increments, too. Deletions from R turn into insertions into the difference, and insertions into R cause deletions from the difference. Rephrased informally, this proposition expresses the following “laws” of change propagation into differences (already addressed in the motivating example):

1. A new element is introduced into the difference of Q and R , if it is either inserted into Q and will not be in R after the change, or if it is deleted from R and will be in Q after the change.
2. An old element disappears from the difference, if it has either been deleted from Q and has not been in R before, or if it has been inserted into R and has been in Q before.

2.4.4 Propagation into Projections

The only unary Boolean operator we are considering in this paper is the projection operator. Increments of projections can be computed according to

Proposition 4 *The delta sets of $P = \Pi_i Q$ satisfy the incremental equations*

$$P^+ = \Pi_i Q^+ \setminus \Pi_i Q^{\text{old}} \quad (26)$$

$$P^- = \Pi_i Q^- \setminus \Pi_i Q^{\text{new}} \quad (27)$$

Proof:

$$\begin{aligned} (\Pi_i Q)^+ &= \\ (\Pi_i Q)^{\text{new}} \setminus (\Pi_i Q)^{\text{old}} &= \\ (\Pi_i Q^0 \cup \Pi_i Q^+) \setminus \Pi_i Q^{\text{old}} &= \\ (\Pi_i Q^0 \setminus \Pi_i Q^{\text{old}}) \cup (\Pi_i Q^+ \setminus \Pi_i Q^{\text{old}}) &= \\ \Pi_i Q^+ \setminus \Pi_i Q^{\text{old}} &\quad \square \end{aligned}$$

Apart from the definitions of the increments, the following lemmata have been used within this proof:

- The second step makes use of the incremental definition of the new state of a set in terms of positive increment and of preserved elements (10).
- The third step can be performed like this because of the distribution lemma for difference over a single union (13)
- The fourth (and most important) step of the proof is based on the fact that the expression $\Pi_i Q^0 \setminus \Pi_i Q^{\text{old}}$ denotes the empty set. This is due to (2) and (9) stating that all preserved elements have already been in the old state of a set.

The proof for the negative delta set can be performed in a similar way, making use of (11), (1) and (9). It should be noted, that the unary operator Π "behaves" in a similar way as the binary ones did, in that the increment of its operand has to be combined with a residue expression which is intended to eliminate induced insertions for which a different derivation already existed in the old state (or to eliminate induced deletions for which another derivation will exist in the new state, resp.).

3 The Rules of Change

In order to translate the equations characterizing incremental propagation steps into a rule-based format, we first have to agree on a way of expressing Boolean operators in rule-based form. We don't do this in form of a general, formal definition, but for ease of presentation (and without loss of generality) we assume that we are dealing with unary relations in case of binary Boolean operators, and with binary relations in case of the relational algebra operators projection and join. It should be obvious how the rule format looks like for relations of higher arity. The fact that we are now dealing with a particular data model, the relational one, does not mean that the transformation of the algebraic characterizations into rules works for this data model only. A similar (though not quite as straightforward) reformulation in rule form is possible for other models with deductive rules as well. Boolean operators can be represented by means of deductive, relational rules as follows:

operator	equation	rule
projection	$P = \pi_1(Q)$	$p(X) \Leftarrow q(X, Y)$
union	$P = Q \cup R$	$p(X) \Leftarrow q(X)$ $p(X) \Leftarrow r(X)$
intersection	$P = Q \cap R$	$p(X) \Leftarrow q(X), r(X)$
difference	$P = Q \setminus R$	$p(X) \Leftarrow q(X), \text{not } r(X)$
product	$P = Q \times R$	$p(X, Y) \Leftarrow q(X), r(Y)$
join	$P = Q \bowtie R$	$p(X, Y, Z) \Leftarrow q(X, Z), r(Z, Y)$

3.1 Incremental Rules for Delta Relations

Each incremental characterization of a delta set is now turned into a set of rules defining delta relations, derived from the relation names p , q , and r by means of superscripts $+$, $-$ in exactly the same way as we did for the corresponding delta sets. The new and old versions of the respective relations will be denoted by meta predicates superscribed by **new** and **old**, preserved elements by 0 . We will address the question how to realize these predicates later. Now we are in a position to straightforwardly encode the equations for update propagation in the algebraic case into rules defining delta relations:

- projection:

$$\boxed{p(X) \Leftarrow q(X, Y)}$$

$$p^+(X) \Leftarrow q^+(X, Y), \text{not } q^{\text{old}}(X, Y_1)$$

$$p^-(X) \Leftarrow q^-(X, Y), \text{not } q^{\text{new}}(X, Y_1)$$

- union:

$$\boxed{p(X) \Leftarrow q(X) \\ p(X) \Leftarrow r(X)}$$

$$p^+(X) \Leftarrow q^+(X), \text{ not } r^{\text{old}}(X) \\ p^+(X) \Leftarrow r^+(X), \text{ not } q^{\text{old}}(X)$$

$$p^-(X) \Leftarrow q^-(X), \text{ not } r^{\text{new}}(X) \\ p^-(X) \Leftarrow r^-(X), \text{ not } q^{\text{new}}(X)$$

- intersection:

$$\boxed{p(X) \Leftarrow q(X), r(X)}$$

$$p^+(X) \Leftarrow q^+(X), r^0(X) \\ p^+(X) \Leftarrow r^+(X), q^+(X) \\ p^+(X) \Leftarrow r^+(X), q^0(X)$$

$$p^-(X) \Leftarrow q^-(X), r^0(X) \\ p^-(X) \Leftarrow r^-(X), q^-(X) \\ p^-(X) \Leftarrow r^-(X), q^0(X)$$

- difference:

$$\boxed{p(X) \Leftarrow q(X), \text{ not } r(X)}$$

$$p^+(X) \Leftarrow q^+(X), \text{ not } r^{\text{new}}(X) \\ p^+(X) \Leftarrow r^-(X), q^{\text{new}}(X)$$

$$p^-(X) \Leftarrow q^-(X), \text{ not } r^{\text{old}}(X) \\ p^-(X) \Leftarrow r^-(X), q^{\text{old}}(X)$$

- product:

$$\boxed{p(X,Y) \Leftarrow q(X), r(Y)}$$

$$p^+(X,Y) \Leftarrow q^+(X), r^0(Y) \\ p^+(X,Y) \Leftarrow r^+(Y), q^+(X) \\ p^+(X,Y) \Leftarrow r^+(Y), q^0(X)$$

$$p^-(X,Y) \Leftarrow q^-(X), r^0(Y) \\ p^-(X,Y) \Leftarrow r^-(Y), q^-(X) \\ p^-(X,Y) \Leftarrow r^-(Y), q^0(X)$$

- join:

$$\boxed{p(X,Y,Z) \Leftarrow q(X,Z), r(Z,Y)}$$

$$\begin{aligned} p^+(X, Y, Z) &\Leftarrow q^+(X, Y), r^0(Z, Y) \\ p^+(X, Y, Z) &\Leftarrow r^+(Z, Y), q^+(X, Z) \\ p^+(X, Y, Z) &\Leftarrow r^+(Z, Y), q^0(X, Z) \end{aligned}$$

$$\begin{aligned} p^-(X, Y, Z) &\Leftarrow q^-(X, Z), r^0(Z, Y) \\ p^-(X, Y, Z) &\Leftarrow r^-(Z, Y), q^-(X, Z) \\ p^-(X, Y, Z) &\Leftarrow r^-(Z, Y), q^0(X, Z) \end{aligned}$$

A direct comparison of these “rules of change” with the propositions characterizing deltas in the previous section should be sufficient for exhibiting their relationships. In the general case, deductive rules are not necessarily in a format which directly corresponds to one of the Boolean operators. However, it is possible to rewrite any rule set into an equivalent one consisting only of rules that are in “algebra format”. By unfolding complex rule bodies and replacing subexpressions by new, auxiliary predicates such a transformation can be achieved very easily. Delta rules for the unfolded source rules can then be directly derived from the rule schemata listed above.

We would like to demonstrate this approach by means of two example rules defining links, i.e., paths of length 1 or 2, in a directed graph (where **edge** is assumed to be a base relation):

$$\begin{aligned} \text{link}(X, Y) &\Leftarrow \text{edge}(X, Y) \\ \text{link}(X, Y) &\Leftarrow \text{edge}(X, Z), \text{edge}(Z, Y) \end{aligned}$$

In order to correctly express the corresponding delta rules in *Datalog^{neg}* syntax and in order to arrive at a more compact formulation of the delta rules, we introduce auxiliary predicates **link1** and **link2**, reflecting the fact that the second rule is composed of a join *and* a projection, i.e., we are working with the four rules

$$\begin{aligned} \text{link}(X, Y) &\Leftarrow \text{edge}(X, Y) \\ \text{link}(X, Y) &\Leftarrow \text{link1}(X, Y) \\ \text{link1}(X, Y) &\Leftarrow \text{link2}(X, Y, Z) \\ \text{link2}(X, Y, Z) &\Leftarrow \text{edge}(X, Z), \text{edge}(Z, Y) \end{aligned}$$

The delta rules for **link1** are those for a projection predicate

$$\begin{aligned} \text{link1}^+(X, Y) &\Leftarrow \text{link2}^+(X, Y, Z), \text{not link2}^{\text{old}}(X, Y, Z_1) \\ \text{link1}^-(X, Y) &\Leftarrow \text{link2}^-(X, Y, Z), \text{not link2}^{\text{new}}(X, Y, Z_1) \end{aligned}$$

The delta rules for **link2** are those for a join predicate

$$\begin{aligned} \text{link2}^+(X, Y, Z) &\Leftarrow \text{edge}^+(X, Z), \text{edge}^0(Z, Y) \\ \text{link2}^+(X, Y, Z) &\Leftarrow \text{edge}^+(Z, Y), \text{edge}^+(X, Z) \\ \text{link2}^+(X, Y, Z) &\Leftarrow \text{edge}^+(Z, Y), \text{edge}^0(X, Z) \end{aligned}$$

$$\begin{aligned} \text{link2}^-(X, Y, Z) &\Leftarrow \text{edge}^-(X, Z), \text{edge}^0(Z, Y) \\ \text{link2}^-(X, Y, Z) &\Leftarrow \text{edge}^-(Z, Y), \text{edge}^-(X, Z) \\ \text{link2}^-(X, Y, Z) &\Leftarrow \text{edge}^-(Z, Y), \text{edge}^0(X, Z) \end{aligned}$$

Finally, the delta rules obtained for **link** are rules for a union predicate

$$\begin{aligned} \text{link}^+(X, Y) &\Leftarrow \text{edge}^+(X, Y), \text{not link1}^{\text{old}}(X, Y) \\ \text{link}^+(X, Y) &\Leftarrow \text{link1}^+(X, Y), \text{not edge}^{\text{old}}(X, Y) \\ \\ \text{link}^-(X, Y) &\Leftarrow \text{edge}^-(X, Y), \text{not link1}^{\text{new}}(X, Y) \\ \text{link}^-(X, Y) &\Leftarrow \text{link1}^-(X, Y), \text{not edge}^{\text{new}}(X, Y) \end{aligned}$$

In addition we need rules defining evaluation over the new state (assuming that a direct evaluation of a literal corresponds to an evaluation over the old state). This can be achieved by turning the incremental definition of new states of data sets (10) into deductive rules as well. For the relations **edge** such rules are as follows:

$$\begin{aligned} \text{edge}^{\text{new}}(X, Y) &\Leftarrow \text{edge}^0(X, Y) \\ \text{edge}^{\text{new}}(X, Y) &\Leftarrow \text{edge}^+(X, Y) \end{aligned}$$

This definition in turn refers to the set of preserved elements, which can be defined via a rule, too, making use of (9):

$$\text{edge}^0(X, Y) \Leftarrow \text{edge}^{\text{old}}(X, Y), \text{not edge}^-(X, Y)$$

In an analogous manner, new and preserved versions of **link1** and **link2** can be obtained.

In principle, we are now able to apply this rather big set of rules to a particular database state and a particular change by simply materializing the data that can be derived by means of these rules. However, a first obstacle arising is that by doing so we would not just materialize induced changes (i.e., the extension of the delta relations), but - due to the need to define new state and preserved elements - we would also materialize the entire extension of **edge**, **link1** and **link2** in the changed state. Of course, this is not what we want to do, because it would completely ruin the effort invested in view of computing increments only! In particular, we would have to materialize all those facts that have been in the **edge** relation and did not change two times: once in **edge**^{new} and once in **edge**⁰. The same applies to the two auxiliary relations, too, by the way.

A way out consists in not defining ^{new}- and ⁰-predicates by means of rules, but to fold the respective definitions into the delta rules where required, thus eliminating the new and preserved predicates syntactically. The advantage would be to avoid useless materialization of new and preserved facts altogether and to reach truly incremental delta rules, referring to increments and to the old state only. Doing so leads to the following rewritten set of delta rules:

$$\begin{aligned} \text{link1}^+(X, Y) &\Leftarrow \text{link2}^+(X, Y, Z), \text{not link2}^{\text{old}}(X, Y, Z_1) \\ \text{link1}^-(X, Y) &\Leftarrow \text{link2}^-(X, Y, Z), \text{not link2}^+(X, Y, Z_1)^1 \\ \\ \text{link2}^+(X, Y, Z) &\Leftarrow \text{edge}^+(X, Z), \text{edge}^{\text{old}}(Z, Y), \text{not edge}^-(Z, Y) \end{aligned}$$

¹The second rule resulting from folding the definition of $\text{link2}^{\text{new}}(X, Y, Z_1)$ is subsumed by the first one.

$\text{link2}^+(\text{X}, \text{Y}, \text{Z}) \Leftarrow \text{edge}^+(\text{Z}, \text{Y}), \text{edge}^+(\text{X}, \text{Z})$
 $\text{link2}^+(\text{X}, \text{Y}, \text{Z}) \Leftarrow \text{edge}^+(\text{Z}, \text{Y}), \text{edge}^{\text{old}}(\text{X}, \text{Z}), \text{not edge}^-(\text{X}, \text{Z})$

$\text{link2}^-(\text{X}, \text{Y}, \text{Z}) \Leftarrow \text{edge}^-(\text{X}, \text{Z}), \text{edge}^{\text{old}}(\text{Z}, \text{Y}), \text{not edge}^-(\text{Z}, \text{Y})$
 $\text{link2}^-(\text{X}, \text{Y}, \text{Z}) \Leftarrow \text{edge}^-(\text{Z}, \text{Y}), \text{edge}^-(\text{X}, \text{Z})$
 $\text{link2}^-(\text{X}, \text{Y}, \text{Z}) \Leftarrow \text{edge}^-(\text{Z}, \text{Y}), \text{edge}^{\text{old}}(\text{X}, \text{Z}), \text{not edge}^-(\text{X}, \text{Z})$

$\text{link}^+(\text{X}, \text{Y}) \Leftarrow \text{edge}^+(\text{X}, \text{Y}), \text{not link1}^{\text{old}}(\text{X}, \text{Y})$
 $\text{link}^+(\text{X}, \text{Y}) \Leftarrow \text{link1}^+(\text{X}, \text{Y}), \text{not edge}^{\text{old}}(\text{X}, \text{Y})$

$\text{link}^-(\text{X}, \text{Y}) \Leftarrow \text{edge}^-(\text{X}, \text{Y}), \text{not link1}^{\text{old}}(\text{X}, \text{Y}), \text{not link1}^+(\text{X}, \text{Y})$
 $\text{link}^-(\text{X}, \text{Y}) \Leftarrow \text{edge}^-(\text{X}, \text{Y}), \text{link1}^-(\text{X}, \text{Y}), \text{not link1}^+(\text{X}, \text{Y})$
 $\text{link}^-(\text{X}, \text{Y}) \Leftarrow \text{link1}^-(\text{X}, \text{Y}), \text{not edge}^{\text{old}}(\text{X}, \text{Y}), \text{not edge}^+(\text{X}, \text{Y})$
 $\text{link}^-(\text{X}, \text{Y}) \Leftarrow \text{link1}^-(\text{X}, \text{Y}), \text{edge}^-(\text{X}, \text{Y}), \text{not edge}^+(\text{X}, \text{Y})$

3.2 Propagating Updates by Fixpoint Computation

After all these lengthy transformations we are in a situation where the delta rules - in proper incremental form - can finally be "fired". For this purpose let us consider a very small and simple database state, consisting of just three edges:

$\text{edge}(1, 2) \qquad \text{edge}(2, 3) \qquad \text{edge}(3, 4).$

The links derivable from these edges are

$\text{link}(1, 2) \qquad \text{link}(2, 3) \qquad \text{link}(3, 4)$
 $\text{link}(1, 3) \qquad \text{link}(2, 4).$

Let us change this graph by removing the last edge and by adding an edge which goes back from 4 to 3, i.e., let the base data change be represented by the following deltas:

$\text{edge}^-(3, 4) \qquad \text{edge}^+(4, 3)$

Which are the changes induced to the link relation? If inspecting the above set of delta rules, we can easily discover that it is non-recursive and thus stratifiable. Fixpoint computation applied to these rules, to the old extension of **edge** and **link** as well as to the **edge** increments proceeds as follows. In the first iteration, the delta fact $\text{edge}^+(4, 3)$ can be propagated into the three delta rules defining link2^+ as well as into the first delta rule defining link^+ . The link2^+ -rules do not fire, but the link^+ one does, yielding

$\text{link}^+(4, 3).$

Similarly the delta fact $\text{edge}^-(3, 4)$ leads to the derivation of

$\text{link}^-(3, 4) \qquad \text{link2}^-(2, 4, 3).$

These three facts are the only ones that can be derived during the first iteration. In a second iteration, neither of the two newly derived **link**-deltas can be propagated any further. The link2^- -fact, however, leads to a materialization of

$\text{link1}^-(2, 4).$

In a third iteration this fact gives rise to the materialization of

$\text{link}^-(2, 4)$.

In a fourth (and final) iteration no new propagation steps can be performed anymore. Thus a fixpoint has been reached and the induced updates on the derived relation **link** which have been computed are

$\text{link}^+(4, 3)$ $\text{link}^-(3, 4)$ $\text{link}^-(2, 4)$

as could be expected.

This example doesn't prove anything, but it simply illustrates in a rather plausible way that - at least for this example - delta rules compute induced changes correctly if driven by (semi-)naive fixpoint iteration. Without being able (and willing) to prove that this applies to any non-recursive rule set in this paper, we hope that the reader is at least partially convinced that this result can readily be claimed. A brief look at the general format of the delta rules as given in the previous paragraph will be sufficient for understanding that these rules *do not introduce recursion* if the respective source rules did not contain any recursion.

The link example has been chosen because it is very close to one of the most well-known and most simple recursive rule sets, namely that for transitive closure. By replacing the predicate **link** by the predicate **path** and by making the second rule recursive, we can generalize the definition to paths of arbitrary links. It should be obvious to understand that delta rules can be obtained in a similar manner as demonstrated in detail above. The result of doing so is the following set of delta rules:

$\text{path1}^+(X, Y) \Leftarrow \text{path2}^+(X, Y, Z_1), \text{ not path2}^{\text{old}}(X, Y, Z)$

$\text{path1}^-(X, Y) \Leftarrow \text{path2}^-(X, Y, Z_1), \text{ not path2}^+(X, Y, Z)$

$\text{path2}^+(X, Y, Z) \Leftarrow \text{edge}^+(X, Z), \text{ path}^{\text{old}}(Z, Y), \text{ not path}^-(Z, Y)$

$\text{path2}^+(X, Y, Z) \Leftarrow \text{path}^+(Z, Y), \text{ edge}^+(X, Z)$

$\text{path2}^+(X, Y, Z) \Leftarrow \text{path}^+(Z, Y), \text{ edge}^{\text{old}}(X, Z), \text{ not edge}^-(X, Z)$

$\text{path2}^-(X, Y, Z) \Leftarrow \text{edge}^-(X, Z), \text{ path}^{\text{old}}(Z, Y), \text{ not path}^-(Z, Y)$

$\text{path2}^-(X, Y, Z) \Leftarrow \text{path}^-(Z, Y), \text{ edge}^-(X, Z)$

$\text{path2}^-(X, Y, Z) \Leftarrow \text{path}^-(Z, Y), \text{ edge}^{\text{old}}(X, Z), \text{ not edge}^-(X, Z)$

$\text{path}^+(X, Y) \Leftarrow \text{edge}^+(X, Y), \text{ not path1}^{\text{old}}(X, Y)$

$\text{path}^+(X, Y) \Leftarrow \text{path1}^+(X, Y), \text{ not edge}^{\text{old}}(X, Y)$

$\text{path}^-(X, Y) \Leftarrow \text{edge}^-(X, Y), \text{ not path1}^{\text{old}}(X, Y), \text{ not path1}^+(X, Y)$

$\text{path}^-(X, Y) \Leftarrow \text{edge}^-(X, Y), \text{ path1}^-(X, Y), \text{ not path1}^+(X, Y)$

$\text{path}^-(X, Y) \Leftarrow \text{path1}^-(X, Y), \text{ not edge}^{\text{old}}(X, Y), \text{ not edge}^+(X, Y)$

$\text{path}^-(X, Y) \Leftarrow \text{path1}^-(X, Y), \text{ edge}^-(X, Y), \text{ not edge}^+(X, Y)$

If looking at the dependency graph of this set of delta rules, we will discover that the following cycle has been introduced (due to the exchange of the second **edge** literal in the second **link** rule by a recursive **path** literal in the second **path** rule):

$\text{path}^- \rightarrow \text{path2}^+ \rightarrow \text{path1}^- \rightarrow \text{path}^-$.

The first link in this cycle, namely

$$\mathbf{path}^- \rightarrow \mathbf{path2}^+$$

is a negative one, as it comes from the delta rule

$$\mathbf{path2}^+(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) \Leftarrow \mathbf{edge}^+(\mathbf{X}, \mathbf{Z}), \mathbf{path}^{\text{old}}(\mathbf{Z}, \mathbf{Y}), \text{ not } \mathbf{path}^-(\mathbf{Z}, \mathbf{Y}).$$

The occurrence of any cycle involving recursion in the dependency graph of a rule set means that the respective rule set is unstratifiable and thus cannot be properly handled by means of fixpoint iteration, even not if an iterated fixpoint operator is applied! Intuitively this means that we cannot compute induced insertions for **path** without knowing all the induced deletions, and that we cannot compute induced deletions without knowing at least those induced insertions resulting from the recursive rule (represented by **path2**).

Depending on the particular change to be performed, the combination of facts and rules may nevertheless turn out to be at least locally stratifiable (which means that in such cases iterated fixpoint iteration might compute the correct result). However, in our example even this way out doesn't help, because we have - viciously - chosen the update in such a way that the insertion of **edge(4,3)** seems to introduce new paths (namely **path(4,4)** and **path(3,3)**), but the simultaneous deletion of **edge(3,4)** invalidates these induced insertions again. This invalidation could only be detected if deletions induced on **path** were known while propagating induced insertions. Because of the cyclic, negative dependency mentioned exactly this is not possible, which means that the two pretended induced insertions will erroneously be "taken for real" by fixpoint iteration over the unstratifiable rule set.

This (counter-)example *does* prove something, namely that a simple fixpoint materialization of derived deltas according to the fully incremental delta rules is not always feasible. We don't have the space in this paper to elaborate on the solutions to this problem which have been proposed in the literature, but will only give hints to relevant references. One possible way out consists in evaluating those literals referring to the new state in the original version of our delta rules by different means at least for recursive predicates. "Different means" can, e.g., be a top-down evaluation of a meta-rule specification of evaluation over the new state as has been proposed in [BDM88]. In our example this could mean that new **path** facts can be derived by applying the rules

$$\begin{aligned} \mathbf{path}^{\text{new}}(\mathbf{X}, \mathbf{Y}) &\Leftarrow \mathbf{edge}^{\text{new}}(\mathbf{X}, \mathbf{Y}) \\ \mathbf{path}^{\text{new}}(\mathbf{X}, \mathbf{Y}) &\Leftarrow \mathbf{edge}^{\text{new}}(\mathbf{X}, \mathbf{Z}), \mathbf{path}^{\text{new}}(\mathbf{Z}, \mathbf{Y}) \end{aligned}$$

in a "top-down" manner during fixpoint iteration. Such a top-down component may itself be implemented by means of fixpoint computations again, e.g., by transforming the above rules according to the magic set rewriting. Thus an interleaved process consisting of update propagation steps and magic set computations driven by a common fixpoint procedure will result.

Another way out could be to "break" the cycle in the dependency graph compromising stratifiability by slightly modifying the delta rules. By doing so we would accept to compute too many induced updates, e.g., to identify induced deletions that in fact reflect deleted derivation paths only, for which alternative derivations still remain accessible after the change. Such erroneous deltas could then be eliminated in a second phase to be performed after the respective change has been committed. Doing so has been proposed, e.g., in [GMS93] and [CW91]. However, it seems that such techniques are only applicable if update propagation is performed for incremental maintenance of materialized views.

4 Conclusion

In this paper we have addressed two related problems arising when aiming at a general purpose update propagation method. The first problem is to establish and prove general laws that could serve as a basis for any incremental definition of delta sets for derived concepts in a deductive database. We presented a new attempt to solve this problem, which we hope will be able to improve the insight in this matter.

The second problem addressed is how to properly implement update propagation by means of deductive rules defining delta sets. Based on our (algebraic) characterizations of delta sets, we have given a rule-based definition of delta relations, which can be straightforwardly derived from the algebraic equations. In addition we discussed the question whether a materialization of these delta relations can be achieved by directly applying a general-purpose fixpoint procedure to the delta rules. Whereas this seems to be feasible without problems in case of non-recursive rules, we have shown by means of an example that in the recursive case stratifiability of the delta rules is lost and thus materialization might derive incorrect deltas.

Despite the fact that we have been able to outline some of the possible solutions, the matter of how to adequately solve the problem of unstratifiable delta rules in case of recursive source rules definitely requires more investigations and a better, more thorough understanding than we are able to provide today. The contribution of this paper has been to point out the problem and to - hopefully - clarify a few of the issues arising from the attempt to arrive at a truly incremental, rule-based implementation of update propagation.

References

- [BDM88] F. Bry, H. Decker, and R. Manthey: "A Uniform Approach to Constraint Satisfaction and Constraint Satisfiability in Deductive Databases", in: Proc. EDBT'88, Venice, Febr. 1988, LNCS 303, 488-505
- [CGMD94] M. Celma, C. Garcíá, L. Mota, and H. Decker: "Comparing and Synthesizing Integrity Checking Methods for Deductive Databases", in: Proc. ICDE'94, Houston, Febr. 1994
- [CW90] S. Ceri and J. Widom: "Deriving production rules for constraint maintenance", in: Proc. VLDB'90, Brisbane, Aug. 1990, 566-577
- [CW91] S. Ceri and J. Widom: "Deriving production rules for incremental view maintenance", in: Proc. VLDB'91, Barcelona, Sept. 1991, 577-589
- [Dec86] H. Decker: "Integrity Enforcement on Deductive Databases", in: Proc. 1st Intern. Conf. on Expert Database Systems, Charleston, April 1986, 271-285
- [GM94] U. Griefahn and R. Manthey: "Update Propagation in Chimera, an Active DOOD Language", in: Proc. DAISD'94, Aiguablava, Sept. 1994 (to appear)
- [GMS93] A. Gupta, I.S. Mumick and V.S. Subrahmanian: "Maintaining Views Incrementally", in: Proc. ACM-SIGMOD'93, 157-166
- [Kue91] V. Küchenhoff: "On the Efficient Computation of the Difference Between Consecutive Database States", in: Proc. DOOD'91, Springer LNCS 566, 478-502,

- [Oli91] A. Olivé: "Integrity constraints checking in deductive databases", in: Proc. VLDB'91, Barcelona, Sept. 1991, 513-523
- [Qian91] X. Qian and G. Wiederhold: "Incremental Recomputation of Active Relational Expressions", in: IEEE Transactions on Knowledge and Data Engineering, Vol. 3:3, Sept. 1991, 337-341
- [RCB89] A. Rosenthal, S. Chakravarthy, and B. Blaustein: "Situation monitoring for active databases", in: Proc. VLDB'89, Amsterdam, 455-464
- [Ull88] J. Ullman: "Principles of Database and Knowledge-Base Systems", Vol. 1, Computer Science Press, 1988, chapter 8.1, 447-452
- [UO92] T. Urpí and A. Olivé: "A Method for Change Computation in Deductive Databases". in" Proc. VLDB'92, Vancouver, Aug. 1992, 225-237
- [VBK91] L. Vieille, P. Bayer, and V. Küchenhoff: "Integrity Checking and Materialized Views Handling by Update Propagation in the EKS-V1 System", ECRC Technical Report TR-KB-35, June 1991