Soft Stratification for Transformation-Based Approaches to Deductive Databases

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Andreas Behrend

aus Rostock

Bonn
2004

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Referent: Univ.-Prof. Dr. rer. nat. Rainer Manthey

2. Referent: Univ.-Prof. Dr. rer. nat. Armin B. Cremers

Tag der Promotion:

Meiner Familie

Abstract

The efficient evaluation of recursive views is a crucial issue in the research field of deductive databases. Results in this area are especially relevant for systems which will implement the new SQL:1999 standard, and hence will allow the definition of stratifiable recursive views. In particular, transformation-based solutions to query evaluation seem to be well-suited for extending existing relational databases as they are easy to implement and independent of other optimization methods such as index structures or algebraic manipulation techniques.

The application of transformation-based approaches, however, may lead to unstratifiable recursion which requires an elaborate, and consequently very expensive evaluation of these kinds of views in general. In this thesis, we present a solution to this problem by identifying the new class of so-called softly stratifiable views which allow for a more efficient evaluation than arbitrary unstratifiable views. This subclass of unstratifiable views is especially relevant as it covers views resulting from the rewriting of an originally stratifiable schema. We will show that the concept soft stratification can be used in various database services such as query evaluation and update propagation. Additionally, it can be employed as a basic evaluation technique for the efficient computation of the general well-founded semantics of unstratifiable schemata.

With respect to transformation-based approaches, we focus on the Magic Sets rewriting of (function-free) stratifiable databases as this method has evolved into a kind of standard in the field of query evaluation. The language Datalog is used as a syntactical basis because of its simplicity which makes it particularly well-suited for presenting transformation-based techniques. We will show that Kerisit's weak stratification approach for evaluating Magic Sets rewritten schemata may lead to a set of answers which is neither sound nor complete with respect to the well-founded model. This problem is cured by introducing the new soft consequence operator in combination with the concept soft stratification, instead. Afterwards, it will be shown that this approach is suited for solving the problem of existential query evaluation, too. To this end, we develop the so-called Existential Magic Sets rewriting which extends the Magic Sets transformation in such a way that the computation of alternative answers with respect to (derived) existential queries is avoided.

In case of update propagation, a novel deductive rule rewriting technique is developed incorporating the task of update propagation as well as Magic Sets optimizations into deductive propagation rules. To this end, Griefahn's structured update propagation approach is extended such that the resulting rule sets becomes less complicated and softly stratifiable. The results from both services, i.e., query evaluation and update propagation, are then combined for developing the new soft alternating fixpoint computation approach to determining the well-founded model of unstratifiable databases.

The algorithms and concepts presented in this thesis are developed by means of the abstract database language Datalog. The results can be almost directly transferred into the SQL context although additional language concepts of SQL such as Null values, multisets and aggregate functions have not been considered yet. However, it is our belief that the concept of soft stratification may already provide a realistic framework for extending the expressive power of relational database systems.

Acknowledgments

I like to thank my supervisor Rainer Manthey for his guidance and encouragements during the work on this thesis. I am also very grateful to Armin B. Cremers for his willingness to be co-referent. Particular thanks are due to Christian Dorau for insightful discussions which have been of immense value.

Contents

Chapter 1

Introduction

The notion of a deductive database has emerged during the 1970s in order to describe database systems capable of inferring new knowledge using rules. Within this research area three kinds of rule-based extensions for traditional databases have been intensively studied: active, deductive and normative rules. Nowadays, SQL databases widely use these extensions in form of triggers, views, and integrity constraints such that almost every commercial database system ought to be regarded as a deductive database system.

However, rule concepts have been implemented in commercial products (such as, e.g., Oracle or DB2) in a very limited way up to now. With respect to deductive rules, for example, the definition of general recursive views in SQL is still not possible. Recursion allows for computing the transitive closure of database relations and in general extends the expressive power of a relational database language such as prior SQL versions, QBE or QUEL. It plays an important role for path computations (e.g., in geographic information systems) or for traversing hierarchies of data (e.g., recursive bill of material queries). The implementation of recursive views has been avoided so far due to efficiency reasons as the evaluation of recursive queries poses several problems to classical database optimization techniques. However, the importance of this concept is widely accepted by now such that the new SQL:1999 standard has been extended by some kind of limited (stratifiable) recursive views. As database developers for commercial SQL-based systems try to implement the guidelines of the SQL standard as far as possible, efficient methods for evaluating recursive views are needed which are suitable for extending existing relational database systems.

In the field of deductive databases, a considerable amount of research has been devoted to the efficient evaluation of recursive views. Results in this area are usually presented in the database language Datalog and can be divided into top-down and bottom-up approaches. While SQL is based on a mixture of both, tuple-relational calculus and relational algebra, Datalog relies on domain-relational calculus. Syntactically, Datalog is very similar to Prolog, but it is based on a set-oriented bottom-up semantics in form of well-founded models. The reason for

using Datalog is its syntactic simplicity which makes it well-suited for presenting transformation-based techniques (as shown later on). However, the application of Datalog and the ongoing discussion about the question whether top-down or bottom-up approaches are more suited as a basic evaluation mechanism for recursive rules, have made it difficult to see how results from the research area of deductive databases can be used for extending the 'purely' top-down evaluation strategies in SQL.

Indeed, the transfer of 'bottom-up' approaches from a Datalog context to the SQL world is obviously possible in principle, though intricate in detail (cf. [MP94, Pie01]). With respect to the discussion about top-down and bottom-up approaches, in [Bry90b] it has been shown that both approaches are basically equivalent as top-down approaches with tabulation can be simulated by bottom-up ones and vice versa. As a matter of fact, even solutions to classical top-down problems like query evaluation can be significantly enhanced by incorporating bottom-up techniques, e.g., in dynamic query processing [Beh00]. On the other hand, typical bottom-up techniques for update propagation can be improved by incorporating top-down methods which has been shown in [Gri97] (cf. also Chapter 5). Thus, it can be concluded that the results in the area of deductive databases can provide relevant solutions to the problem of evaluating recursive views in SQL-based systems as well. In this thesis, we will substantiate this by providing solutions to the problem of query evaluation and update propagation with respect to recursively defined views. To this end, the new soft stratification approach is developed in Datalog for an efficient evaluation of transformation-based solutions to these problems which is well-suited for being transferred into the SQL context.

Transformation-based Approaches

Several proposals to the efficient evaluation of recursion in the database context have been made. The reason for developing this kind of specialized methods is that known algorithms from graph theory like Warshall's transitive closure or Dijkstra's shortest path algorithm are not appropriate for being directly implemented in a database system. This is due to the fact that evaluation techniques in databases ought to allow for a parallel computation of facts in a set-oriented way. In addition, they should be independent of other optimization techniques such as index structures or algebraic manipulation techniques which have led to the wide acceptance of relational database systems. Transformation-based approaches satisfy these requirements and additionally are particularly well-suited for extending existing relational database techniques.

The basic idea is to automatically transform a given database schema into a new one such that the evaluation of the rewritten schema simultaneously solves a certain database task with respect to the original schema. Extensive research into such rewriting techniques originated from the Magic Sets approach [BR86]

for query evaluation with respect to recursively defined relations. Since then, many similar as well as analogous approaches have been presented dealing with various kinds of database tasks such as update propagation, integrity checking, and view updating. As an example for the Magic Sets approach, consider the following Datalog rules for defining the relation `path` as the transitive closure of a base relation `edge`

$$\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{edge}(\texttt{X},\texttt{Y})$$
$$\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{edge}(\texttt{X},\texttt{Z}) \wedge \texttt{path}(\texttt{Z},\texttt{Y})$$

and the query $? - \texttt{path}(1,\texttt{Y})$, asking for all nodes reachable from node 1. According to the Magic Sets approach these rules are transformed into:

$$\begin{aligned}
\texttt{path}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_path}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{edge}(\texttt{X},\texttt{Y}) \\
\texttt{path}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_path}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{edge}(\texttt{X},\texttt{Z}) \wedge \texttt{path}(\texttt{Z},\texttt{Y}) \\
\texttt{m_path}_{\texttt{bf}}(\texttt{Z}) &\leftarrow \texttt{m_path}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{edge}(\texttt{X},\texttt{Z}).
\end{aligned}$$

The evaluation of the rewritten rules together with the transliterated query in form of a so-called magic seed fact $\texttt{m\_path}_{\texttt{bf}}(1)$ leads to the derivation of all possible answers with respect to the original query while avoiding the generation of irrelevant facts.

It is obvious that this kind of transformation-based approaches is well-suited for extending database systems as new algorithmic ideas are solely incorporated into the transformation process, leaving the actual database engine with its own optimizations techniques unchanged. In fact, rewriting techniques allow for implementing various database functionalities on the basis of one common inference mechanism. However, the application of transformation-based approaches with respect to stratifiable views may lead to unstratifiable recursion within the rewritten schemata which requires an elaborate, and consequently very expensive inference mechanism in general. This is the case for the kind of recursive views proposed by the new SQL:1999 standard too, as they cover the class of stratifiable views.

Goals

Transformation-based approaches can be used for extending existing relational databases but may require a very expensive inference mechanism, e.g. the alternating fixpoint operator by Van Gelder [vG89], if applied to an originally stratifiable schema. The Magic Sets method for query evaluation represents such a rewriting approach which may result in an unstratifiable rule set. As this method has evolved into a kind of standard in the field of query processing with respect to recursively defined views, a solution to this problem is needed which avoids the costly application of too general evaluation techniques like the alternating fixpoint.

Goal 1: Developing a new inference mechanism which is well-suited for implementing query evaluation based on the Magic Sets approach.

Query optimization represents a typical top-down problem which plays an important role in other database services as well. For instance, in [Gri97] it has been shown that such optimization techniques can be also used for improving bottom-up evaluation methods to update propagation which led to the so-called structured update propagation approach. However, the applied transformation technique may again (as for Magic Sets) lead to unstratifiable rules such that structured update propagation is partly based on computing alternating fixpoints, too. Hence, we aim at developing a new transformation technique which allows the application of the same inference mechanism as for the efficient evaluation of Magic Sets transformed rules.

Goal 2: Improving the efficiency of update propagation in stratified databases on the basis of the Magic Sets approach.

Up till now, we have solely considered a certain subclass of unstratifiable rules which resulted from the transformation of originally stratifiable ones. However, in [Kol91] it has been shown that general unstratifiable rules are more expressive than stratifiable ones and that there are interesting queries which cannot be formulated by means of stratifiable rules. Thus, although the SQL:1999 standard does not include unstratifiable recursion yet, it seems to be worthwhile to consider this most general class of recursive views as well.

Several approaches to computing the well-founded model of arbitrary, i.e., possibly unstratifiable, deductive databases have been made among which the alternating fixpoint by Van Gelder [vG89] has become the most established one. This iterative method computes overestimations of facts considered to be definitely false in order to successively derive definitely true facts. However, in each iteration round superfluous calculations with respect to definitely true and definitely false facts are performed when applying the alternating fixpoint to unstratified relations. Hence, our last goal is to identify and eliminate these redundancies by using our results from the discussions above.

Goal 3: Improving efficiency of the alternating fixpoint approach for computing the well-founded model of general deductive databases.

Altogether, the main objective of this thesis is to improve existing transformation-based methods and to develop new ones for evaluating stratifiable as well as unstratifiable recursion. The results ought to provide a realistic framework of efficient evaluation techniques for extending existing relational database systems.

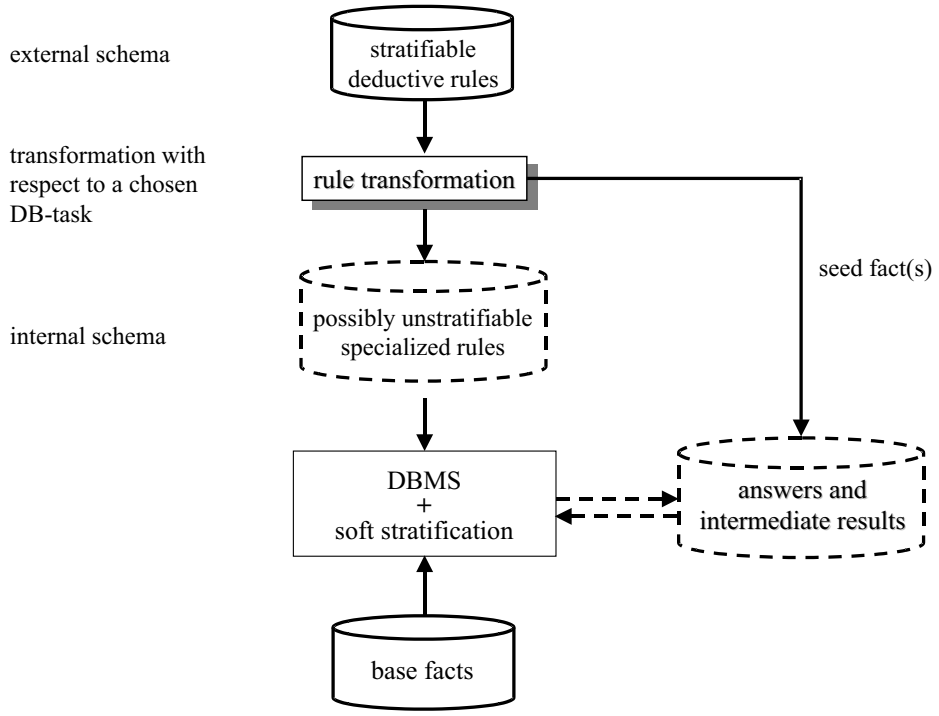


Figure 1.1: Architecture of a transformation-based deductive DBS

Approach

In order to realize the above goals, optimizations like Magic Sets or tasks like update propagation are incorporated into the deductive rules by means of suitable rule transformations such that all facts representing the result of the respective task will be automatically generated by an appropriate inference mechanism. For evaluating such rewritten and consequently possibly unstratifiable rule sets we propose the concept of soft stratification together with the soft consequence operator [Beh03] as a new efficient inference mechanism. This mechanism avoids any redundant generation of facts when evaluating unstratifiable rules by taking into account the specific reason for unstratifiability, i.e., the application of a transformation-based approach to an originally stratified rule set.

Figure 1.1 graphically illustrates our proposed architecture of a deductive database which uses transformation-based techniques for realizing its functionalities. To this end, an external schema consisting of user-defined deductive rules is distinguished from an internal one which results from the rewriting of the external schema in order to generate specialized rules with respect to a certain database task. The (temporary) internal schema together with corresponding auxiliary seed facts (e.g. propagation seeds when performing update propagation) are then applied to the inference component by which the DBMS of the considered re-

lational database system has been extended. The inference component is based on the soft stratification approach and derives all facts which are relevant for generating the result facts of the required service.

The considered inference component is based on soft stratification in order to handle stratifiable as well as a certain subclass of unstratifiable recursion. The soft stratification approach represents a fixpoint-based evaluation mechanism which is well-suited for extending existing relational DBMS components as it poses as little restrictions as possible to the evaluation of (transformed) views. We will show that our proposed rule transformations with respect to query evaluation and update propagation always lead to softly stratifiable rules such that this inference mechanism represents the appropriate evaluation technique. In addition, it can be used as a basic component for the efficient implementation of even more general inference mechanisms like the alternating fixpoint in order to handle general unstratifiable recursion, too.

Results

The main result of this thesis is a uniform approach to handling recursion in stratifiable databases on the basis of query evaluation and update propagation which is based on the soft stratification method. This approach is used to improve the alternating fixpoint method for evaluating general unstratifiable recursion as well. According to our goals established above, we have in particular obtained the following individual results:

> **Result 1:** We introduce a new bottom-up query evaluation method for stratified deductive databases based on the Magic Sets approach. We show that the application of the alternative weak stratification approach by Kerisit and Pugin [KP88] may lead to a set of answers which is neither sound nor complete with respect to the well-founded model of magic rewritten rules. This problem is cured by introducing the new concepts soft stratification and soft consequence operator instead.
>
> **Result 2:** On the basis of the soft stratification approach, a new solution to the problem of optimizing existential queries is presented. To this end, the so-called Existential Magic Sets rewriting is developed as an extension of the Magic Sets approach. The evaluation of such rewritten rules by using the soft stratification method partly avoids the redundant computations of alternative answer facts with respect to (derived) existential queries.
>
> **Result 3:** We introduce a new transformation-based approach to update propagation in stratifiable deductive databases which com-

bines the advantages of bottom-up and top-down propagation methods. Our soft update propagation method is based on a variant of the so-called Magic Updates transformation which itself is part of the structured update propagation approach proposed by Griefahn in [Gri97]. However, as the rewritten rules are potentially unstratifiable, this approach is based on the alternating fixpoint computation leading to an inefficient evaluation because the specific reason for unstratifiability is not taken into account. Therefore, we propose a less complex Magic Updates transformation resulting in a set of rules which is not only smaller but may in addition be efficiently evaluated using the soft stratification approach. Thus, less joins have to be performed and less facts are generated in comparison to the related structured update propagation approach.

Result 4: We present a new bottom-up algorithm for computing the well-founded model of general deductive databases. The drawback of repeated computation of facts from which Van Gelder's alternating fixpoint procedure is suffering is avoided by using the soft stratification as well as the soft update propagation approach. The resulting soft alternating fixpoint computation represents a generalization of the differential fixpoint computation well-known for stratifiable deductive databases.

Note that **Result 1** has been already published in [Beh03], while parts of **Result 4** have been presented in [Beh01]. All proposed transformation-based approaches have been developed in the context of Datalog. Soundness and completeness of the proposed methods have been shown with respect to an agreed declarative semantics of Datalog in form of well-founded models. With respect to complexity results, the most important properties of the methods discussed in this thesis are the number of generated facts and the number of iteration rounds needed for computing these facts. However, the improvement of efficiency by using our proposed methods will not be justified with new (lower) complexity bounds. This is due to the fact that for finite Herbrand universes and fixed rule sets, any of the discussed approaches requires time polynomial in the size of the Herbrand universe (cf. Section 2.1.2). Therefore, the improved efficiency will be shown by examples and by the following structural argument.

In general, for evaluating an unstratifiable negative literal it is not possible to determine all true conclusions and afterwards simply applying the negation as failure-principle [Cla78] for determining all true negative conclusions with respect to this literal. Instead, existing methods needed to overestimate the sets of true positive as well as true negative conclusions in order to successively derive the facts considered to be definitely true. Our proposed inference mechanism based on soft stratification, however, avoids any overestimations of facts such that less

facts are generated and a smaller number of iteration rounds is needed in the average case. Since in the worst case our proposed methods perform at most the same computations as the original approaches and are generally more flexible when applying further optimizations like algebraic manipulations, the claim of improved efficiency is justified.

Outline of the Thesis

Chapter 2 provides a collection of concepts in order to clarify the basis from which the investigations of this dissertation emerge. It introduces all relevant notions related to deductive databases as well as an update language allowing for the modification of extensional relations. Additionally, a model-based semantics for the different classes of semi-positive, stratifiable and general deductive databases is presented in a non-constructive way.

Based on this classification scheme, *Chapter 3* provides constructive methods for computing the semantics of a deductive database by means of fixpoint computations. To this end, we recall the known differential fixpoint, the iterated fixpoint, as well as the alternating fixpoint computation for determining the semantics of semi-positive, stratifiable, and general databases, respectively. In order to formalize the derivation of facts in this context, we define different consequence operators based on the immediate consequence operator by van Emden and Kowalski [vEK76]. Among them, the soft consequence operator is introduced which serves as the basic evaluation mechanism for softly stratifiable rules and related transformation-based approaches proposed in subsequent chapters. It is shown that this operator can already be used for implementing the differential and iterated fixpoint computation for evaluating stratifiable recursion.

In *Chapter 4* we consider the problem of query processing in stratifiable databases on the basis of the Magic Sets approach. This chapter can be divided into two parts: The aim of the first part is to provide a new inference mechanism for evaluating Magic Sets transformed rules. The second part then deals with the problem of existential query evaluation in this context.

First, we discuss the weak stratification approach by Kerisit and Pugin [KP88] as a potential evaluation technique for magic rewritten rules. We show that this method may lead to answers which are neither sound nor complete with respect to the total well-founded model of magic rules. As a solution to this problem, the new concept of soft stratification is introduced which, together with the soft consequence operator, provides the bottom-up inference method for evaluating this subclass of unstratifiable views. In a subsequent discussion, the efficiency of this approach is compared to other fixpoint query evaluation techniques.

In the second part, an improvement of the Magic Set approach is presented which incorporates the optimized evaluation of existential queries. To this end, the Existential Magic Sets transformation is introduced which allows for the speci-

fication of a subset of existential (derived) queries occurring during the evaluation of a Magic Sets rewritten rule set. It is shown that the evaluation of Existential Magic Sets rewritten rules using the soft consequence operator partly avoids the redundant computations of alternative answer facts with respect to such existential queries.

Chapter 5 shows how deductive rule rewriting can be used for implementing update propagation in stratifiable databases. To this end, we recall known transformation techniques for generating deductive propagation rules specifying the true changes in which the old database state differs from the new one after a base update has been applied. Afterwards, we adopt the idea of structured update propagation by incorporating Magic Sets optimizations into the proposed update propagation rules. To this end, a modified Magic Updates transformation [Gri97, Man94] is presented which always yields softly stratifiable rules such that the soft stratification approach can be used for their efficient evaluation. This in turn represents a solution to stratification problems occurring in related approaches like the structure update propagation method because the costly application of too general inference mechanisms can be avoided. The proposed Magic Updates transformation together with the soft stratification evaluation technique then represents our soft update propagation approach. After comparing it to the related structured update propagation method, its application to integrity checking and materialized view maintenance is briefly discussed.

Chapter 6 is concerned with optimizing the alternating fixpoint computation by using the results presented in previous chapters. After introducing the doubled program approach to implementing the alternating fixpoint, the general positive impacts of using update propagation rules for evaluating doubled programs is discussed. Subsequently, with the sequential consequence operator a simplified version of the soft consequence operator is presented for evaluating Magic Updates transformed rules in doubled programs. This chapter concludes with a comparison of our proposed soft alternating fixpoint method to the related approaches for the efficient evaluation of residual programs.

Related Work

In the sequel we will outline the global context of the work in this dissertation. To this end, we give a brief overview of related sub-areas of deductive rule research by referring to selected publications.

First, we describe related work on *query evaluation* and on *existential query optimization* with respect to stratified relations. Then we refer to publications dealing with *update propagation* in stratifiable deductive databases. Finally, we will provide a brief overview of methods for *computing the well-founded semantics* of general, i.e., possibly unstratifiable, databases.

Note that the following expositions aim at illustrating research that is pursued in the respective areas. Therefore, we will not particularly discuss the sources from which our own investigations emerge. Such publications are investigated in more detail in the chapters where they are referenced.

Query Evaluation

Query processing represents a problem which can be efficiently solved only by a top-down evaluation strategy. This is due to the fact that constants occurring in a query with respect to a derived relation need to be pushed down as far as possible to the underlying base relations in order to keep intermediate results small. As already mentioned above, however, it is possible to simulate a top-down evaluation strategy by a bottom-up inference mechanism. Therefore, we distinguish proposals to query evaluation in deductive databases with respect to the kind of inference mechanism they are based upon.

Top-down methods perform query evaluation in goal directed manner such that computation is naturally limited to relevant parts of the database only. Clark's SLDNF resolution [Cla78] represents one of the earliest top-down method which has the advantage of a goal-directed evaluation and an efficient stack-based memory management. Since SLDNF cannot guarantee termination in presence of recursion, and additionally may perform a lot of repeated computations of identical sub-goals, several extensions of SLD(NF) resolution with memoing have been proposed, including extension tables [DW86], OLDT resolution [TS86], QSQ [Vie88] and QRGT [Ull89] . The main idea is to keep a global table of sub-goals and their answers which have been computed. If a sub-goal is identical to or subsumed by a previous one, it is solved by solely using the answers already computed for the previous sub-goal. These techniques have been generalized to stratifiable recursion in [KT88, SI88]. The disadvantage of these 'pure' top-down solutions is that an expensive 'logic' control is needed in order to provide completeness and soundness in presence of recursion.

Bottom-up approaches avoid this drawback by simulating a top-down evaluation strategy using transformed deductive rules on the basis of a very simple materialization process. The most important transformation-based query evaluation methods result from the seminal proposals of the Magic Sets approach [BR86, BMSU86] and the related Alexander Method [RLK86] which have been independently developed. Based on these two approaches, several proposals have been made aiming at refinement and extension of the original methods. These include Generalized Magic Sets [BR91], Magic Templates [Ram91], Generalized Supplementary Magic Sets [BR91], Magic Counting [SZ87b], Generalized Magic Counting [BR91], Generalized Supplementary Magic Counting [BR91], Magic Conditions [MFPR96], Minimagic Sets [SZ87a], Envelopes [Sag90], SLD-Magic [Bra96] and Alexander Templates [Sek89].

In [KP88, BPRM91, Che93, KSS95, Mor93] the applicability of Magic Sets to stratifiable or even unstratifiable deductive databases is investigated. The weak stratification approach by Kerisit and Pugin [KP88] represents a fixpoint-based solution and is closely related to our proposed soft stratification method. However, the weak stratification method is neither complete nor sound as shown in Section 4.2.2. In contrast, the structured bottom-up method by Balbin et al. [BPRM91] represents a correct solution to evaluating Magic Sets rewritten rules in stratified databases. It uses a function for evaluating negative literals which recursively performs local fixpoint computations over the relevant portion of the Magic Sets transformed rules. The evaluation basically coincides with the one performed by the soft stratification approach. However, the disadvantage of their solution is that the nested fixpoint computations make it difficult or even impossible to employ further rule optimization techniques.

Existential Query Optimization

An existential query is a query which contains no free variables such that the generation of one answer fact is sufficient while all other derivations of the same fact are not needed and ought to be avoided. The problem of optimizing (derived) existential queries, however, has received little attention in the database community up till now, and there exists no general solution yet. The first approach by Ramakrishnan et al. [RBK88] suggested a solution based on pushing projections into recursive rules. However, in principle this cannot solve the general problem as the question of finding an equivalent schema with a different arrangement of projections is undecidable. Another branch of research has focussed on existential queries within a Magic Sets-transformed rule set, e.g. [NRSU89, Aze97, Beh00]. Naughton et al. [NRSU89] propose an optimization of the Magic Sets transformation such that the arity of the recursive answer predicates in the transformed rules is reduced. In [Aze97, Beh00] subsumption effects between magic sub-queries are used to avoid redundant computations. As an example, the query $? - \texttt{path}(\texttt{1}, \texttt{Y})$ subsumes the existential query $? - \texttt{path}(\texttt{1}, \texttt{4})$ such that the evaluation of the latter can be stopped after generating the corresponding sub-query fact $\texttt{m\_path}_{\texttt{bf}}(\texttt{1})$ with respect to the first query. However, these methods still do not represent a complete solution to existential query optimization because it is necessary to find more general sub-queries in order to avoid the redundant computation of existential queries.

Update Propagation

Methods for update propagation have been mainly studied in the context of Datalog (e.g. [Dec86, KSS87, LST87, BDM88, Küc91, Oli91, BMM91, UO92, GMS93, CW94, Man94, UO94, TO95, LL96, MT99, MT00]), relational algebra

(e.g. [QW91, Man94, GL95, CGL+96, CKL+97, BDD+98, DS00, SBLC00]), and SQL (e.g. [CW90, CW91]). Methods in Datalog can be divided into approaches based on a top-down or bottom-up evaluation strategy. A top-down evaluation for integrity checking is proposed by Olivé in [Oli91] where SLDNF resolution is used as basic inference mechanism. However, SLDNF resolution cannot guarantee termination for recursively defined predicates, and its tuple-oriented evaluation technique is not well-suited for the database context. Bottom-up methods either provide no goal-directed rule evaluation with respect to induced updates (e.g. [Küc91]) or suffer from stratification problems arising when transforming an original stratifiable schema (e.g. [Gri97, Man94]). Hence, for the latter approaches (for an overview cf. [Gri97]) the expensive application of more general evaluation techniques like the alternating fixpoint [vG93] is needed.

In general, approaches formulated in relational algebra or SQL are not capable of handling (non-linear) recursion, the latter usually based on transformed views or specialized triggers. Transformed SQL-views directly correspond to our proposed soft update propagation method for the non-recursive case. The application of triggers (e.g. production rules even for recursive relations in [CW94]), however, does not allow for the reuse of intermediate results obtained by querying the derivability and effectiveness tests. In [GL95] an algebraic approach to view maintenance is presented which is capable of handling duplicates but cannot be applied to general recursive views. For recursive views, [GMS93] proposes the "Delete and Rederive"-method which avoids the costly test of alternative derivations when computing induced deletions. However, this approach needs to compute overestimations of the tuples to be deleted, and additional pretests are necessary to check whether a view is affected by a given update [LS93].

The importance of integrating Magic Sets with traditional relational optimizations has been discussed already in [MP94]. The structured propagation method in [Gri97] represents a bottom-up approach for computing Magic Sets transformed propagation rules. However, as these rules are potentially unstratifiable, this approach is based on the alternating fixpoint computation [vG93] leading to an inefficient evaluation.

Computing the Well-founded Semantics

The well-founded semantics has been introduced by Van Gelder et al. in [vGRS88, vGRS91]. In contrast to the stable model semantics (which can yield more than one model or no model at all), the well-founded semantics always yields a unique model but is in general non-normal, i.e., there may be undefined atoms.

Approaches to the computation of well-founded models can be divided into methods using the alternating fixpoint (e.g. [vG93, KSS91, KSS95, SNV95]) and into those based on the residual program method (e.g. [Bry89, DK89, BD95, BZF96]). In [KSS91, KSS95], Kemp et al. propose a transformation-based ap-

proach to the efficient implementation of the alternating fixpoint based on the so-called doubled program rewriting. This approach will be discussed in more detail in Chapter 6 where we propose further improvements leading to our soft alternating fixpoint method.

The alternative residual program approach is based on so-called conditional facts, i.e., facts depending on a set of 'delayed' negative literals. They might be seen as ground rules solely consisting of negative body literals making negative dependencies within a deductive rule set explicit. Methods for efficiently evaluating residual programs have been suggested in [BZF96, BZF97, BDFZ01] where the authors propose a delayed generation and reduction of certain conditional facts. At the end of Chapter 6, we will show that the same optimization effects can be achieved in a much simpler way by our soft alternating fixpoint approach which additionally fits well with database context.

Chapter 2

Deductive Databases

This chapter introduces basic concepts of deductive databases. Section 2.1 is devoted to facts and rules. Queries and static integrity constraints are considered in Section 2.2 and 2.3, respectively, whereas Section 2.4 is concerned with updates. The syntax of each concept, based on the well-known database language Datalog, is presented first while the underlying semantics is considered afterwards. Note that the presentation of these concepts is partly based on [CGH94] and [Gri97].

2.1 Facts and Rules

Throughout this thesis, we assume that deductive databases consist of facts, deductive rules, and integrity constraints. In principle, every database model and every declarative query language can be used for formulating these concepts. We use Datalog as a syntactical basis, as this language has evolved into a kind of standard in the field of deductive databases. In contrast to SQL:1999 views, Datalog rules are mainly used because of their syntactic simplicity which makes them especially suited for transformation-based techniques. Another reason for the wide acceptance of Datalog is that it is based on an agreed declarative semantics in form of well-founded models.

2.1.1 Syntax

The syntax of Datalog is based on function-free Horn clauses. In the following we assume that a fixed alphabet is given including all symbols which may be used for constructing database clauses and update statements. Apart from connectives and punctuation symbols, we distinguish a universe of constants $U = \{a, b, c, \ldots\}$, a set of variables $\{X, Y, Z, \ldots\}$ and a set of predicate symbols $\{p, q, r, \ldots\}$. Each predicate symbol (or relation symbol) is associated with a certain arity $n \geq 0$ and defines an n-ary relation over U.

Definition 2.1 (Datalog Term) *A Datalog term is either a constant or a variable (i.e., we restrict ourselves to function-free terms).*

Definition 2.2 (Atomic Datalog Formula) *Let p be an n-ary predicate symbol and t_i ($i = 1, \ldots, n$ and $n \geq 0$) Datalog terms, then*

$$p(t_1, \ldots, t_n)$$

(or simply $p(\vec{t})$) is denoted atomic formula or atom. If $n = 0$, we write p instead of $p()$. An atom $p(t_1, \ldots, t_n)$ is ground, if every term t_i is a constant.

In the following, atoms will also be used for representing queries, and ground atoms are used for representing integrity constraints.

Definition 2.3 (Datalog Formula) *A Datalog formula is either*

1. *the propositional constant* `true`,
2. *an atomic formula (or positive literal) A,*
3. *a negated atomic formula (or negative literal) $\neg A$, or*
4. *a conjunction $L_1 \wedge \ldots \wedge L_n$ of literals where $n \geq 1$. A conjunction $L_1 \wedge \ldots \wedge L_n$ may also be considered as a set $\{L_1, \ldots, L_n\}$.*

If L is a literal (positive or negative), we use `pred`*(L) to refer to the predicate symbol occurring in L.*

In this thesis we exclusively deal with allowed (safe or range-restricted) facts, updates and rules, respectively. Several different definitions of allowedness have appeared in the literature [Nic82, Ull85, Cla78]. The following notions are used to define allowedness equivalent to Clark's original definition [Cla78] as this concept will be refined later when query evaluation is considered.

Definition 2.4 (Variable Occurrences) *For a formula W, the set of all variables occurring in W is denoted* `vars`*(W). If* `vars`*$(W) = \emptyset$, the formula W is called ground. By* `vars`$^-$ *we denote all variables solely occurring within negative literals, whereas* `vars`$^+$ *denotes variables occurring in at least one positive literal. For any atom A and any conjunction of literals $W \equiv L_1 \wedge \ldots \wedge L_n$ ($n \geq 1$) we define:*

- $\texttt{vars}^-(A) := \emptyset$ *and* $\texttt{vars}^+(A) := \texttt{vars}(A)$,
- $\texttt{vars}^-(\neg A) := \texttt{vars}(A)$ *and* $\texttt{vars}^+(\neg A) := \emptyset$,
- $\texttt{vars}^-(W) := \bigcup_{i=1,\ldots,n} \texttt{vars}^-(L_i) \setminus \bigcup_{i=1,\ldots,n} \texttt{vars}^+(L_i)$, *and*
  $\texttt{vars}^+(W) := \bigcup_{i=1,\ldots,n} \texttt{vars}^+(L_i)$.

The following notion of a database clause is used to formally introduce the notions fact and deductive rule subsequently.

Definition 2.5 (Database Clause, Fact, Rule) *A database clause is an expression of the form*

$$A \leftarrow W$$

where A is an atom and W is a formula. The atom A is called head and the formula W body of the database clause. If $W \equiv \texttt{true}$, then the body and the implication arrow can be omitted and A is called a fact. Otherwise, the clause is called a deductive rule.

If A is the head of a given database clause $C \equiv A \leftarrow W$, $\texttt{pred}(\mathrm{C})$ denotes the predicate symbol of A. For a set of clauses $\mathcal{C}$, $\texttt{pred}(\mathcal{C})$ is defined as $\bigcup_{c \in \mathcal{C}} \texttt{pred}(c)$.

As already mentioned above, the concept *allowedness* plays an important role in the context of deductive databases. A database clause is allowed if all variables occurring in the rule's head do also occur in at least one positive literal of the rule's body. In addition, there must be no variable solely occurring in negative body literals.

Definition 2.6 (Allowed Database Clause) *A database clause $A \leftarrow W$ is called allowed, if the following conditions hold:*

$$\texttt{vars}(A) \subseteq \texttt{vars}^{+}(W) \quad \textit{and} \quad \texttt{vars}^{-}(W) = \emptyset.$$

Note that this definition requires (allowed) facts to be ground. In the following we will always assume a database clause (fact or deductive rule) to be allowed (safe).

Definition 2.7 (Deductive Database) *A deductive database $\mathcal{D}$ is a tuple $\langle \mathcal{F}, \mathcal{R} \rangle$ where $\mathcal{F}$ is a finite set of facts, and $\mathcal{R}$ is a finite set of deductive rules such that $\texttt{pred}(\mathcal{F}) \cap \texttt{pred}(\mathcal{R}) = \emptyset$. Within a deductive database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$, a predicate symbol p is called derived (view predicate), if $p \in \texttt{pred}(\mathcal{R})$. The predicate p is called extensional (or base predicate), if $p \in \texttt{pred}(\mathcal{F})$. Furthermore, $\texttt{pred}(\mathcal{D}) := \texttt{pred}(\mathcal{F}) \uplus \texttt{pred}(\mathcal{R})$ denotes the set of all predicate symbols occurring in $\mathcal{D}$.*

For simplicity of exposition, and without loss of generality, we assume that a predicate is either base or derived, but not both, and that constants do neither occur in rule heads nor in body literals referring to a derived relation. Both conditions can be easily achieved by rewriting a given database.

Example 2.1 *As an example consider the following allowed deductive rules for defining the derived relations* `path` *and* `one_way`*:*

$\texttt{one\_way}(\texttt{X}) \leftarrow \texttt{path}(\texttt{X},\texttt{Y}) \wedge \neg\texttt{path}(\texttt{Y},\texttt{X})$
$\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{edge}(\texttt{X},\texttt{Y})$
$\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{edge}(\texttt{X},\texttt{Z}) \wedge \texttt{path}(\texttt{Z},\texttt{Y})$

Relation `path` *is the transitive closure of* `edge`. *Relation* `one_way` *is the first projection of facts in* `edge` *which do not participate in cycles within the transitive closure* `path`.

Although the main focus of this thesis lies on stratifiable databases, we will also refer to other classes of deductive databases or rule sets, respectively. The classification of databases is purely syntactic and depends on the use of negation within the rules considered. It is defined by means of dependencies between predicates in a given rule set.

Definition 2.8 (Predicate Dependency Graph) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database and $\texttt{pred}(\mathcal{D})$ the set of all predicate symbols in $\mathcal{D}$. The predicate dependency graph of $\mathcal{D}$ is the labelled directed graph $G_\mathcal{D} = \langle V, E \rangle$ where $V = \texttt{pred}(\mathcal{D})$ and E is a set of labelled edges. With $p, q \in \texttt{pred}(\mathcal{D})$, the set of rules containing positive dependencies $R^+_{p,q}$ is defined as*

$$R^+_{p,q} := \{A \leftarrow W \in \mathcal{R} \mid \texttt{pred}(A) = p \text{ and } W \text{ contains a positive literal } L \text{ with } \texttt{pred}(L) = q\}$$

and the set of rules containing negative dependencies $R^-_{p,q}$ is defined as

$$R^-_{p,q} := \{A \leftarrow W \in \mathcal{R} \mid \texttt{pred}(A) = p \text{ and } W \text{ contains a negative literal } L \text{ with } \texttt{pred}(L) = q\}.$$

E contains a negative edge (q, p, neg) with $p, q \in \texttt{pred}(\mathcal{D})$ iff $R^-_{p,q} \neq \emptyset$ and, E contains a positive edge (q, p, pos) iff $R^+_{p,q} \neq \emptyset$ and $(q, p, neg) \notin E$.

Definition 2.9 (Predicate Dependencies) *Let $\mathcal{D}$ be a deductive database and p and q predicate symbols occurring in $\mathcal{D}$, i.e., $p, q \in \texttt{pred}(\mathcal{D})$. We say that*

1. *p depends on q ($p \dashleftarrow q$) iff the predicate dependency graph $G_\mathcal{D}$ of $\mathcal{D}$ contains a path from q to p of length $n \geq 1$,*

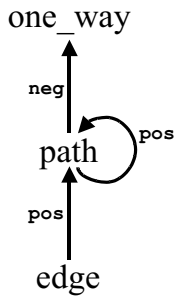
2. *p depends negatively on q ($p \overset{-}{\dashleftarrow} q$) iff $p \dashleftarrow q$ and there is a path from q to p that includes at least one negative edge,*

3. *p depends positively on q ($p \overset{+}{\dashleftarrow} q$) iff $p \dashleftarrow q$ and p depends not negatively on q,*

4. *p and q are mutually dependent on each other ($p \approx q$) iff $p \dashleftarrow q$ and $q \dashleftarrow p$.*

With $p \in \mathtt{pred}(\mathcal{D})$ and $\mathtt{dep}_{\mathcal{R}}(p) = \{q \in \mathtt{pred}(\mathcal{D}) \mid p \leftarrow\!\!- q\}$, we denote the set of rules by which relation p is defined as $\mathtt{def}_{\mathcal{R}}(p) = \{r \in \mathcal{R} \mid \mathtt{pred}(r) \in \mathtt{dep}_{\mathcal{R}}(p) \cup \{p\}\}$.

As an example consider again the deductive rules from Example 2.1. The dependency graph contains no positively labelled edge from `path` to `one_way` because of the negative dependency between these relations. The corresponding dependency graph then is as follows:



In the following we introduce various database classes which are relevant for subsequent discussions. In positive databases, the usage of negation is disallowed, while in semi-positive databases negative references are permitted to extensional relations only. In stratifiable databases, derived relations may be negatively referenced as well, but recursion through negative predicate occurrences is not allowed. A database is hierarchical, if its predicate dependency graph does not contain cycles, i.e., there are no recursive rules in the database.

Definition 2.10 (Database Classes) *Let $\mathcal{R}$ be a deductive rule set and $Rel_{\mathcal{R}}$ the set of all predicate symbols occurring in $\mathcal{R}$. Then $\mathcal{R}$ is called*

1. *positive iff there are no predicate symbols $p, q \in Rel_{\mathcal{R}}$ such that $p \overset{-}{\leftarrow\!\!-} q$.*
2. *semi-positive iff there are no predicate symbols $p, q \in Rel_{\mathcal{R}}$ such that $p \overset{-}{\leftarrow\!\!-} q$ and $q \in \mathtt{pred}(\mathcal{R})$.*
3. *hierarchical iff there is no predicate symbol $p \in Rel_{\mathcal{R}}$ such that $p \approx p$.*
4. *stratifiable, iff there is no predicate symbol $p \in Rel_{\mathcal{R}}$ such that $p \overset{-}{\leftarrow\!\!-} p$.*

A deductive database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ is called positive, semi-positive, hierarchical, or stratifiable if $\mathcal{R}$ is positive, semi-positive, hierarchical, or stratifiable, respectively.

In the literature, the notions stratifiable database and stratified database are often used as synonyms. In this work, however, we will distinguish between these

two notions. In the following a database is called stratifiable if the rule set may be partitioned into rule sets such that none of them contains a negative dependency. In contrast to this, a stratifiable database is denoted stratified only if a particular stratification is given.

The reason for this distinction is that we will extend the concept stratification to general (possibly unstratifiable) databases and use the notion layering instead. In contrast to the concept stratification, a layering allows databases to be partitioned in such a way that individual layers may also contain negative dependencies. These layers are then called unstratified even though they may be stratifiable as well.

Definition 2.11 (Layering) *Let $\mathcal{R}$ be a deductive rule set. A layering λ on $\mathcal{R}$ is a mapping from the set of all predicate symbols $Rel_\mathcal{R}$ occurring in $\mathcal{R}$ to the set of non-negative integers $\mathbb{N}$ such that for all predicate symbols $p, q \in Rel_\mathcal{R}$ the following holds:*

$$\begin{array}{lll} p \in Rel_\mathcal{R} \setminus \mathtt{pred}(\mathcal{R}) & \iff & \lambda(p) = 0 \\ p \in \mathtt{pred}(\mathcal{R}) & \iff & \lambda(p) \geq 1 \\ p \leftarrow\!\!\text{-} \; q & \implies & \lambda(p) \geq \lambda(q). \end{array}$$

In addition, λ defines a partition $\mathcal{R}_1 \uplus \ldots \uplus \mathcal{R}_n$ of $\mathcal{R}$ such that for each predicate symbol $p \in \mathtt{pred}(\mathcal{R})$ all rules r referencing p in their heads, i.e., $\mathtt{pred}(r) = p$, are included in $\mathcal{R}_{\lambda(p)}$.

A stratification is a special layering which induces a partition of a given rule set such that all positive derivations of relations can be determined before a negative literal with respect to one of those relations is evaluated.

Definition 2.12 (Stratification) *Let $\mathcal{R}$ be a deductive rule set. A layering λ on $\mathcal{R}$ is called stratification on $\mathcal{R}$ iff in addition to the layering conditions for all predicates $p, q \in \mathtt{pred}(\mathcal{R})$:*

$$p \overset{-}{\leftarrow\!\!\text{-}} \; q \implies \lambda(p) > \lambda(q).$$

If a is a stratification, $\mathcal{R}$ is called stratified with respect to λ, and each layer is called a stratum.

Obviously, a rule set $\mathcal{R}$ is stratifiable iff a stratification of $\mathcal{R}$ exists. Apart from generally classifying a given deductive rule set $\mathcal{R}$, it is possible to further distinguish subsets of $\mathcal{R}$ according to the different way negation is applied in the rules of $\mathcal{R}$. This classification plays an important role as it allows for further optimizations when the rule set is evaluated.

Definition 2.13 (Deductive Rule Classes) *Let $\mathcal{R}$ be a deductive rule set and λ a layering on $\mathcal{R}$ partitioning the rule set $\mathcal{R}$ into subsets $\mathcal{R}_1, \ldots, \mathcal{R}_m$. Then the deductive rules of each partition $\mathcal{R}_i$ are further divided into the rule classes $\mathcal{R}_i^\circ$, $\mathcal{R}_i^\times$, and $\mathcal{R}_i^*$.*

1. *The class $\mathcal{R}_i^\circ$ comprises all hierarchical rules from $\mathcal{R}_i$, that is, all rules defining relations which may reference relations of lower layers only:*

$$\mathcal{R}_i^\circ := \{\ r \mid\ r \equiv A \leftarrow W \in \mathcal{R}_i \text{ such that for all literals } L \text{ in } W\text{: } \lambda(L) < i\}.$$

2. *The class $\mathcal{R}_i^\times$ comprises all stratified rules from $\mathcal{R}_i$ which positively refer to at least one relation of the same layer but negatively reference relations of lower layers only:*

$$\mathcal{R}_i^\times := \{\ r \mid\ r \equiv A \leftarrow W \in \mathcal{R}_i \text{ such that there exists a positive literal } L \text{ in } W \text{ where } \lambda(L) = i, \text{ and for all negative literals } L \text{ in } W\text{: } \lambda(L) < i\}.$$

3. *The class $\mathcal{R}_i^*$ comprises all unstratifiable rules from $\mathcal{R}_i$ which include at least one negative reference to a relation of the same layer:*

$$\mathcal{R}_i^* := \{\ r \mid\ r \equiv A \leftarrow W \in \mathcal{R}_i \text{ such that there exists a negative literal } L \text{ in } W \text{ where } \lambda(L) = i\}.$$

If $\mathcal{R}$ is a semi-positive rule set and the rule classes are established with respect to the minimal layering on $\mathcal{R}$ resulting in the partition $\mathcal{R} = \mathcal{R}_1$, the rule class $\mathcal{R}_1^\circ$ includes all rules referencing base relations only while $\mathcal{R}_1^\times$ comprises all other rules, and $\mathcal{R}_1^*$ is empty.

2.1.2 Semantics

In this section we present a model-based semantics for deductive databases in a non-constructive way. At the beginning we briefly review the notions of Herbrand base and Herbrand model, as these concepts form a basis of the following expositions. For a more detailed presentation we refer to [Llo87] and [Apt90].

Definition 2.14 (Herbrand Base) *Let $\mathcal{D}$ be a deductive database. The Herbrand base $\mathcal{H}_\mathcal{D}$ of $\mathcal{D}$ is the set of all ground atoms that can be constructed from the predicate symbols and constants occurring in $\mathcal{D}$.*

As we consider finite sets of deductive rules with function-free literals only, the Herbrand bases will always be finite.

Definition 2.15 (Herbrand Interpretation, Model) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database and $\mathcal{H}_\mathcal{D}$ the Herbrand base of $\mathcal{D}$. Any subset $\mathcal{I}$ of $\mathcal{H}_\mathcal{D}$ is a Herbrand interpretation of $\mathcal{D}$. $\mathcal{I}$ is a Herbrand model of $\mathcal{D}$ if $\mathcal{I}$ is a model of $\mathcal{F} \cup \mathcal{R}$, i.e., $\forall f \in \mathcal{F} \cup \mathcal{R} : \mathcal{I} \models f$. A Herbrand model of $\mathcal{D}$ is the least Herbrand model of $\mathcal{D}$ if it is included in every Herbrand Model of $\mathcal{D}$, and it is called minimal if none of its subsets is a Herbrand model of $\mathcal{D}$.*

A deductive database is syntactically given by a set of facts and a set of rules which we call the *explicit state* of the database. In contrast to this we define the *implicit state* of a database to be the set of all positive and negative conclusions that can be derived from the explicit state.

Definition 2.16 (Implicit Database State) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database. The implicit database state $\mathcal{M}_\mathcal{D}$ of $\mathcal{D}$ is defined as the well-founded model [vGRS91] for $\mathcal{F} \cup \mathcal{R}$:*

$$\mathcal{M}_\mathcal{D} = \mathcal{I}^+ \uplus \neg \cdot \mathcal{I}^-$$

where $\mathcal{I}^+, \mathcal{I}^- \subseteq \mathcal{H}_\mathcal{D}$ are sets of ground atoms. The set $\mathcal{I}^+$ represents the true portion of the well-founded model while $\neg \cdot \mathcal{I}^-$ comprises all true negative conclusions, i.e., $\mathcal{I}^-$ includes all false atoms and $\neg \cdot \mathcal{I}^-$ includes all atoms in $\mathcal{I}^-$ in negated form. The set of undefined atoms is implicitly given by $\mathcal{H}_\mathcal{D} \setminus (\mathcal{I}^+ \uplus \mathcal{I}^-)$ comprising all ground atoms of the Herbrand base which are neither true nor false.

According to the definition above, the implicit state of a database partitions the Herbrand base into the set of true conclusions, the set of negative conclusions, and the set of undefined atoms. However, for databases having a total well-founded model, as guaranteed for stratifiable ones, the set of undefined atoms is known to be empty, and the set of false atoms can be derived from the set of true conclusions. In this case, the implicit state can be solely represented by the set of true conclusions (while the Herbrand base is implicitly given), i.e.,

$$\mathcal{M}_\mathcal{D} = \mathcal{I}^+ \uplus \neg \cdot \overline{\mathcal{I}^+}$$

where $\overline{\mathcal{I}^+}$ denotes the complement of $\mathcal{I}^+$ with respect to the Herbrand base, i.e., $\mathcal{H}_\mathcal{D} \setminus \mathcal{I}^+$. Based on these considerations the implicit state of a stratifiable database may also be represented by the set of true atoms only, i.e., $\mathcal{M}_\mathcal{D} = \mathcal{I}^+$. With respect to an arbitrary well-founded model $\mathcal{M}_\mathcal{D}$, we will use $\mathcal{M}_\mathcal{D}^+$ for referring to the true portion of it, i.e., $\mathcal{M}_\mathcal{D}^+ = \mathcal{I}^+$ with $\mathcal{I}^+ \uplus \neg \cdot \mathcal{I}^- = \mathcal{M}_\mathcal{D}$. In addition, we will employ the following equivalences.

Lemma 2.1

1. *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable database. Then the well-founded model $\mathcal{M}_\mathcal{D}$ of $\mathcal{D}$ coincides with the perfect model [1] of $\mathcal{D}$.*
2. *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a positive or semi-positive database. Then the well-founded model $\mathcal{M}_\mathcal{D}$ of $\mathcal{D}$ coincides with the least Herbrand model of $\mathcal{D}$.*

[1] For a definition of perfect models we refer to Section 3.2 where we recall a constructive method for determining the perfect model of a stratifiable database by means of iterated fixpoint computation [ABW88]. Note that perfect models are defined in [Prz88] in a more general form.

Proof:

1. cf. [vGRS91].
2. For a semi-positive database there exists only one minimal Herbrand model which is identical with its least Herbrand model. As each perfect model is a minimal Herbrand model [ABW88] the proposition follows.

□

This model-based semantics is not well-suited for computing the implicit state of a given database as it defines the semantics in a non-constructive way. Therefore, in Chapter 3 we will present the fixpoint semantics for the different classes of deductive databases in order to provide constructive methods for computing the corresponding well-founded models. Nevertheless, the introduced model-based semantics represent the theoretical basis of propositions and proofs in subsequent sections.

2.2 Queries

A database language can usually be divided into data definition language (DDL) and data manipulation language (DML), the latter one including the data query language (DQL). A query is to be formulated by means of the data query language specifying a new temporary relation which does not belong to the actual database. In Datalog, a query is given by an atom referencing an existing base or derived relation. More complex queries may require additional rules to be added to the database schema. In this way, Datalog's DQL represents a query language which is relationally complete when built-in predicates are included as well.

2.2.1 Syntax

Each query represents a subset of an existing extensional or intensional relation of a given database.

Definition 2.17 (Database Query) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database. A database query with respect to $\mathcal{D}$ is an expression of the form*

$$? - A$$

where A is an atom referencing a relation in $\mathcal{D}$, i.e., $\texttt{pred}(A) \in \texttt{pred}(\mathcal{D})$.

As rules for defining an atomic query are part of the database schema, they are assumed to be safe. Therefore, it is not necessary to additionally define safe queries as this property is already provided. Note that this restricted view of queries is assumed only for the sake of simplicity of exposition and does not restrict the expressiveness of the query language considered. In the following section we will introduce the semantics of queries using the former introduced semantics of deductive databases.

2.2.2 Semantics

The semantics of a query is defined by means of its answer set which comprises all true conclusions matching the query.

Definition 2.18 (Answer Set) *Let $\mathcal{D}$ be a deductive database, Q a query with respect to $\mathcal{D}$ and $\mathcal{M}_\mathcal{D}$ the well-founded model of $\mathcal{D}$. The answer set of Q, denoted by* $\texttt{ans}(Q, \mathcal{D})$*, is defined as*

$$\texttt{ans}(Q, \mathcal{D}) := \{L \mid L \equiv Q\sigma,\ \sigma \text{ is a ground substitution for all variables in } Q \text{ and } L \in \mathcal{M}_\mathcal{D}\}.$$

Note that a boolean query is represented by a ground atom and is evaluated to true if the corresponding answer set contains this ground atom. Otherwise, the answer set is empty. Boolean queries form the basis for integrity constraints which will be considered in the following section.

2.3 Integrity Constraints

In a database context, static and dynamic constraints are distinguished. A static integrity constraint induces a boolean condition which has to be satisfied in every consistent database state whereas dynamic constraints induce restrictions on database state transitions. In the following we will solely consider static constraints which may reference base as well as derived relations.

2.3.1 Syntax

Integrity constraints are represented by means of ground atoms which have to be derivable in every state of a database. Similar to queries, we assume more complex integrity constraints to be formulated by means of ground atoms which reference derived relations whose corresponding defining rules are added to the database schema. However, these rules are not part of a constraint definition but are considered as 'regular' deductive rules.

Definition 2.19 (Integrity Constraint) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database. An integrity constraint c with respect to $\mathcal{D}$ is a ground atom such that* $\texttt{pred}(c) \in \texttt{pred}(\mathcal{D})$.

Given a nonempty set of integrity constraints $\mathcal{C}$ with respect to a deductive database $\mathcal{D}$, we will use the triple $\langle \mathcal{F}, \mathcal{R}, \mathcal{C} \rangle$ to specify $\mathcal{D}$ in the following. In the next section the semantics of integrity constraints is introduced by means of consistent database states.

2.3.2 Semantics

In a consistent database state every static integrity constraint must be satisfied; that is, every ground atom specified as an integrity constraint must be derivable.

Definition 2.20 (Consistent Database State) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}, \mathcal{C} \rangle$ be a deductive database with constraints and $\mathcal{M}_\mathcal{D}$ the well-founded model of $\langle \mathcal{F}, \mathcal{R} \rangle$. $\mathcal{D}$ is called consistent iff $\mathcal{C} \subseteq \mathcal{M}_\mathcal{D}$. Otherwise $\mathcal{D}$ is called inconsistent.*

In the context of integrity constraints, the semantics of a deductive database is defined if and only if all constraints are satisfied.

Definition 2.21 (Semantics of Deductive Databases with Constraints) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}, \mathcal{C} \rangle$ be a stratifiable deductive database with constraints and $\mathcal{M}_\mathcal{D}$ the well-founded model of $\langle \mathcal{F}, \mathcal{R} \rangle$. If $\mathcal{D}$ is consistent, then $\mathcal{M}_\mathcal{D}$ is the semantics of $\mathcal{D}$. Otherwise the semantics of $\mathcal{D}$ is undefined.*

Example 2.2 *The following stratifiable rules, constraints and facts represent a consistent deductive database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}, \mathcal{C} \rangle$:*

$\mathcal{R}$:

$\mathtt{ic}_1 \leftarrow \mathtt{path}(\mathtt{X}, \mathtt{Y})$
$\mathtt{ic}_2 \leftarrow \neg\mathtt{aux}$
$\mathtt{aux} \leftarrow \mathtt{edge}(\mathtt{X}, \mathtt{X})$
$\mathtt{one\_way}(\mathtt{X}) \leftarrow \mathtt{path}(\mathtt{X}, \mathtt{Y}) \wedge \neg\mathtt{path}(\mathtt{Y}, \mathtt{X})$
$\mathtt{path}(\mathtt{X}, \mathtt{Y}) \leftarrow \mathtt{edge}(\mathtt{X}, \mathtt{Y})$
$\mathtt{path}(\mathtt{X}, \mathtt{Y}) \leftarrow \mathtt{edge}(\mathtt{X}, \mathtt{Z}) \wedge \mathtt{path}(\mathtt{Z}, \mathtt{Y})$

$\mathcal{C}$:

$\mathtt{ic}_1$, $\mathtt{ic}_2$

$\mathcal{F}$:

```
edge(1,2), edge(1,4), edge(2,3)
```

Apart from the rules of the previous Example 2.1 for defining the relations `path` *and* `one_way`*, this example contains further rules for defining the derived relations ic_1 and ic_2 which are used for specifying corresponding integrity constraints. Constraint ic_1 requires that in every consistent database state at least one path-tuple exists. Constraint ic_2 is used to prevent cycles in edge. The semantics of $\mathcal{D}$ is given by its total well-founded model $\mathcal{M}_\mathcal{D} = \mathcal{F} \cup \{path(1,2), path(1,4), path(2,3), path(1,3)\} \cup \{one\_way(1), one\_way(2)\} \cup \{ic_1, ic_2\}$.*

Integrity constraints are invariant against state modifications caused by update operations. The following section defines an update language well going with deductive databases as defined above.

2.4 Updates

In this section we introduce the syntax and semantics of modifications on extensional as well as intensional relations of a given deductive database. We refrain from presenting a concrete update language but rather concentrate on the resulting sets of update primitives specifying insertions and deletions of individual facts. In principle, every set-oriented update language can be used that allows for specification of modifications of this kind. After introducing the syntax of our update primitives, we define the semantics of an update in a set-oriented way which fits with the semantics of deductive databases introduced above. We will use the notion *Update* to denote the 'true' changes caused by a transaction only; that is, we restrict the set of facts to be updated to the minimal set of updates where compensation effects (given by an insertion and deletion of the same fact or the insertion of facts which already exist in the database) are already considered. Therefore, updates may be seen as the effect of an applied transaction.

Definition 2.22 (Update) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable deductive database. An update u_D is a pair $\langle u_D^+, u_D^- \rangle$ where u_D^+ and u_D^- are sets of base facts with $\texttt{pred}(u_D^+ \cup u_D^-) \subseteq \texttt{pred}(\mathcal{F})$, $u_D^+ \cap u_D^- = \emptyset$, $u_D^+ \cap \mathcal{F} = \emptyset$ and $u_D^- \subseteq \mathcal{F}$. The atoms u_D^+ represent facts to be inserted into $\mathcal{D}$, whereas u_D^- contains the facts to be deleted from $\mathcal{D}$.*

We will use the notion *induced update* to refer to the entire set of facts in which the new state of the database differs from the old state after an update of base tables has been applied.

Definition 2.23 (Induced Update) *Let $\mathcal{D}$ be a stratifiable database, $\mathcal{M}_\mathcal{D}$ the semantics of $\mathcal{D}$ and $u_\mathcal{D}$ an update. Then $u_\mathcal{D}$ leads to an induced update $u_{D \to D'}$ from D to D' which is a pair $\langle u_{D \to D'}^+, u_{D \to D'}^- \rangle$ of sets of ground atoms such that $u_{D \to D'}^+ = \mathcal{M}_{D'}^+ \backslash \mathcal{M}_D^+$ and $u_{D \to D'}^- = \mathcal{M}_D^+ \backslash \mathcal{M}_{D'}^+$. The atoms $u_{D \to D'}^+$ represent the induced insertions, whereas $u_{D \to D'}^-$ consists of the induced deletions.*

The computation of the induced updates of derived relations resulting from an explicitly performed update of the extensional fact base is called update propagation and will be considered in more detail in Section 5. As each induced update $u_{D \to D'}$ contains the 'net' difference between old and new database state, it is possible to compute the old state from the new one, and vice versa. However, for computing the other state of a database efficiently, it is necessary to refer to the specific changes of relations occurring in $\mathcal{D}$. We will use the notion *delta relation* to access induced insertions or deletions explicitly.

Definition 2.24 (Delta Relation) *Let $\mathcal{D}$ be a stratifiable database and $u_\mathcal{D}$ an update. For each predicate symbol $p \in \texttt{pred}(\mathcal{D})$, a pair of delta relations $\langle \Delta^+\texttt{p}, \Delta^-\texttt{p} \rangle$*

is defined for representing the insertions and deletions induced on p by the update $u_{\mathcal{D}}$. The delta relations defined for a predicate p have the same arity and type as p, i.e., if p is extensional respectively derived, then $\Delta^{+}\mathtt{p}$ and $\Delta^{-}\mathtt{p}$ are extensional respectively derived as well.

In general, update propagation methods analyze the deductive rules of a given database in order to systematically determine such delta relations which provide a focus on the specific changes of relations after an update has been applied.

Chapter 3

Model Computation

This chapter deals with computing the well-founded model, i.e., the implicit state, of a deductive database by means of fixpoint computations. We will use the fixpoint semantics of deductive databases which provide constructive methods for determining the semantics of a deductive rule set with respect to a given database. In this denotational semantics approach, a deductive rule denotes a fact-generating function instead of a logical formula. In order to formalize the derivation of facts in this context we define different consequence operators based on the immediate consequence operator introduced by van Emden and Kowalski [vEK76]. Among them, the *soft consequence* operator [Beh03] from Section 3.2.2 represents the most important one in this thesis because it serves as the basic evaluation mechanism for softly stratifiable rules and related transformation-based techniques proposed in subsequent chapters.

All consequence operators presented in the following were originally introduced for providing a fixpoint-based characterization of the semantics of different deductive database classes, namely semi-positive, stratifiable, and general databases. Section 3.1.1 is concerned with determining the implicit state of semi-positive databases using a transformation-based approach to the well-known *differential fixpoint computation*. Section 3.2.1 then deals with stratifiable databases and shows how the soft consequence operator can be used for implementing the *iterated fixpoint computation*. Finally, Section 3.3.2 investigates arbitrary, i.e., possibly unstratifiable, databases. It presents the approach proposed by Kemp, Srivasta, and Stuckey in [KSS91, KSS95] for implementing the *alternating fixpoint computation* introduced by Van Gelder in [vG89]. In Section 6.2 we will further enhance their method by avoiding its drawback of repeated computations. In our approach, such recomputation is prevented by incorporating update propagation and soft stratification leading to an incremental algorithm which extends the differential evaluation techniques for stratifiable databases.

The individual fixpoint computations are not defined in isolation, but are based upon each other. In the following, they will serve as basic evaluation methods for computing the semantics of the three different database classes. However, they

do not represent realistic database engines as many classical techniques for query optimization in relational systems, e.g. algebraic manipulation, have not been considered. In addition, further methods for optimizing recursive rule evaluation, e.g. [BR86, Han88, RBK88, NRSU89, KRS90, RSS94, LTD95, NRSU95, VM96, SMK97, Cha98], orthogonal to the overall fixpoint computation processes have been omitted as well. Therefore, it is important for the quality of the proposed fixpoint computation methods to pose as few restrictions as possible to the application of rule sets in each iteration round. The original definition of stratifiable rules [ABW88, vG88, Naq86] is an example of a too restrictive way of handling rules with negation as dependencies between predicates and not between rules are considered. It is, however, often sufficient to delay the application of rules which actually have negated derived body literals only and not the entire set of rules that define a relation (cf. also [RSS94, IN88]). Fixpoint computations defined in a most general manner then can be used for extending existing relational systems allowing the correct evaluation of recursive views as proposed in the new SQL:1999 standard and still being most flexible in the underlying relational optimization phase.

3.1 Differential Fixpoint Computation

The implicit state of a deductive database is generally defined as its well-founded model, which in case of semi-positive databases coincides with the least Herbrand model. Differential fixpoint computation or rather semi-naive materialization is an approach to efficiently computing the least Herbrand model of semi-positive databases which has been commonly accepted as 'the method of choice in the deductive database literature' [NR91]. This is in particular caused by the fact that this approach forms an essential component of iterated (cf. Section 3.2) as well as alternating fixpoint computation (cf. Section 3.3).

In Section 3.1.1 we present the theoretical foundations of the fixpoint semantics by means of consequence operators. Section 3.1.2 then gives an example showing the general course of differential fixpoint computation. Afterwards we discuss various approaches to implementing differential fixpoint computations.

3.1.1 Computing the Least Herbrand Model

In order to formalize the derivation of facts in this denotational semantics approach we recall the immediate consequence operator adopting the presentation in [Man03]. Note that this consequence operator is based on the derivation operator introduced by van Emden and Kowalski [vEK76].

Definition 3.1 (Immediate Consequence Operator) *Let $\mathcal{R}$ be a set of deductive rules and f an arbitrary set of facts.*

1. *Given a rule $r \equiv H \leftarrow A_1 \wedge \ldots \wedge A_n \wedge \neg B_1 \wedge \ldots \wedge \neg B_m \in \mathcal{R}$ with $n > 0$ and $m \geq 0$, the consequence operator T defines the set of all facts which can be derived by a single application of r with respect to f:*

$$T[r](f) := \{ H\sigma \mid \sigma \text{ is a ground substitution, } \forall\, 1 \leq i \leq n\text{: } A_i\sigma \in f \text{ and } \forall\, 1 \leq j \leq m\text{: } B_i\sigma \notin f \}.$$

2. *The consequence operator $T_{\mathcal{R}}$ defines the set of all facts derivable by the simultaneous application of all rules contained in $\mathcal{R}$:*

$$T_{\mathcal{R}}(f) := \bigcup_{r \in \mathcal{R}} T[r](f).$$

3. *The immediate consequence operator $T^{\star}_{\mathcal{R}}$ accumulates the set of input facts and the set of derivable facts:*

$$T^{\star}_{\mathcal{R}}(f) := T_{\mathcal{R}}(f) \cup f.$$

The consequence operator T defines the set of all facts which can be derived by a single application of a deductive rule. Negative literals are evaluated according to the *negation as failure*-principle [Cla78] which itself is based on the *closed world assumption* [Rei78]. The simultaneous application of a set of deductive rules is defined by the operator $T_{\mathcal{R}}$. During the evaluation, the newly derived facts of any $T[r_i]$-application are not visible to other $T[r_j]$-computations. Therefore any rules depending on derived relations may necessitate further $T_{\mathcal{R}}$-applications. As the intermediate results of these applications must be kept in the course of the overall derivation process, the operator $T^{\star}_{\mathcal{R}}$ accumulates its input facts and the set of derivable facts.

As the immediate consequence operator $T^{\star}_{\mathcal{R}}$ is monotonic for semi-positive databases, its least fixpoint $\mathtt{lfp}(T^{\star}_{\mathcal{R}}, \mathcal{F})$ exists [vEK76], where $\mathtt{lfp}(T^{\star}_{\mathcal{R}}, \mathcal{F})$ denotes the least fixpoint of operator $T^{\star}_{\mathcal{R}}$ containing the set of facts $\mathcal{F}$.

Lemma 3.1 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a positive or semi-positive database. The positive portion of the total well-founded model $\mathcal{M}_{\mathcal{D}}$ of $\langle \mathcal{F}, \mathcal{R} \rangle$ coincides with the least fixpoint of $T^{\star}_{\mathcal{R}}$, i.e.,*

$$\mathcal{M}_{\mathcal{D}} = \mathtt{lfp}(T^{\star}_{\mathcal{R}}, \mathcal{F}) \uplus \neg \cdot \overline{\mathtt{lfp}(T^{\star}_{\mathcal{R}}, \mathcal{F})}.$$

Proof: eg. [Llo87, p. 37-38]. □

Lemma 3.1 points out the way towards an iterative set-oriented bottom-up implementation for materializing the well-founded model of semi-positive databases.

In such a realization the immediate consequence operator is iteratively applied to the database facts until all positive conclusions have been inferred. The disadvantage of this naive procedure, however, is that in each iteration round all positive conclusions of preceding iteration rounds are repeatedly derived. This well-known drawback is avoided in the semi-naive materialization strategy which will be discussed in the following section.

3.1.2 Semi-naive Materialization

The semi-naive materialization approach is based on the idea that new facts can only be derived, if at least one of the literals in a rule's body refers to a fact which has been newly inferred in the preceding iteration round. We will only present an informal description of the global course of differential fixpoint computation. For a more technical presentation we refer to publications like [Ban86, Ull89, CGT90, CGH94].

The global course of differential fixpoint computation can usually be divided into two phases. In the first one, this approach computes all facts which can be directly derived from base facts; that is, all hierarchical rules solely referring to extensional relations are applied once. Afterwards, in an iterative phase all further facts are incrementally computed starting from the initially obtained derivations. In each iteration round, the evaluation of at least one derived body literal is restricted to the set of new facts which have been obtained in the preceding iteration round. This guarantees that each derivation relies on at least one new fact, and thus has not been computed before.

For storing newly derived facts of an iteration round we use a delta relation Δp for every derived predicate p[1]. Delta relations contain all facts which have been newly derived for a corresponding derived relation in the preceding iteration round. In a transformation-based approach, these relations may be defined by means of delta rules which can be derived from the original rule set.

Definition 3.2 (Delta Rules) *Let $\mathcal{R}$ be a semi-positive rule set. The differential fixpoint transformation maps $\mathcal{R}$ to a set of delta rules $\mathcal{R}_\Delta$ which are defined as follows:*

1. *For each deductive rule $r \equiv p(\vec{x}) \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}$ which contains no derived body literal a derived delta rule of the form*

   $$\Delta p(\vec{x}) \leftarrow L_1 \wedge \ldots \wedge L_n \wedge \neg p(\vec{x})$$

   is in $\mathcal{R}^d_\Delta$, where Δp represents the delta relation of $\texttt{pred}(p(\vec{x}))$.

[1]Note that the notion delta relation will also be used in the context of update propagation in Chapter 5 but with a quite different meaning. However, it will be always clear from the context which kind of delta relation is meant.

Algorithm 1: Differential fixpoint computation

Initialization Phase:

$\Delta f := T_{\mathcal{R}^d_\Delta}(\mathcal{F})$;
$f \quad := T_{\mathcal{R}^b_\Delta}(\Delta f) \cup \mathcal{F}$;

Iteration Phase:

repeat
  $\Delta f := T_{\mathcal{R}^d_\Delta}(f \cup \Delta f)$;
  $f \quad := T_{\mathcal{R}^b_\Delta}(\Delta f) \cup f$;
until $\Delta f = \emptyset$;

2. *For each deductive rule* $r \equiv p(\vec{x}) \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}$ *and for each derived body literal* $L_i \equiv q(\vec{y})$ $(1 \leq i \leq n)$ *a derived delta rule of the form*

$$\Delta p(\vec{x}) \leftarrow L_1 \wedge \ldots \wedge L_{i-1} \wedge \Delta q(\vec{y}) \wedge L_{i+1} \wedge \ldots \wedge L_n \wedge \neg p(\vec{x})$$

is in $\mathcal{R}^d_\Delta$*, where* Δp *and* Δq *represent the delta relation of* $\texttt{pred}(p(\vec{x}))$ *and* $\texttt{pred}(L_i)$*, respectively.*

3. *For each n-ary derived relation* p *with* $\texttt{pred}(p) \in \texttt{pred}(\mathcal{R})$ *a basic delta rule of the form*

$$p(x_1, \ldots, x_n) \leftarrow \Delta p(x_1, \ldots, x_n)$$

is in $\mathcal{R}^b_\Delta$*, where* Δp *represents the delta relation of* p *and* $\{x_1, \ldots, x_n\}$ *are distinct variables.*

4. *No other rules are in* $\mathcal{R}_\Delta := \mathcal{R}^d_\Delta \uplus \mathcal{R}^b_\Delta$.

The application of the delta rules $\mathcal{R}_\Delta$ according to the Algorithm 1 presented above corresponds to a semi-naive materialization of the original rule set $\mathcal{R}$. In this scheme, f denotes the intermediate state of the database comprising all facts which have been computed in previous iteration rounds. In contrast to this, the set Δf comprises the extensions of all delta relations consisting of all facts which have been newly inferred in the preceding iteration round.

In the initialization phase, all hierarchical rules in $\mathcal{R}^d_\Delta$ which solely refer to base relations are applied leading to the first delta facts of derived relations. Afterwards, these derived relations are initialized with these delta facts as well, using the basic delta rules $\mathcal{R}^b_\Delta$. Their application leads to a first 'inflation' of the fact base $\mathcal{F}$ which is stored in the set f. During the iteration phase, all rules in $\mathcal{R}^d_\Delta$ are applied which refer to a non-empty delta relation in their rule body. The added negated head literal in the bodies of these derived delta rules

ensures that only new facts are stored in the corresponding delta relations. Note that the consistent application of the non-cumulative $T_{\mathcal{R}}(f)$-operator leads to an overriding of previously obtained delta facts. Therefore, it is necessary to retain previously computed facts in f during the next step when delta facts are 'copied' to derived relations by applying the basic delta rules $\mathcal{R}^b_\Delta$. These steps are iterated until no more facts can be inferred, i.e., until all delta relations are empty. Afterwards, the database will be entirely materialized.

As an example consider the following (slightly modified) deductive database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ which defines a derived relation `path` as the transitive closure of the extensional relation `edge`:

$\underline{\mathcal{R}}$:

$$\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{edge}(\texttt{X},\texttt{Y})$$
$$\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{path}(\texttt{X},\texttt{Z}) \wedge \texttt{path}(\texttt{Z},\texttt{Y})$$

$\underline{\mathcal{F}}$:

```
edge(1,2), edge(2,3), edge(3,4)
```

The differential fixpoint transformation of $\mathcal{R}$ would yield the following delta rules $\mathcal{R}_\Delta = \mathcal{R}^b_\Delta \uplus \mathcal{R}^d_\Delta$:

$\underline{\mathcal{R}^b_\Delta}$:

$$\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \Delta\texttt{path}(\texttt{X},\texttt{Y})$$

$\underline{\mathcal{R}^d_\Delta}$:

$$\Delta\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{edge}(\texttt{X},\texttt{Y}) \wedge \neg\texttt{path}(\texttt{X},\texttt{Y})$$
$$\Delta\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \Delta\texttt{path}(\texttt{X},\texttt{Z}) \wedge \texttt{path}(\texttt{Z},\texttt{Y}) \wedge \neg\texttt{path}(\texttt{X},\texttt{Y})$$
$$\Delta\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{path}(\texttt{X},\texttt{Z}) \wedge \Delta\texttt{path}(\texttt{Z},\texttt{Y}) \wedge \neg\texttt{path}(\texttt{X},\texttt{Y})$$

The application of these rules using the scheme in Algorithm 1 induces the following sequence of sets:

$$\left.\begin{array}{rl} \Delta f &:= \{\Delta\texttt{path}(1,2), \Delta\texttt{path}(2,3), \Delta\texttt{path}(3,4)\} \\ f &:= \{\texttt{path}(1,2), \texttt{path}(2,3), \texttt{path}(3,4)\} \cup \mathcal{F} \end{array}\right\} \texttt{initialization phase}$$

$$\left.\begin{array}{rl} \Delta f &:= \{\Delta\texttt{path}(1,3), \Delta\texttt{path}(2,4)\} \\ f &:= \{\texttt{path}(1,3), \texttt{path}(2,4)\} \cup f \\ \Delta f &:= \{\Delta\texttt{path}(1,4)\} \\ f &:= \{\texttt{path}(1,4)\} \cup f \\ \Delta f &:= \emptyset \\ f &:= \emptyset \cup f \end{array}\right\} \texttt{iteration phase}$$

The result in f coincides with the true portion $\mathcal{M}^+_\mathcal{D}$ of the total well-founded model of $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$, i.e., $f = \texttt{lfp}(T^\star_\mathcal{R}, \mathcal{F})$. During the materialization process

the recomputation of `path`-facts obtained in previous iteration rounds is avoided. On the other hand, during the iteration phase all delta facts are computed twice which is basically caused by the redundant storage of newly derived facts in delta relations as well as in derived relations. This effect can be avoided by storing newly derived facts of the i-th iteration round in delta relations only while the corresponding derived relations still contain the facts of the previous (i-1) iteration rounds. In this case, however, the derived delta rules must be inferred from the original rule set by substituting not only a single derived body literal but any subset of derived body literals by a corresponding delta relation. This would lead to an exponential number of derived delta rules. In addition, this approach still leaves room for redundancy but can be further refined such that no derivations are considered twice during the course of the iteration process [BR87, RSS94] realizing the so-called non-repetition property.

The transformation-based approach to differential fixpoint computation of semi-positive databases forms a basis for further transformation-based approaches capable of handling more general rule classes. In the following section stratifiable rule sets are considered which are in particular interesting for systems allowing the definition of recursive views according to SQL:1999.

3.2 Iterated Fixpoint Computation

Differential fixpoint computation as considered in the previous section correctly determines the least Herbrand model of a semi-positive database. If in addition we allow stratified negation with respect to derived relations, the approach has to be extended such that negative literals are correctly handled according to the negation as failure principle. This leads to a fixpoint computation process which is iteratively applied to each stratum of the given rule set.

In Section 3.2.1 we recall the theoretical foundations of iterated fixpoint computation based on the immediate consequence operator introduced above. In Section 3.2.2 an alternative approach is presented based on the soft consequence operator which itself represents a variant of Kerisit's weak consequence operator [KP88]. This approach covers even more general rule classes, i.e., weakly and softly stratifiable rules.

3.2.1 Computing the Perfect Model

The well-founded model of a stratifiable database coincides with its perfect model, for which a constructive definition has been given, e.g., in [Apt90]. The perfect model of a stratifiable database in turn can be determined by means of iterated fixpoint computation [ABW88]. The basic idea of this computation method is to postpone the evaluation of negative literals until all possible positive conclusions

have been made. Afterwards, negative literals can be evaluated according to the negation as failure principle.

Definition 3.3 (Iterated Fixpoint) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable deductive database and λ a stratification on $\mathcal{D}$. The partition $\mathcal{R}_1 \cup \ldots \cup \mathcal{R}_n$ of $\mathcal{R}$ defined by λ induces a sequence of least Herbrand models $\mathcal{F}_0, \ldots, \mathcal{F}_n$ as follows:*

$\mathcal{F}_0 := \mathcal{F}$
$\mathcal{F}_i := \mathtt{lfp}(T^{\star}_{\mathcal{R}_i}, \mathcal{F}_{i-1})$ *with* $1 \leq i \leq n$.

The iterated fixpoint $\mathcal{IF}_{\mathcal{D}}$ of $\mathcal{D}$ is defined as $\mathcal{IF}_{\mathcal{D}} := \mathcal{F}_n$.

Lemma 3.2 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable deductive database, λ a stratification on $\mathcal{D}$, $\mathcal{R}_1 \cup \ldots \cup \mathcal{R}_n$ the partition of $\mathcal{R}$ induced by λ and $\mathcal{IF}_{\mathcal{D}}$ the iterated fixpoint of $\mathcal{D}$. Then the positive portion of the total well-founded model $\mathcal{M}_D$ of $\mathcal{D}$ coincides with the iterated fixpoint, i.e.,*

$\mathcal{M}_{\mathcal{D}} = \mathcal{IF}_{\mathcal{D}} \cup \neg \cdot \overline{\mathcal{IF}_{\mathcal{D}}}$.

Proof: This proposition follows from the fact that the well-founded model of a stratifiable database $\mathcal{D}$ is identical with the perfect model of $\mathcal{D}$ (cf. [vGRS91]) whose positive portion coincides with the iterated fixpoint of $\mathcal{D}$ (cf. [Prz88]).
□

The iterated fixpoint is obtained by subsequently 'materializing' the strata of a given stratification from bottom to top. As each stratum negatively refers to already materialized strata only, each pair $\langle \mathcal{F}_{i-1}, \mathcal{R}_i \rangle$ with $1 \leq i \leq n$ corresponds to a semi-positive database. Thus, differential fixpoint computation as considered in Section 3.1.1 is applicable to each individual stratum for computing the corresponding sequence of intermediate least fixpoints.

3.2.2 The Soft Consequence Operator

In this section, an alternative approach to computing the perfect model is presented which is based on the soft consequence operator [Beh03], a variant of the weak consequence operator introduced by Kerisit and Pugin in [KP88]. The weak consequence operator and the concept of weak stratification are part of the Alexander method for query evaluation in stratifiable databases [RLK86] and form a basis for our soft stratification approach for transformation-based methods in subsequent sections (cf. Chapter 4). In this section, however, we concentrate on the application and suitability of the soft consequence operator for materializing stratifiable databases only.

The soft consequence operator is a modified version of the immediate consequence operator proposed for determining the semantics of *softly stratified rule sets* which will be introduced in Section 4.2.2. The basic idea is to integrate the iteration over the given 'strata' into the consequence operator itself.

Definition 3.4 (Soft Consequence Operator) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database and $\mathcal{P} = P_1 \cup \ldots \cup P_n$ an arbitrary partition of $\mathcal{R}$. The soft consequence operator $T^s_{\mathcal{P}}$ is a mapping on sets of ground atoms and is defined for $\mathcal{I} \subseteq \mathcal{H}_{\mathcal{D}}$ as follows:*

$$T^s_{\mathcal{P}}(\mathcal{I}) := \begin{cases} \mathcal{I} & \text{if there is no } j \in \{1, \ldots, n\} \text{ such that } T^\star_{P_j}(\mathcal{I}) \supsetneq \mathcal{I} \\ \\ T^\star_{P_i}(\mathcal{I}) & \text{with } i := \mathtt{min}\{j \mid T^\star_{P_j}(\mathcal{I}) \supsetneq \mathcal{I}\}, \text{ otherwise.} \end{cases}$$

As the soft consequence operator is monotonic, its least fixpoint exists and is given by lfp $(T^s_{\mathcal{P}}, \mathcal{F})$. Although the operator $T^s_{\mathcal{P}}$ is intended for handling softly stratified rules, it is defined for an arbitrary partition of the input rule set making it applicable to various kinds of stratification concepts. The following lemma shows that $T^s_{\mathcal{P}}$ can be already used for determining the semantics of stratified deductive databases.

Lemma 3.3 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable deductive database and λ a stratification of $\mathcal{R}$ inducing the partition $\mathcal{P}$ of $\mathcal{R}$. The positive portion of the total well-founded model $\mathcal{M}_{\mathcal{D}}$ of $\langle \mathcal{F}, \mathcal{R} \rangle$ is identical with the least fixpoint of $T^s_{\mathcal{P}}$, i.e.,*

$$\mathcal{M}_{\mathcal{D}} = \mathtt{lfp}(T^s_{\mathcal{P}}, \mathcal{F}) \cup \neg \cdot \overline{\mathtt{lfp}(T^s_{\mathcal{P}}, \mathcal{F})}.$$

Proof: The proposition of the lemma is shown by induction on the number of components in the partition $\mathcal{P}$ induced by the stratification λ on $\mathcal{R}$. Let $\mathcal{M}^+_D$ denote the true portion of the total well-founded model $\mathcal{M}_{\mathcal{D}}$ of $\mathcal{D}$ in the following.

Suppose that $l = 1$: All negative literals in P_1 solely refer to base relations. Hence, the true portion of the well-founded model of the semi-positive rule set P_1 for an arbitrary fact base X is given by

$$\begin{aligned} \mathcal{M}^+_{\langle P_1, X \rangle} &= \text{lfp } (T^\star_{P_1}, X) \\ &=_{def} T^\star_{P_1}(T^\star_{P_1}(\ldots T^\star_{P_1}(X) \ldots) \\ &=_{def} \text{lfp } (T^s_{P_1}, X) \end{aligned}$$

This holds in particular for the fact base $X = \mathcal{F}$.

Suppose that $l > 1$: Assuming

$$\mathtt{lfp}\ (T^s_{P_1 \cup \ldots \cup P_{l-1}}, X) = \mathcal{M}^+_{\langle P_1 \cup \ldots \cup P_{l-1}, X \rangle}$$

holds for any fact base X, we have to show that

$$\mathtt{lfp}\ (T^s_{P_1 \cup \ldots \cup P_l}, X) = \mathcal{M}^+_{\langle P_1 \cup \ldots \cup P_l, X \rangle}.$$

As the partition $\mathcal{P} = P_1 \cup \ldots \cup P_l$ is induced by the stratification λ of $\mathcal{R}$ the following condition must hold:

$\texttt{pred}(P_l) \cap \texttt{pred}(P_1 \cup \ldots \cup P_{l-1}) = \emptyset$.

According to definition 3.4 the soft consequence operator $T^s_{\mathcal{P}}$ always selects the first component of $\mathcal{P}$ for evaluation which leads to a derivation of new facts. Assuming a correct evaluation of components $P_1, \ldots, P_{l-1}$, only the stratum P_l may lead to new derivations. Because of the condition above these newly computed facts cannot lead to new derivations in a subsequent iteration round if applied to $T^\star_{P_i}$ with $1 \leq i \leq l-1$. Since all negative literals in P_l reference relations in lower strata only, we have

$$\begin{aligned}\texttt{lfp}\ (T^s_{P_1 \cup \ldots \cup P_l}, X) &= T^\star_{P_l}(T^\star_{P_l}(\ldots T^\star_{P_l}(\mathcal{M}^+_{\langle P_1 \cup \ldots \cup P_{l-1}, X\rangle})\ldots) \\ &= \texttt{lfp}\ (T^\star_{P_l}, \mathcal{M}^+_{\langle P_1 \cup \ldots \cup P_{l-1}, X\rangle}) \\ &= \mathcal{M}^+_{\langle P_1 \cup \ldots \cup P_l, X\rangle}\end{aligned}$$

which holds in particular for the fact base $X = \mathcal{F}$. □

The proof of Lemma 3.3 shows that in case of stratified rules $\mathcal{R}$ the least fixpoint computation of $T^s_{\mathcal{R}}$ coincides with the iterated fixpoint computation of $\mathcal{R}$. Thus, differential fixpoint computation as considered in Section 3.1.1 is applicable. However, the soft consequence operator is more general, thus allowing the application of partitioned rules $P_1 \cup \ldots \cup P_n$ for which the condition

$\texttt{pred}(P_i) \cap \texttt{pred}(P_j) = \emptyset$

with $i \neq j$ does not necessarily hold. In this case newly derived facts may allow further derivations in lower partition sets. Therefore, for computing the least fixpoint of $T^s_{\mathcal{P}}$ it is necessary to start every iteration round with the lowest partition set again. Differential fixpoint computation, however, becomes much more complicated since every delta relation must be locally considered for every partition set. Nevertheless, these partitions of rules in connection with the soft consequence operator play an important role in query evaluation and in the soft stratification approach which will be discussed in subsequent sections.

In order to increase efficiency of query evaluation the application of control expressions has been proposed, e.g. in [RSS94]. Such control expressions allow to describe any sequential application order of a set of deductive rules. Although these expressions were originally discussed for semi-positive databases only, the following example shows that rule ordering may as well be useful for an iterated fixpoint computation:

$$\begin{aligned}R_1 &: \ \texttt{p(X,Y,Y)} \leftarrow \texttt{p(X,Y,X)} \\ R_2 &: \ \texttt{p(X,Y,X)} \leftarrow \texttt{p(X,Z,X)} \wedge \texttt{p(Z,Y,X)} \wedge \neg\texttt{r(Y)} \\ R_3 &: \ \texttt{p(X,Y,X)} \leftarrow \texttt{e(X,Y)} \wedge \texttt{a(Y)}\end{aligned}$$

$$\begin{aligned}R_4 &: \ \texttt{q(X)} \leftarrow \texttt{e(X,Y)} \wedge \texttt{a(Y)} \\ R_5 &: \ \texttt{r(Y)} \leftarrow \texttt{a(Y)}\end{aligned}$$

Relation p negatively depends on the derived relation r and contains the transitive closure of the extensional relation e in the first two positions. The third parameter of p is used to record all values in e. Suppose these rules are partitioned using a given stratification and subsequently applied to an underlying relational database. Before the derived relations are materialized, several optimization steps are executed, e.g. algebraic simplification is performed intended to improve the cost of rule evaluation independent of the actual data or physical structure of the data.

A partition $\mathcal{P}$ induced by a stratification which separates the evaluation of the derived relations p and r could be $\mathcal{P} = P_1 \cup P_2$ with $P_1 = \{R_4, R_5\}$ and $P_2 = \{R_1, R_2, R_3\}$. In a semi-naive bottom-up evaluation of P_2, however, it might be more efficient to apply R_1 only once after R_2, R_3 have been evaluated completely in order to avoid the repeated application of R_1 for each newly inferred p-fact. Additionally, the common subexpression in the rule body of R_3 and R_4 could be more efficiently processed if both rules were placed in the same partition component. Thus, the partition $\mathcal{P}' = P_1 \cup P_2 \cup P_3$ with $P_1 = \{R_3, R_4, R_5\}$, $P_2 = \{R_2\}$ and $P_3 = \{R_1\}$ seems to be more suitable for evaluation in spite of not being directly induced by the given stratification. Only necessary dependencies between rules ought to be considered resulting in a partial ordering of the rule set rather than a mapping from the relations into strata. This shift from predicate dependencies to rule dependencies seems to be more adequate when actually computing the iterated fixpoint and already points to the basic idea of the soft stratification approach.

In contrast to stratifiable databases, the well-founded model of an unstratifiable rule set is not necessarily a total model such that a fixpoint-based computation of positive conclusions only is not sufficient. Instead at least two of the three sets of positive, negative and undefined conclusions have to be determined while the third set again can be implicitly derived by complementing the two computed sets of conclusions with respect to the given Herbrand base. A possible approach for computing the well-founded model of general deductive databases is the alternating fixpoint computation by Van Gelder [vG89, vG93]. It will be presented in the following section while an efficient transformation-based approach to general well-founded model computation will be discussed in Chapter 6.

3.3 Alternating Fixpoint Computation

In this section we discuss how the three valued well-founded model of arbitrary, i.e., possibly unstratifiable, deductive databases can be determined by means of fixpoint computations. The reason for dealing with this most general class of databases is twofold: On the one hand, it is known from [Kol91] that unstratifiable (function-free) databases are strictly more expressive than stratifiable ones and that there are actually interesting queries not expressible by stratifiable

databases. On the other hand, unstratifiable rule sets may result from rewriting techniques which transform an originally stratifiable rule set into a new rule set which might be more suited for performing certain database tasks. A well-known example is the Magic Sets transformation for query evaluation which may result in unstratifiable rules if applied to an originally stratifiable rule set.

As far as the Magic Sets approach is concerned, however, our soft stratification approach for Magic Sets transformed rules (cf. Chapter 4) avoids the expensive computation of the well-founded model for arbitrarily unstratifiable databases using the soft consequence operator. Additionally, all further transformation-based approaches presented in subsequent sections of this thesis result in softly stratifiable rules and thus allow the application of the soft stratification approach which is more efficient than general well-founded model computation.

Despite of these results, it is nevertheless reasonable to provide a general mechanism for computing the well-founded model of arbitrary databases. Such a component is most flexible as it can be exploited for any deductive database service requiring the materialization of a rewritten database. In fact, it subsumes even approaches where Magic Sets is applied to a certain class of unstratifiable rules on which this transformation is still known to be sound [KSS95].

Bottom-up approaches to the computation of well-founded models for arbitrary databases have been proposed in [KSS91, KSS95, SNV95, Bry89, BZF96] whereas other related approaches like [Ros90, LR92, RSS92] restrict the considered database class in the one or the other way. The approach by Kemp, Srivasta, and Stuckey in [KSS91, KSS95] is a direct implementation of the alternating fixpoint semantics by Van Gelder [vG89, vG93]. This method is essentially composed of least Herbrand model computations which are arranged in an appropriate order. Thus the transformation-based approach to semi-naive evaluation presented above may be used for their computation. The goal of this section is to present the alternating fixpoint computation on the basis of the work in [KSS91, KSS95]. In Chapter 6 we enhance this method further by providing a transformation-based approach to alternating fixpoint materialization on the basis of soft stratification, update propagation [Beh01], and soft update propagation.

For the sake of completeness, Section 3.3.1 starts by generally explaining how the well-founded model of deductive databases can be computed. We provide an example for introducing the alternating fixpoint computation according to the definition given in [vG93]. This approach has the advantage that all relevant derivation steps are made with respect to explicitly given sets of positive and negative conclusions providing an intuitive understanding of the alternating computation processes. However, as this approach utilizes negative conclusions, it is not particularly suitable for a direct implementation. Therefore, we finish this section by presenting a slightly modified formulation which is solely based on positive facts leaving negative conclusions implicit. In Section 3.3.2 we describe

the approach of [KSS91, KSS95] which is based on the reformulated alternating fixpoint definition.

3.3.1 Introduction to AFP Computation

The basic idea of alternating fixpoint computation [vG89, vG93] is to repeatedly compute fixpoints of the given database, each time evaluating negative literals with respect to the complement of the previously obtained fixpoint. Assuming a fixed semantics for negative literals, even unstratifiable databases are reduced to semi-positive ones, such that two-valued fixpoint semantics is applicable. The subsequently performed fixpoint computations alternately yield underestimates and overestimates of the set of actually true negative conclusions. The composition, however, of two such fixpoint computations is monotonic. Starting from an empty set of negative literals, the set of negative conclusions is constructed monotonically.

In order to work on negative conclusions, the stable consequence operator $\widetilde{T}_{\mathcal{R},\mathcal{N}}(\mathcal{I}^+)$ is used which computes the set of all positive conclusions derivable from $\mathcal{R}$ using the input set of positive literals in $\mathcal{I}^+$ and the fixed set of negative literals $\mathcal{N}$. During an application of $\widetilde{T}_{\mathcal{R},\mathcal{N}}$, a negative literal $\neg A$ is considered true if $\neg A$ is present in $\mathcal{N}$. Based on this operator we define the alternating fixpoint model $\widetilde{\mathcal{M}}_{\mathcal{D}}$ according to the characterization given in [vG89] where the author additionally proved it to be equivalent to the well-founded model $\mathcal{M}_{\mathcal{D}}$ of $\mathcal{D}$.

Definition 3.5 (VG Alternating Fixpoint Model) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, $\mathcal{I}^+, \mathcal{I}^- \subseteq \mathcal{H}_{\mathcal{D}}$ sets of ground atoms, $\mathcal{N}$ the negated set of atoms in $\mathcal{I}^-$, i.e., $\mathcal{N} = \neg \cdot \mathcal{I}^-$ and $[[\mathcal{R}]]_{\mathcal{I}^+}$ the set of all ground instances of rules in $\mathcal{R}$ with respect to the set $\mathcal{I}^+$. Then we define*

1. *the stable consequence operator $\widetilde{T}_{\mathcal{R},\mathcal{N}}$ as*

$$\widetilde{T}_{\mathcal{R},\mathcal{N}}(\mathcal{I}^+) := \{H \mid \exists r \in [[\mathcal{R}]]_{\mathcal{I}^+} : r \equiv H \leftarrow A_1 \wedge \ldots \wedge A_n \wedge \neg B_1 \wedge \ldots \wedge \neg B_m \text{ such that } A_i \in \mathcal{I}^+ \text{ for all positive literals } A_i \text{ and } \neg B_j \in \mathcal{N} \text{ for all negative literals } B_j\},$$

2. *the stability consequence transformations $S_{\mathcal{D}}$, $\widetilde{S}_{\mathcal{D}}$ as*

$$S_{\mathcal{D}}(\mathcal{N}) := \texttt{lfp}(\widetilde{T}_{\mathcal{R},\mathcal{N}}, \mathcal{F}), \text{ and}$$

$$\widetilde{S}_{\mathcal{D}}(\mathcal{N}) := \neg \cdot \overline{S_{\mathcal{D}}(\mathcal{N})},$$

where given a set of negative conclusions $\mathcal{N}$ the transformation $S_{\mathcal{D}}(\mathcal{N})$ returns a set of positive conclusions while $\widetilde{S}_{\mathcal{D}}(\mathcal{N})$ yields a set of negative conclusions,

3. and the VG Alternating Fixpoint Model $\widetilde{\mathcal{M}}_{\mathcal{D}}$ *as*

$$\widetilde{\mathcal{M}}_{\mathcal{D}} := S_{\mathcal{D}}(\mathtt{lfp}(\widetilde{S}^2_{\mathcal{D}}, \emptyset)) \uplus \mathtt{lfp}(\widetilde{S}^2_{\mathcal{D}}, \emptyset)$$

where $\widetilde{S}^2_{\mathcal{D}}$ *denotes the nested application of the stability consequence transformation* $\widetilde{S}_{\mathcal{D}}$*, i.e.,* $\widetilde{S}^2_{\mathcal{D}}(\mathcal{N}) = \widetilde{S}_{\mathcal{D}}(\widetilde{S}_{\mathcal{D}}(\mathcal{N}))$*.*

In the following we describe the course of computing the alternating fixpoint using the following sample database.

Example 3.1 *Consider the following unstratifiable deductive database* $\mathcal{D} = \langle \mathcal{R}, \mathcal{F} \rangle$ *consisting of the rule*

$$\mathtt{e(X)} \leftarrow \mathtt{succ(X,Y)} \wedge \neg \mathtt{e(Y)}$$

and the facts

```
succ(0,1),succ(1,2),succ(2,3),succ(3,4),succ(4,5).
```

The deductive rule defines the 'even' numbers between 0 and 5. From the results in [Kol91] it can be inferred that the intended meaning of this database is not expressible by any (function-free) stratifiable database [Ros90]. The implicit state of $\mathcal{D}$*, or its three-valued well-founded model, is given by* $\mathcal{M}_{\mathcal{D}} = \mathcal{I}^+_{\mathcal{D}} \uplus \neg \cdot \mathcal{I}^-_{\mathcal{D}}$ *where the set of true positive conclusions* $\mathcal{I}^+_{\mathcal{D}}$ *consists of the fact base* $\mathcal{F}$ *as well as the derived facts* $\{e(0), e(2), e(4)\}$*. The set of true negative conclusions* $\neg \cdot \mathcal{I}^-_{\mathcal{D}}$ *comprises the negations of all non-existing succ-facts as well as* $\{\neg e(1), \neg e(3), \neg e(5)\}$*.*

At the beginning of the alternating fixpoint computation we assume all negative literals to be false, i.e., $\mathcal{N} = \emptyset$. Consequently, facts can only be derived from rules which contain no negative body literals. As we consider one negative rule in the given example only, the first fixpoint coincides with the given fact base

$$\text{lfp } (\widetilde{T}_{\mathcal{R},\emptyset}, \mathcal{F}) = \mathcal{F} \qquad (= DT^1)$$

where the least fixpoint of the operator $\widetilde{T}_{\mathcal{R},\mathcal{N}}$ gives the set of all positive conclusions derivable from $\mathcal{D}$ and the fixed set of negative literals $\mathcal{N}$. This first application of $\widetilde{T}_{\mathcal{R},\mathcal{N}}$ with respect to an overestimation of the set of true negative conclusions leads to the set $DT^1 \subseteq \mathcal{I}^+_{\mathcal{D}}$ including only facts which are "definitely true". However, for all ground atoms of the Herbrand base of $\mathcal{D}$ which are not included in DT^1 it is not yet known whether they are true, false, or undefined. These are `e(0)`, `e(1)`, `e(2)`, `e(3)`, `e(4)`, and `e(5)` as well as all `succ`-facts which are not in $\mathcal{F}$. In the next step we assume all these atoms to be false and hence the respective negated literals included in the conjugate of DT^1, i.e., $\neg \cdot \overline{DT^1} = N_0 \cup \{\neg e(0), \neg e(1), \neg e(2), \neg e(3), \neg e(4), \neg e(5)\}$ where N_0 contains all `succ`-facts not in $\mathcal{F}$, to be true. Under these conditions the resulting fixpoint is given by

$$\text{lfp } (\widetilde{T}_{\mathcal{R},N_0\cup\neg\cdot\{e(0),e(1),e(2),e(3),e(4),e(5)\}},\mathcal{F}) = \mathcal{F}\cup\{e(0),\ldots,e(3),e(4)\}\ (=NDF^1)$$

which is a superset of $\mathcal{I}^+_{\mathcal{D}}$, since the evaluation of negative literals is based on an overestimation of the set of true negative conclusions. Therefore, the set NDF^1 may comprise facts which are no true positive conclusions. However, all facts not included (for `e` this is `e(5)`) are known to be definitely false, as they could not be derived under the most general overestimation of the set of negative conclusions. The identifier NDF is used to indicate that the set NDF^i contains facts which are "not known to be definitely false" at the current computation step.

During the next fixpoint computation all definitely true facts are determined which can be derived from the given database and the (now known) definitely true negative literal $\neg e(5)$.

$$\text{lfp } (\widetilde{T}_{\mathcal{R},N_0\cup\neg\cdot\{e(5)\}},\mathcal{F}) = \mathcal{F}\cup\{e(4)\} \qquad (=DT^2)$$

The set DT^2 again includes true positive conclusions only, as it is guaranteed that only negative literals known to be definitely true may have induced further derivations. The subsequent applications produce the following sequence:

$$\begin{aligned}
&\text{lfp } (\widetilde{T}_{\mathcal{R},N_0\cup\neg\cdot\{e(0),e(1),e(2),e(3),e(5)\}},\mathcal{F}) &&= \mathcal{F}\cup\{e(0),e(1),e(2),e(4)\} && (=NDF^2)\\
&\text{lfp } (\widetilde{T}_{\mathcal{R},N_0\cup\neg\cdot\{e(3),e(5)\}},\mathcal{F}) &&= \mathcal{F}\cup\{e(2),e(4)\} && (=DT^3)\\
&\text{lfp } (\widetilde{T}_{\mathcal{R},N_0\cup\neg\cdot\{e(0),e(1),e(3),e(5)\}},\mathcal{F}) &&= \mathcal{F}\cup\{e(0),e(2),e(4)\} && (=NDF^3)\\
&\text{lfp } (\widetilde{T}_{\mathcal{R},N_0\cup\neg\cdot\{e(1),e(3),e(5)\}},\mathcal{F}) &&= \mathcal{F}\cup\{e(0),e(2),e(4)\} && (=DT^4)\\
&\text{lfp } (\widetilde{T}_{\mathcal{R},N_0\cup\neg\cdot\{e(1),e(3),e(5)\}},\mathcal{F}) &&= \mathcal{F}\cup\{e(0),e(2),e(4)\} && (=NDF^4)\\
&\text{lfp } (\widetilde{T}_{\mathcal{R},N_0\cup\neg\cdot\{e(1),e(3),e(5)\}},\mathcal{F}) &&= \mathcal{F}\cup\{e(0),e(2),e(4)\} && (=DT^5)
\end{aligned}$$

The calculation alternates between the computation of subsets of definitely true facts (DT^i) and the computation of supersets of not definitely false facts (NDF^i) using subsets of definitely false and supersets of not definitely true facts in $\mathcal{N}$, respectively. The composition of two steps is monotonic, i.e., the set of true facts as well as the set of definitely false facts is monotonically increasing. A fixpoint has been reached when the set of definitely false facts does not change anymore, i.e., until $\neg\cdot\overline{NDF^i} = \neg\cdot\overline{NDF^{i-1}}$. In the example the VG Alternating Fixpoint Model is then given by

$$\widetilde{\mathcal{M}}_{\mathcal{D}} = DT^5 \uplus \neg\cdot\overline{NDF^4}$$

with the set of true conclusions DT^5=$\mathcal{F}\cup\{e(0),e(2),e(4)\}$, the set of true negative conclusions $\neg\cdot\overline{NDF^4}$=$\{\neg e(1),\neg e(3),\neg e(5),\neg succ(1,1),\ldots\}$ and the empty set of undefined facts.

A graphical representation of the general course of alternating fixpoint computation is presented in Figure 3.1 showing that the computation runs in two alternating phases. During the first phase subsets of definitely true facts (DT^i)

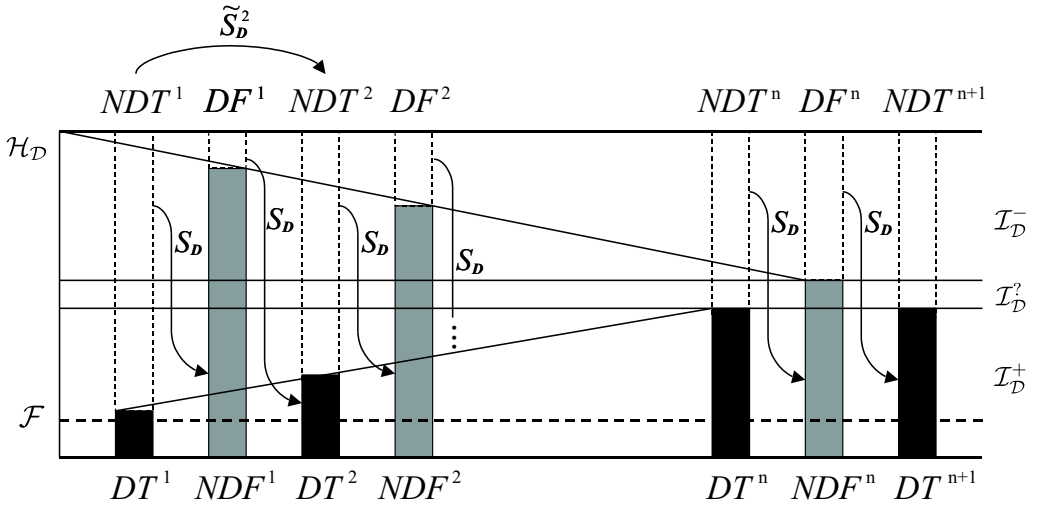


Figure 3.1: Alternating fixpoint computation

are calculated from subsets of definitely false facts ($DF^i = \overline{NDF^i}$). In contrast to this, the second phase determines supersets of definitely true facts (NDF^i) from supersets of false facts, i.e., those not known to be definitely true ($NDT^i = \overline{DT^i}$). The composition of two phases is monotonic, i.e., the sets of true facts (DT^i) as well as those of definitely false facts (DF^i) are monotonically increasing. When these sets do not change anymore a fixpoint is reached which partitions the given Herbrand Base $\mathcal{H}_\mathcal{D}$ into the set of definitely true facts $\mathcal{I}^+_\mathcal{D} = DT^{n+1}$, the set of definitely false facts $\mathcal{I}^-_\mathcal{D} = DF^n$ ($= \overline{NDF^n}$), and the set of undefined atoms $\mathcal{I}^?_\mathcal{D} = NDF^n \setminus DT^{n+1}$.

3.3.2 Computing the Well-founded Model

The alternating fixpoint semantics as proposed by Van Gelder [vG93] is not particularly well-suited for direct implementation, as it works on negative conclusions (i.e., definitely false facts as well as not definitely true facts). Instead, it would be preferable to deal with positive facts only, since these can be more easily represented in and retrieved from a database. Therefore we will reformulate alternating fixpoint computation as proposed by Van Gelder such that it deals with positive conclusions only. Such a reformulation has been presented in [KSS95] where the sets of definitely true facts and not definitely false facts are explicitly stored and only the complement of the latter is used to refer to true negative conclusions implicitly.

Definition 3.6 (KSS Alternating Fixpoint Model) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, $\mathcal{I}^+, \mathcal{I}^- \subseteq \mathcal{H}_\mathcal{D}$ sets of ground atoms, and $[[\mathcal{R}]]_{\mathcal{I}^+}$ the set of all ground instances of rules in $\mathcal{R}$ with respect to the set $\mathcal{I}^+$. Then we define*

1. *the eventual consequence operator* $\widehat{T}_{\mathcal{R}}\langle\mathcal{I}^-\rangle$ *as*

$$\widehat{T}_{\mathcal{R}}\langle\mathcal{I}^-\rangle(\mathcal{I}^+) := \{H \mid \exists r \in [[\mathcal{R}]]_{\mathcal{I}^+} : r \equiv H \leftarrow L_1 \wedge \ldots \wedge L_n \\ \text{such that } L_i \in \mathcal{I}^+ \text{ for all positive literals } L_i \\ \text{and } L \notin \mathcal{I}^- \text{ for all negative literals } L_j \equiv \neg L\},$$

2. *the eventual consequence transformation* $\widehat{S}_{\mathcal{D}}$ *as*

$$\widehat{S}_{\mathcal{D}}(\mathcal{I}^-) := \texttt{lfp}(\widehat{T}_{\mathcal{R}}\langle\mathcal{I}^-\rangle, \mathcal{F}),$$

3. *and the KSS Alternating Fixpoint Model* $\widehat{M}_{\mathcal{D}}$ *as*

$$\widehat{M}_{\mathcal{D}} := \texttt{lfp}\ (\widehat{S}^2_{\mathcal{D}}, \emptyset) \uplus \neg \cdot \overline{\widehat{S}^2_{\mathcal{D}}(\texttt{lfp}\ (\widehat{S}^2_{\mathcal{D}}, \emptyset))}\ ,$$

where $\widehat{S}^2_{\mathcal{D}}$ *denotes the nested application of the eventual consequence transformation, i.e.,* $\widehat{S}^2_{\mathcal{D}}(\mathcal{I}^-) = \widehat{S}_{\mathcal{D}}(\widehat{S}_{\mathcal{D}}(\mathcal{I}^-))$.

The following theorem shows that both forms of alternating fixpoint computation correctly yield the well-founded model $\mathcal{M}_{\mathcal{D}}$ of a given database $\mathcal{D}$.

Theorem 3.1 *Let* $\mathcal{D} = \langle\mathcal{F}, \mathcal{R}\rangle$ *be a deductive database. The KSS Alternating Fixpoint Model* $\widehat{M}_{\mathcal{D}}$ *of* $\mathcal{D}$ *is identical with the VG Alternating Fixpoint Model* $\widetilde{M}_{\mathcal{D}}$ *of* $\mathcal{D}$ *which itself coincides with well-founded model* $\mathcal{M}_{\mathcal{D}}$ *of* $\mathcal{D}$.

Proof: cf. [Gri97, p. 108-109]. □

In contrast to the stable consequence operator $\widetilde{T}_{\mathcal{R}\cup N}$ used in the example above, the eventual consequence operator $\widehat{T}_{\mathcal{R}}\langle\mathcal{I}^-\rangle$ operates on positive atoms only. It evaluates negative literals $\neg A$ by checking whether A is not in $\mathcal{I}^-$ rather than by testing whether $\neg A$ is in a set of negative literals. Therefore, the eventual consequence operator $\widehat{T}_{\mathcal{R}}\langle\mathcal{I}^-\rangle$ only implicitly refers to the conjugate of $\mathcal{I}^-$ when evaluating a negative literal. It is obvious that for any database $\mathcal{D}$ and any sets of ground atoms $\mathcal{I}^+, \mathcal{I}^- \subseteq \mathcal{H}_{\mathcal{D}}$ both operators obtain the same result:

$$\widehat{T}_{\mathcal{R}}\langle\mathcal{I}^-\rangle(\mathcal{I}^+) = \widetilde{T}_{\mathcal{R},\neg\cdot\overline{\mathcal{I}^-}}(\mathcal{I}^+).$$

In addition, the least fixpoint of $\widehat{S}^2_{\mathcal{D}}$ is a set of true positive conclusions rather than a set of true negative conclusions as given by the least fixpoint of $\widetilde{S}^2_{\mathcal{D}}$ in the alternating fixpoint approach defined by Van Gelder.

The computation of the KSS Alternating Fixpoint Model starts with an empty set of positive conclusions (implying that all negative literals are assumed to be true) rather than with an empty set of negative conclusions (implying that

Algorithm 2: Alternating fixpoint computation

$i := 0$;
$DT^0 := \emptyset$;
repeat
  $i := i + 1$;
  $NDF^i := \mathtt{lfp}(\widehat{T}_{\mathcal{R}}\langle DT^{i-1}\rangle, \mathcal{F})$;
  $DT^i := \mathtt{lfp}(\widehat{T}_{\mathcal{R}}\langle NDF^i\rangle, \mathcal{F})$;
until $DT^i = DT^{i-1}$;
$DT := DT^i$;
$NDF := NDF^i$;

all negative literals are assumed to be false). In this way, the first application of $\widehat{S}_{\mathcal{D}}(\mathcal{I}^-)$ obtains a superset of the set of true positive conclusions, i.e., not definitely false facts (NDF) and the second a subset of the set of definitely true facts (DT). Hence the order of fixpoint computations is exchanged with respect to the original definition of Van Gelder which starts with computing a subset of the definitely true facts.

In the following we will introduce the algorithm presented in [KSS91] for computing the well-founded model of a deductive database. The reason for presenting this approach is twofold: On the one hand, this algorithm serves as a basis for the doubled programm approach which will be discussed in Section 6.1. On the other hand, we will show how to improve the efficiency of this approach by incorporating update propagation and soft stratification in Section 6.2. We begin by describing the general course of alternating fixpoint computation as introduced in Definition 3.6 using the simple iteration scheme presented in Algorithm 2 and go on by refining this scheme.

The scheme in Algorithm 2 organizes alternating fixpoint computation as follows: At the beginning, the set DT^0 is initialized with the empty set of true positive conclusions. Afterwards, in each round of the iteration phase the eventual consequence transformation $\widehat{S}^2_{\mathcal{D}}$ is implemented by first computing not definitely false facts and then definitely true facts, each time employing the previously obtained fixpoint for evaluating negative literals. The iteration is continued until the set of definitely true facts does not change anymore. The well-founded model $\mathcal{M}_{\mathcal{D}}$ is then given by $\mathcal{M}_{\mathcal{D}} = DT \cup \neg \cdot \overline{NDF}$. In contrast to the original definition of Van Gelder, another fixpoint computation for obtaining the final set of not definitely false facts is not performed because the identity of DT^i and DT^{i-1} implies that the sets NDF^i and NDF^{i+1} are equal as well.

Up till now we have considered alternating fixpoint computation for non-layered rule sets only. However, efficiency can be significantly increased if the rule set can be partitioned into layers in an appropriate way. Firstly, evaluating

Algorithm 3: Iterated alternating fixpoint computation

$NDF_0 := \mathcal{F}$;
$DT_0 := \mathcal{F}$;
for each layer $l = 1, \ldots, m$ of $\mathcal{R}$ **do**
  $i := 0$;
  $DT_l^0 := \mathtt{lfp}(\widehat{T}_{\mathcal{R}_l^\circ \cup \mathcal{R}_l^\times}\langle NDF_{l-1}\rangle, DT_{l-1})$;
  repeat
    $i := i + 1$;
    $NDF_l^i := \mathtt{lfp}(\widehat{T}_{\mathcal{R}_l}\langle DT_l^{i-1}\rangle, NDF_{l-1} \cup DT_l^{i-1})$;
    $DT_l^i := \mathtt{lfp}(\widehat{T}_{\mathcal{R}_l^\times \cup \mathcal{R}_l^*}\langle NDF_l^i\rangle, DT_l^{i-1})$;
  until $DT_l^i = DT_l^{i-1}$;
  $NDF_l := NDF_l^i$;
  $DT_l := DT_l^i$;
end for
$DT := DT_m$;
$NDF := NDF_m$;

rules in a certain order may avoid redundant derivations. In addition, the identification of different rule classes within each layer of the original unstratifiable rule set completely excludes certain rules from evaluation. When evaluating the alternating fixpoint in layers, we have to take into account that there are already sets of definitely true and not definitely false facts computed for predicates of lower layers. As for iterated fixpoint computation, these sets form the basis of calculations in higher layers.

When computing not definitely false facts, positive literals referring to predicates of lower strata are evaluated with respect to not definitely false facts NDF_{l-1} already derived during materialization of layers up to $l-1$, and negative literals with respect to the current set of definitely true facts DT_l^{i-1} which already includes all definitely true facts DT_{l-1} computed in lower layers. In contrast to this, determining definitely true facts requires positive literals to be checked against definitely true facts DT_l^{i-1} computed so far, and negative ones against the current set of not definitely false facts NDF_l^i. Note that the set DT_{l-1} is entirely included in the sets DT_{l-1}^{i-1} for $i > 0$ while the set NDF_{l-1} represents a superset of the sets NDF_l^i for $i > 0$. Hence, we obtain the scheme in Algorithm 3 of iterated alternating fixpoint computation.

The iteration scheme of Algorithm 3 still includes further improvements in comparison to the original scheme of Algorithm 2. In Algorithm 2 the computations of DT- and NDF-facts starts with the assumption that no atom is true. However, it is preferable to initialize DT_0 and NDF_0 with the base facts $\mathcal{F}$ which are known to be unconditionally true and a subset of the final sets of definitely

true facts DT as well as not definitely false facts NDF. In addition, it is advantageous to initialize the definitely true facts of each layer DT_l^0 with all facts that can be derived while assuming that all negative literals of the current layer are false. This can be achieved by restricting the given rule set to all those rules not including any unstratified negation and computing the fixpoint

$$DT_l^0 := \texttt{lfp}(\widehat{T}_{\mathcal{R}_l^\circ \cup \mathcal{R}_l^\times}\langle NDF_{l-1}\rangle, DT_{l-1}).$$

As the set of definitely true facts is monotonically increasing, it is not necessary to recompute all facts which have been obtained in previous iteration rounds. Instead, we can start with the previously computed DT_l^{i-1} set when determining the i-th set of definitely true facts DT_l^i. In addition, we can ignore all rules in $\mathcal{R}_l^\circ$ as they cannot produce any facts not yet included in DT_l^{i-1}. Thus, the set DT_l^i can be calculated by the following expression:

$$DT_l^i := \texttt{lfp}(\widehat{T}_{\mathcal{R}_l^\times \cup \mathcal{R}_l^*}\langle NDF_l^i\rangle, DT_l^{i-1}).$$

It is not possible to apply the same technique for not definitely false facts as these sets are decreasing in each round. However, each set of not definitely false facts forms a superset of the definitely true facts obtained in the previous iteration round. Thus, we can compute not definitely false facts starting from those already known to be true, i.e.,

$$NDF^i := \texttt{lfp}(\widehat{T}_{\mathcal{R}_l}\langle DT_l^{i-1}\rangle, NDF_{l-1} \cup DT_l^{i-1}).$$

This time we have to keep the explicit reference to NDF_{l-1} as this set is possibly not entirely covered by DT_l^{i-1}. For the same reason we cannot omit the rules in $\mathcal{R}^\circ$, as they may still lead to new consequences not contained in DT_l^{i-1}. The iteration scheme in Algorithm 3 basically yields the approach presented in [KSS91]. However, we will not present the entire algorithm in this place, as we will still propose another improvement in Chapter 6.

Chapter 4

Query Evaluation

In Chapter 3, fixpoint algorithms have been introduced, each of them suitable for materializing the implicit state of a certain class of databases only. In this chapter, we will concentrate on stratifiable databases. We will show how the corresponding algorithms can be employed for query evaluation. Stratifiable databases represent the most important database class from a practical point of view as their view concept directly corresponds to those views permitted by SQL:1999.

The most simple approach to answering a query against a stratifiable deductive database would be to determine the corresponding well-founded model by means of iterated fixpoint computation and to select respective answer tuples after rule processing has terminated. This kind of bottom-up computation of answers can naturally employ the existing optimization techniques developed for relational databases. However, proceeding this naive way has the well-known disadvantage that most facts produced during the course of materialization are not relevant for answering the given query. Top-down methods on the other hand perform query evaluation in a goal-directed manner such that materialization is very naturally limited to relevant parts of the given database, only. However, a pure top-down approach, as proposed for example by methods like OLDT resolution [TS86], QSQ [Vie88] or QRGT [Ull89], has the disadvantage that particularly in presence of recursion an expensive 'logic' control is needed in order to provide completeness and soundness. Therefore, various rewriting techniques for query evaluation in deductive databases have been proposed which combine the advantages of top-down and bottom-up approaches. The basic idea is to rewrite deductive rules with respect to a given query such that bottom-up materialization is performed in a goal-directed way cutting down the number of irrelevant facts generated.

The extensive research on such rewriting techniques originated from the seminal proposals of the Magic Sets approach [BR86] and the Alexander Method [RLK86]. Since then, many proposals have been made aiming at a refinement of the original methods. Among others, there are Generalized Magic Sets [BR91], Magic Templates [Ram91], Generalized Supplementary Magic Sets [BR91], Magic Counting [SZ87b], Generalized Magic Counting [BR91], Generalized Supplemen-

tary Magic Counting [BR91], Magic Conditions [MFPR96], Minimagic Sets [SZ87a], Envelopes [Sag90], SLDMagic [Bra96] and Alexander Templates [Sek89]. Further publications, e.g. [BPRM91, Che93, KSS95, Mor93] investigate the applicability of Magic Sets to stratifiable or even unstratifiable databases. In [Bry90b] it has been shown that query evaluation via Magic Sets is basically equivalent to methods like OLDT resolution and QSQ. This shows the expressiveness of a deductive rule rewriting technique like the Magic Sets approach in comparison with other strategies which dynamically perform all optimizations during the course of evaluation.

In the following we will focus on Magic Sets, as this approach has been accepted as a kind of standard in the field. As the Magic Set rewriting of stratifiable rules may lead to unstratifiable rule sets, we propose a bottom-up evaluation method based on the weak consequence operator [KP88] in order to compute the total well-founded model of magic rules. We show that its application in combination with the concept weak stratification, however, may lead to a set of answers which is neither sound nor complete with respect to the well-founded model. This problem is cured by introducing the new concept soft stratification instead [Beh03]. The overall result of this chapter then is a bottom-up evaluation method for efficiently materializing the implicit state of this class of unstratifiable rules. In subsequent chapters it will be shown that this class of deductive rules plays an important role for transformation-based update propagation and view updating methods as well.

4.1 Magic Sets

We refrain from presenting the Magic Sets approach in detail as introduced in [BR91] or [Ram91], but rather present a simplified version of the Magic Templates algorithm [Ram91] originally proposed by Naughton and Ramakrishnan in [NR91]. Magic Sets rewriting is a two-step transformation in which the first phase consists of constructing an *adorned rule set* [Ull85], while the second phase consists of the actual Magic Sets rewriting. In Section 4.1.1, it will be shown how the adorned rule set can be derived from the original database with respect to the binding pattern of a given query and a choice of *sideways information passing (sip) strategies* [Ram91]. Section 4.1.2 presents the second phase of Magic Sets where the adorned rules are rewritten such that bottom-up materialization of the resulting database implements a top-down evaluation of the original query on the original database. For this purpose, each adorned rule is extended by a magic literal restricting the evaluation of the rule to the given binding in the adornment of the rule's head.

4.1.1 The Adorned Database

The first step of the Magic Sets transformation is to determine the adorned rule set. Within an adorned rule set the predicate symbol of each derived literal is associated with an adornment, which is a string consisting of the symbols 'b' and 'f' representing bound and free argument positions when the literal is to be evaluated.

Definition 4.1 (Adorned Literal) *Let $\mathcal{R}$ be a deductive rule set and $p(t_1, \ldots, t_n)$ a derived literal appearing in a rule in $\mathcal{R}$. Then the adorned literal of $p(t_1, \ldots, t_n)$ is defined as*

$$p_{ad}(t_1, \ldots, t_n)$$

where the adornment ad is a string consisting of the symbols $a_1 \ldots a_n$ which is defined as follows: if t_i is bound to a constant when $p(t_1, \ldots, t_n)$ is due to evaluation then $a_i :=' b'$, otherwise $a_i :=' f'$ for unbound attributes.

The adorned version of the deductive rules is constructed with respect to an adorned query and a selected sip strategy which basically determines for each rule the order in which the body literals are to be evaluated. As an example consider the following rule set

$$\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}(\mathtt{X},\mathtt{Y}) \qquad \mathtt{e}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{b}(\mathtt{X},\mathtt{Y})$$
$$\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}(\mathtt{Z},\mathtt{Y}) \qquad \mathtt{e}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{c}(\mathtt{X},\mathtt{Y})$$

and the query $? - p(1, Y)$ asking for all nodes reachable from node 1. The construction of the adorned rule set starts with determining the adornment of the query:

$$? - p_{bf}(1, Y).$$

In the next step, all deductive rules are selected whose heads unify with the original query and the derived literals in their bodies are considered as sub-goals to be solved. A chosen sip strategy determines the order in which these sub-goals are to be evaluated and which bindings are passed on to the next sub-goal. The evaluation of a sub-goal is performed in the same way as for the original query, starting with the determination of the corresponding adorned literal and the deductive rules whose heads unify with the current sub-goal. Assuming a left-to-right sip strategy for all rules, the adorned rule set with respect to the above example and the adorned query $p_{bf}(1, Y)$ is as follows:

$$\mathtt{p}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \qquad \mathtt{e}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{b}(\mathtt{X},\mathtt{Y})$$
$$\mathtt{p}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}_{\mathtt{bf}}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}_{\mathtt{bf}}(\mathtt{Z},\mathtt{Y}) \qquad \mathtt{e}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{c}(\mathtt{X},\mathtt{Y})$$

In the course of a top-down evaluation of the query $p_{bf}(1, Y)$, each derived literal would be called with the binding pattern encoded in its adornment when rule bodies are evaluated from left-to-right. In case a derived literal for a given sip strategy is called with different binding patterns, the adorned rule set contains variants of the rules for the respective predicate, each of them adorned with a different binding pattern. Note that in the example above we have considered a full sip strategy in which all bindings are passed on to the next sub-goal. It is, however, also possible to consider so-called partial sip strategies which pass on a subset of generated bindings only or no bindings at all. These strategies allow for avoiding redundant computations by taking subsumption effects into account.

The adornment phase is an essential prerequisite for the second phase and already has a strong influence on the overall performance of the Magic Sets approach. Therefore, several optimizations have been proposed in order to either minimize the number of adorned rules or to improve information flow depicted in the adorned rule set. One possible optimization for minimizing the number of adorned rules is to require the input set to satisfy the *unique binding property* [Ram91]. This means that during the top-down analysis according to the selected sip strategy no predicate would be called with different binding patterns avoiding the duplication of rules as mentioned above. For improving information flow Ullman proposed to rectify the input rule set in order to handle variable-to-variable bindings [Ull89]. For the following discussion, however, we will not consider these optimization techniques any further as they can be applied independently.

Another way of improving the quality of the adornment phase is given by the choice of sip strategy. Sip strategies can be generally divided into static and dynamic strategies [Beh00]. Static strategies determine the order in which the body literals are to be evaluated using a criterion that does not change during the subsequent rule evaluation, e.g. a left-to-right strategy. On the contrary, dynamic strategies use conditions which may change during the evaluation of Magic Sets transformed rules, e.g. relation size or selectivity of a join. Dynamic sip strategies form a basis of dynamic query evaluation and usually lead to a more complex evaluation process of magic rules than static strategies. In the following, however, we will concentrate on static sip strategies and especially on stratification problems which may arise if static strategies are applied to an originally stratifiable rule set. Note that the Magic Sets transformation is sound and complete with respect to the described answer set, if the intermediately obtained adorned rules are *adorned allowed* and the adornment rewriting is performed with respect to an *adorned allowed sip strategy* [BPRM91]. An adorned rule is called adorned allowed, if every variable appears in a positive body literal or a bound position of the binding pattern of the head. An allowed sip strategy requires additionally that the order in which body literals are to be evaluated still preserves the range-restriction property of negative literals. In the following we assume that

the adorned rules have been rewritten with respect to a static, adorned allowed sip strategy and satisfy the adorned allowedness property.

4.1.2 Magic Templates

During the second phase of the Magic Sets transformation the adorned rules are rewritten such that bottom-up materialization of the resulting database simulates a top-down evaluation of the original query on the original database. For this purpose, each adorned rule is extended with a magic literal restricting the evaluation of the rule to the given binding in the adornment of the rule's head. The magic predicates are defined by rules computing all values that would be passed in the sequence of body literals according to the sip strategy. The initial values corresponding to the query are given by the so-called magic seed. Before we present the Magic Sets rewriting more precisely, the next definitions specify how magic literals are constructed and how the seed is derived from the query.

Definition 4.2 (Magic Literal) *Let $A \equiv p_{ad}(\vec{x})$ be a positive adorned literal with adornment* `ad` *and* $\mathtt{bd}(\vec{x})$ *the sequence of variables within $\vec{x}$ indicated as bound in the adornment* `ad`*. Then the magic literal of A is defined by*

$$\mathtt{magic}(A) := m\_p_{ad}(\mathtt{bd}(\vec{x})).$$

If $A \equiv \neg p_{ad}(\vec{x})$ is a negative literal, then the magic literal of A is defined as $\mathtt{magic}(A) := m\_p_{ad}(\mathtt{bd}(\vec{x}))$.

Definition 4.3 (Seed/Seed Rule) *Let $Q \equiv p_{ad}(\vec{c})$ be a query with adornment* `ad` *and* $\mathtt{bd}(\vec{c})$ *the sequence of constants in $\vec{c}$ indicated as bound in the adornment* `ad`*. Then the seed of Q is defined by*

$$\mathtt{seed}(Q) := m\_s\_p_{ad}(\mathtt{bd}(\vec{c}))$$

and the corresponding seed rule is defined by

$$\mathtt{seed\_rule}(Q) := m\_p_{ad}(\vec{x}) \leftarrow m\_s\_p_{ad}(\vec{x})$$

where $\vec{x}$ is a vector of distinct variables $x_1, \ldots, x_n$ and n is the length of the sequence $\mathtt{bd}(\vec{x})$.

In order to simplify the definition of the Magic Sets transformation we assume that the body literals have already been ordered from left to right according to the selected sip strategy.

Definition 4.4 (Magic Rules) *Let $\mathcal{R}$ be a stratifiable deductive rule set, $Q \equiv p_{ad}(\vec{c})$ an adorned query with $p \in \mathtt{pred}(\mathcal{R})$, and $\mathcal{R}^Q$ the adorned rule set of $\mathcal{R}$ with respect to the query Q. The Magic Sets rewriting of $\mathcal{R}^Q$ yields the magic rules $\mathtt{ms}(\mathcal{R}^Q)$ defined as the smallest set satisfying the following conditions:*

1. *For each deductive rule* $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^Q$ *an answer rule of the form*

   $$A \leftarrow \texttt{magic}(A) \wedge L_1 \wedge \ldots \wedge L_n$$

   is in $\texttt{ms}(\mathcal{R}^Q)$.

2. *For each deductive rule* $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^Q$ *and each derived body literal* L_i $(1 \leq i \leq n)$ *a sub-query rule of the form*

   $$\texttt{magic}(L_i) \leftarrow \texttt{magic}(A) \wedge L_1 \wedge \ldots \wedge L_{i-1}$$

   is in $\texttt{ms}(\mathcal{R}^Q)$.

Note that the definition of Magic Rules solely depends on the predicate p and adornment ad of a given query $p_{ad}(\vec{c})$ but not on the constants within $\vec{c}$.

Definition 4.5 (Magic DB Transformation) *Let* $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ *be a stratifiable deductive database,* $Q \equiv p_{ad}(\vec{c})$ *an adorned query with* $p \in \texttt{pred}(\mathcal{R})$, *and* $\texttt{ms}(\mathcal{R}^Q)$ *the magic rule set of* $\mathcal{R}$ *with respect to the query* Q. *The Magic DB transformation of* $\mathcal{D}$ *with respect to* Q *then yields the deductive database* $\mathcal{D}^m = \langle \mathcal{F} \cup \{\texttt{seed}(Q)\},\ \texttt{ms}(\mathcal{R}^Q) \cup \{\texttt{seed\_rule}(Q)\} \rangle$.

Note that this definition of Magic Sets slightly differs from the one in [Ram91] as we add the additional magic seed rule to $\texttt{ms}(\mathcal{R}^Q)$ in order to keep the condition $\texttt{pred}(\mathcal{F}) \cap \texttt{pred}(\mathcal{R}) = \emptyset$ in Definition 2.7 of deductive databases satisfied. For our example above, the Magic DB transformation then yields the deductive rule set

$$\begin{array}{l}
\texttt{p}_{\texttt{bf}}(\texttt{X},\texttt{Y}) \leftarrow \texttt{m_p}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{e}(\texttt{X},\texttt{Y}) \\
\texttt{p}_{\texttt{bf}}(\texttt{X},\texttt{Y}) \leftarrow \texttt{m_p}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{e}(\texttt{X},\texttt{Z}) \wedge \texttt{p}_{\texttt{bf}}(\texttt{Z},\texttt{Y}) \\
\texttt{e}_{\texttt{bf}}(\texttt{X},\texttt{Y}) \leftarrow \texttt{m_e}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{b}(\texttt{X},\texttt{Y}) \\
\texttt{e}_{\texttt{bf}}(\texttt{X},\texttt{Y}) \leftarrow \texttt{m_e}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{c}(\texttt{X},\texttt{Y}) \\
\\
\texttt{m_p}_{\texttt{bf}}(\texttt{Z}) \leftarrow \texttt{m_p}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{e}(\texttt{X},\texttt{Z}) \\
\texttt{m_p}_{\texttt{bf}}(\texttt{X}) \leftarrow \texttt{m_s_p}_{\texttt{bf}}(\texttt{X}) \\
\texttt{m_e}_{\texttt{bf}}(\texttt{X}) \leftarrow \texttt{m_p}_{\texttt{bf}}(\texttt{X})
\end{array}$$

as well as the magic seed fact $\texttt{m\_s\_p}_{\texttt{bf}}(\texttt{1})$. The following theorem recalls the correctness of Magic Sets rewriting according to a given query Q by arguing that the answer set of Q with respect to the original database is equivalent to the answer set of the adorned query with respect to the magic rewritten database[1].

[1] For further details about the concept of answer equivalence we refer to [BPRM91, BNR+87, Mah88, Sag88, KK88].

Theorem 4.1 *Let $\mathcal{D}$ be a stratifiable database, Q a query to $\mathcal{D}$, $\mathcal{D}^m$ the database resulting from Magic DB transformation applied to $\mathcal{D}$ with respect to Q, and* $\texttt{ans}(\mathcal{M}_\mathcal{D}, Q)$ *the answer set of Q defined as* $\texttt{ans}(\mathcal{M}_\mathcal{D}, Q) := \{L \mid L \equiv Q\sigma,\ \sigma$ *is a ground substitution for all variables in Q and $L \in \mathcal{M}_\mathcal{D}\}$. Then the answer set of Q with respect to $\mathcal{D}$ is equivalent to the answer set of the adorned query with respect to the rewritten database. Hence, if $Q \equiv p(\vec{c})$, then*

$$p(\vec{c})\sigma \in \texttt{ans}(\mathcal{M}_\mathcal{D}, Q) \iff p_{ad}(\vec{c})\sigma \in \texttt{ans}(\mathcal{M}_{\mathcal{D}^m}, Q^a)$$

where σ is a ground substitution for the variables in Q and $Q^a \equiv p_{ad}(\vec{c})$ is the adorned query.

Proof: See [KSS95, Ram91]. □

In [KSS95] it has been shown that the Magic Sets transformation is sound and complete with respect to the answer set for stratifiable databases. However, the resulting rule set may be no more stratifiable and more general approaches than iterated fixpoint computation are needed. For determining the well-founded model [vGRS91] of general logic programs, the alternating fixpoint computation by Van Gelder [vG89, vG93] or the conditional fixpoint by Bry [Bry89] could be used. The application of these methods, however, is not really efficient as they may compute many irrelevant facts during the course of a fixpoint computation. This is caused by the fact that these methods do not take the specific reason for the unstratifiability of the transformed rule sets into account.

Therefore, other methods have been proposed in order to compute the semantics of unstratifiable databases resulting from a Magic Sets transformation explicitly. The structured bottom-up method proposed by Balbin et al. in [BMR88, BPRM91] realizes a bottom-up materialization process for the rewritten database which is suspended each time a negative literal $\neg A$ is queried with respect to a set of particular bindings. Then the query $? - A$ is evaluated by invoking an appropriate function call which actually performs an intermediate magic sets process initiated by corresponding magic seeds derived from the given bindings. Note that this function has to be recursive as the evaluation of the query $? - A$ itself may depend on the evaluation of other negative literals in deeper layers. Afterwards, the global process is continued and the answers for $? - A$ are used to evaluate the negative literal $\neg A$. The structured bottom-up method is complete and sound but because of its complexity difficult to implement. In addition, an implementation of this fixpoint approach may also be inefficient, as it poses problems to the subsequent algebraic optimization phase in 'real' database systems because of the entwined evaluation process.

Another fixpoint-based method for evaluating magic rules is the weak stratification approach by Kerisit and Pugin in [KP88] which is part of the Alexander

method for query evaluation [RLK86]. The Alexander method (like Magic Sets) is a transformation-based approach to query evaluation as well and basically coincides with the Generalized Supplementary Magic Sets approach in [BR91]. Weak stratification as part of the Alexander method has been proposed for evaluating unstratifiable rules which resulted from the rewriting of an originally stratifiable rule set. In [KP88] the authors additionally claim that this method is also applicable to Magic Sets transformed rules because of the similarities between these rewriting techniques. Because of the efficiency and simplicity of the weak stratification approach we will concentrate on this method in the sequel.

4.2 Evaluating Magic Sets Transformed Rules

In this section, the weak stratification method for evaluating Magic Sets transformed rules proposed by Kerisit and Pugin in [KP88] is discussed in more detail. This approach uses the weak consequence operator and a more general stratification concept, the so-called weak stratification, in order to evaluate negative literals correctly. It is shown, however, that the weak stratification approach may lead to a set of answers which is neither sound nor complete with respect to the well-founded model of magic rules. Therefore, we introduce the new concept soft stratification which combined with the soft consequence operator from Section 3.2.2 provides a sound and complete evaluation method for determining the well-founded semantics of a Magic Sets transformed database.

In Section 4.2.1 the concepts weak stratification and weak consequence operator are recalled. Afterwards, we show in Section 4.2.2 by means of a counter example the erroneous derivations of this method and introduce the soft stratification approach instead. After proving the correctness of this approach, we present a comparison to other methods in Section 4.2.3 showing the efficiency of our approach.

4.2.1 The Weak Stratification Approach

The definition of stratification requires two conditions with respect to positive and negative dependencies between predicates to be satisfied. The concept of weak stratification [KP88] relaxes these conditions by considering negative dependencies between predicates only.

Definition 4.6 (Weak Stratification) *Let $\mathcal{D}$ be a deductive database. A weak stratification λ^ω on $\mathcal{D}$ is a mapping from the set of all predicate symbols $Rel_\mathcal{D}$ in $\mathcal{D}$ to the set of positive integers $\mathbb{N}$ such that for all predicate symbols $p, q \in Rel_D$:*

$$p \text{ depends negatively on } q \quad \Longrightarrow \quad \lambda^\omega(p) > \lambda^\omega(q)$$

A weak stratification induces a weak partition $\mathcal{P} = Pos \cup N_1 \cup \ldots \cup N_n$ *of* $\mathcal{R}$ *such that the following holds:*

1. *If* $A \leftarrow W \subseteq \mathcal{R}$ *is a positive rule (i.e., a rule with no negative body literals), then the rule* $A \leftarrow W$ *is in the set Pos.*
2. *If* $A \leftarrow W \subseteq \mathcal{R}$ *is a negative rule (i.e., a rule with at least one negative body literal) and* $\lambda^\omega(A) = i$*, then the rule* $A \leftarrow W$ *is in the set* N_i*.*

In [KP88] it has been shown that every rule set resulting from the Magic Sets transformation of a stratifiable rule set can be weakly stratified. For materializing weakly stratified databases, the authors propose a modified immediate consequence operator which we call weak consequence operator in the following.

Definition 4.7 (Weak Consequence Operator) *Let* $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ *be a deductive database and* λ^ω *a weak stratification of* $\mathcal{R}$ *inducing the weak partition* $\mathcal{P} = P_0 \cup \ldots \cup P_n$ *of* $\mathcal{R}$ *with* $P_0 = Pos$ *and* $P_i = N_i$ *for* $1 \leq i \leq n$*. The weak consequence operator* $T^\omega_\mathcal{P}$ *is a mapping on sets of ground atoms and is defined for* $\mathcal{I} \subseteq \mathcal{H}_\mathcal{D}$ *as follows:*

$$T^\omega_\mathcal{P}(\mathcal{I}) := \begin{cases} \mathcal{I} & \text{if there is no } j \in \{1, \ldots, n\} \text{ such that } T^\star_{P_j}(\mathcal{I}) \supsetneq \mathcal{I} \\ \\ T^\star_{P_i}(\mathcal{I}) & \text{with } i := \texttt{min}\{j \mid T^\star_{P_j}(\mathcal{I}) \supsetneq \mathcal{I}\}, \text{ otherwise.} \end{cases}$$

In contrast to the soft consequence operator introduced in Section 3.2.2, this operator is defined for weakly stratified rule sets only and distinguishes between positive and negative rules in the sets *Pos* and N_i for $1 \leq i \leq n$, respectively. The general evaluation process, however, coincides with the evaluation induced by the soft consequence operator.

As the weak consequence operator is monotonic, its least fixpoint exists and is given by lfp $(T^\omega_\mathcal{P}, \mathcal{F})$. It is obvious that the application of $T^\omega_\mathcal{P}$ can lead to more positive conclusions than there are within the set of positive conclusions of the corresponding well-founded model. In [KP88] the authors claim, however, that at least the answer relation with respect to a given query is correctly determined by means of lfp $(T^\omega_\mathcal{P}, \mathcal{F})$. We will show in the following section that this is not always true and present a refined version of the concept weak stratification in order to determine the complete well-founded model correctly using the soft consequence operator instead.

4.2.2 The Soft Stratification Approach

In general it is possible to find several distinct weak stratifications for a given rule set. However, not every chosen weak stratification may lead to correct derivations of facts with respect to the well-founded model if the weak consequence operator

is applied. For illustrating this problem consider the following example of a stratifiable deductive database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$

$$\underline{\mathcal{R}}: \quad \mathtt{p(X)} \leftarrow \mathtt{b(X,Y,Z)} \wedge \neg \mathtt{q(X)} \wedge \neg \mathtt{q(Y)} \wedge \neg \mathtt{q(Z)}$$
$$\mathtt{q(X)} \leftarrow \mathtt{d(X)}$$

$$\underline{\mathcal{F}:} \quad \mathtt{b(1,2,3)} \;\; \mathtt{d(2)} \;\; \mathtt{d(3)}$$

and the query $Q \equiv p(1)$. A weak partition $\mathcal{P} = Pos \cup N_1$ of the Magic Sets transformed rule set $\mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{rule\_seed}(Q)\}$ could be as follows:

$\underline{Pos}$:

$$\mathtt{q_b(X)} \leftarrow \mathtt{m\_q_b(X)} \wedge \mathtt{d(X)}$$
$$\mathtt{m\_p_b(X)} \leftarrow \mathtt{m\_s\_p_b(X)}$$
$$\mathtt{m\_q_b(X)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)}$$

$\underline{N_1}$:

$$\mathtt{p_b(X)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg \mathtt{q_b(X)} \wedge \neg \mathtt{q_b(Y)} \wedge \neg \mathtt{q_b(Z)}$$
$$\mathtt{m\_q_b(Y)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg \mathtt{q_b(X)}$$
$$\mathtt{m\_q_b(Z)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg \mathtt{q_b(X)} \wedge \neg \mathtt{q_b(Y)}$$

The magic seed is given by $m\_s\_p_b(1)$. Evaluating these rules using $T^\omega_\mathcal{P}$ would yield $\{m\_p_b(1)\}$, $\{m\_q_b(1)\}$, $\{p_b(1),\ m\_q_b(2),\ m\_q_b(3)\}$ and $\{q_b(2), q_b(3)\}$. With respect to the corresponding well-founded model $\mathcal{M}_{\mathcal{D}^m} := \{m\_p_b(1),\ m\_q_b(1),\ m\_q_b(2),\ q(2)\} \cup \mathcal{F} \cup \{m\_s\_p_b(1)\}$ of $\mathcal{D}^m$, the facts $m\_q_b(3)$ and $q_b(3)$ are erroneous derivations. Additionally, the incorrect answer fact $p_b(1)$ is derived which is clearly wrong as no p-fact is included in the iterated fixpoint of the original database. The erroneous derivations are due to the fact that only negative dependencies are considered in weak partitions but no positive ones. It is necessary, however, to consider also those positive dependencies which ensure that all necessary derivations of query and answer facts have been made before a rule with a corresponding negative literal is evaluated.

A possible solution to this problem is to choose a weak partition in such a way that all rules on which a negative literal positively or negatively depends lie in deeper layers. Consider, for instance, the negative literal $\neg q_b(Y)$ in the rule for defining relation p. This literal also appears in the rule for defining $m\_q_b(Z)$ which ought to be applied after the rules

$$Pos \cup \{\mathtt{m\_q_b(Y)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg \mathtt{q_b(X)}\}$$

have been considered in deeper layers by $T^\omega_\mathcal{P}$ in order to provide all necessary answer and sub-query facts. Additionally, for evaluating the negative literal $\neg q_b(Z)$ in the rule defining p, the rule

$$\mathtt{m\_q_b(Z)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg\mathtt{q_b(X)} \wedge \neg\mathtt{q_b(Y)}$$

must have been considered already in deeper layers. The following definition formalizes the dependency between literals and rules in a Magic Sets transformed rule set.

Definition 4.8 (Required Rules) *Let $\mathcal{R}$ be a set of stratifiable deductive rules, Q an adorned query, $\mathcal{R}^Q$ the adorned rule set of $\mathcal{R}$ with respect to Q and $\mathtt{ms}(\mathcal{R}^Q)$ the corresponding magic set transformed rules. For each rule $R_i \equiv A \leftarrow \mathtt{magic}(B), L_{i,1}, \ldots L_{i,l_i} \in \mathtt{ms}(\mathcal{R}^Q)$ $(i = 1, ..., |\mathtt{ms}(\mathcal{R}^Q)|)$ and each derived body literal $L_{i,j}$ $(j \in \{1, \ldots, l_i\})$ the set of required rules $req(L_{i,j})$ is defined as the smallest set satisfying the following conditions:*

1. *For each derived body literal $L_{i,k}$ $(1 \leq k \leq j)$ a sub-query rule of the form*

   $$\mathtt{magic}(L_{i,k}) \leftarrow \mathtt{magic}(B) \wedge L_{i,1} \wedge \ldots L_{i,k-1}$$

   is in $req(L_{i,j})$.

2. *For each derived body literal $L_{i,k}$ $(1 \leq k \leq j)$ the magic transformed rules $\mathtt{ms}(\mathtt{def}_{\mathcal{R}^Q}(\mathtt{pred}(L_{i,k}))$ are in $req(L_{i,j})$.*

As an example consider again the deductive database discussed above and its adorned rule set $\mathcal{R}^Q$

$$\mathtt{p_b(X)} \leftarrow \mathtt{b(X,Y,Z)} \wedge \neg\mathtt{q_b(X)} \wedge \neg\mathtt{q_b(Y)} \wedge \neg\mathtt{q_b(Z)}$$
$$\mathtt{q_b(X)} \leftarrow \mathtt{d(X)}$$

with respect to the query $Q \equiv p(1)$. Suppose the resulting magic sub-query rule

$$\mathtt{m\_q_b(Z)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg\mathtt{q_b(X)} \wedge \neg\mathtt{q_b(Y)}$$

for defining $m\_q_b(Z)$ has been numbered R_4. The set of required rules for the body literal $L_{4,3} \equiv \neg q_b(Y)$ within this rule then is given by

$$\begin{aligned} req(L_{4,3}) := \{\ & \mathtt{m\_q_b(X)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)}, \\ & \mathtt{m\_q_b(Y)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg\mathtt{q_b(X)}, \\ & \mathtt{q_b(X)} \leftarrow \mathtt{m\_q_b(X)} \wedge \mathtt{d(X)}\ \} \end{aligned}$$

while the latter rule results from the magic set transformation $\mathtt{ms}(\mathtt{def}_{\mathcal{R}^Q}(\mathtt{pred}(L_{4,2}))$ of literal $L_{4,2} \equiv \neg q_b(X)$ and from the magic set transformation $\mathtt{ms}(\mathtt{def}_{\mathcal{R}^Q}(\mathtt{pred}(L_{4,3}))$, respectively.

We will now introduce the notion soft stratification to denote a weak stratification which also takes the sets of required rules for negative literals into account.

Definition 4.9 (Soft Stratification) *Let $\mathcal{R}$ be a stratifiable deductive rule set and $\mathtt{ms}(\mathcal{R}^Q)$ the corresponding set of Magic Set transformed rules with respect to a given query Q. A soft stratification λ^s on $\mathtt{ms}(\mathcal{R}^Q)$ is a mapping from the set of rules $\mathtt{ms}(\mathcal{R}^Q)$ to the set of positive integers $\mathbb{N}$, such that for all negative rules $R_{neg} \in \mathtt{ms}(\mathcal{R}^Q)$ and all negative literals L of R_{neg}*

$$R' \in \mathtt{ms}(\mathcal{R}^Q) \text{ and } R' \in req(L) \implies \lambda^s(R_{neg}) > \lambda^s(R').$$

In a soft stratification, positive as well as negative dependencies are considered, leading to a stronger condition in comparison to weak stratification on the subsequent rule partitioning. Stratification problems introduced by the Magic Sets transformation, however, are avoided because dependencies between rules and not between predicates (as in the original condition of stratification) are considered. However, only necessary dependencies between rules are considered in order to be most flexible in the relational reoptimization phase. For materializing softly stratified databases, we may now use the soft consequence operator and apply it to a partition of a Magic Sets transformed rule set which satisfy the condition of soft stratification.

As an example consider the following partition $\mathcal{P} = P_1 \cup P_2 \cup P_3 \cup P_4$ of the Magic Sets transformed rule set $\mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{rule\_seed}(Q)\}$

$\underline{P_1}$:

$$\mathtt{m\_p_b(X)} \leftarrow \mathtt{m\_s\_p_b(X)}$$
$$\mathtt{m\_q_b(X)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{d(X)}$$
$$\mathtt{q_b(X)} \leftarrow \mathtt{m\_q_b(X)} \wedge \mathtt{d(X)}$$

$\underline{P_2}$:

$$\mathtt{m\_q_b(Y)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg\mathtt{q_b(X)}$$

$\underline{P_3}$:

$$\mathtt{m\_q_b(Z)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg\mathtt{q_b(X)} \wedge \neg\mathtt{q_b(Y)}$$

$\underline{P_4}$:

$$\mathtt{p_b(X)} \leftarrow \mathtt{m\_p_b(X)} \wedge \mathtt{b(X,Y,Z)} \wedge \neg\mathtt{q_b(X)} \wedge \neg\mathtt{q_b(Y)} \wedge \neg\mathtt{q_b(Z)}$$

which satisfies the partial ordering induced by a soft stratification. The determination of lfp $(T^s_{\mathcal{P}}, \mathcal{F} \cup \{m\_s\_p_b(1)\})$ with $F = \{b(1,2,3), d(2), d(3)\}$ using the given soft partition $\mathcal{P}$ then correctly yields the well-founded model $\mathcal{M}_{\mathcal{D}^m} := \{m\_p_b(1), m\_q_b(1)\} \cup \{m\_q_b(2)\} \cup \{q(2)\} \cup \mathcal{F} \cup \{m\_s\_p_b(1)\}$ of $\mathcal{D}^m$.

The following proposition shows that if the Magic Sets transformation is applied to an originally stratifiable rule set, the rewritten rules will always be softly stratifiable.

Proposition 4.10 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable deductive database and $\mathtt{ms}(\mathcal{R}^Q)$ the corresponding set of Magic Sets transformed rules with respect to a given query Q. Then a soft stratification of $\mathtt{ms}(\mathcal{R}^Q)$ exists.*

Proof: Suppose there is no soft stratification of the magic set transformed rules $\mathtt{ms}(\mathcal{R}^Q)$. Then there must be two rules $R_1 \in \mathtt{ms}(\mathcal{R}^Q)$ with a negative body literal L_1 and $R_2 \in \mathtt{ms}(\mathcal{R}^Q)$ with a negative body literal L_2 such that $R_2 \in req(L_1)$ and $R_1 \in req(L_2)$. Without loss of generality we can assume R_1 and R_2 to be answer rules because for every sub-query rule $R' \in \mathtt{ms}(\mathcal{R}^Q)$ with a derived body literal L' there exists a corresponding answer rule $R'' \in \mathtt{ms}(\mathcal{R}^Q)$ with a derived body literal L'' such that $req(L'') = req(L')$.
Therefore, R_2 must be in the set $\mathtt{ms}(\mathtt{def}_{\mathcal{R}^Q}(\mathtt{pred}(L_1))$ and its corresponding adorned rule R_2^Q must be in $\mathtt{def}_{\mathcal{R}^Q}(\mathtt{pred}(L_1))$. As L_1 is a negative literal in R_1, R_2^Q then must depend negatively on R_1^Q. Analogously, you can show that the adorned rule R_1^Q must depend negatively on R_2^Q. Thus, the adorned rule set $\mathcal{R}^Q$ must be unstratifiable and subsequently $\mathcal{R}$, which contradicts the prerequisites of the proposition.

□

The following theorem shows the correctness of the soft stratification approach. To this end, we prove that the true portion of the well-founded model of a magic rewritten database $\mathcal{M}^+_{\mathcal{D}^m}$ with $\mathcal{D}^m = \langle \mathcal{F} \cup \{\mathtt{seed}(Q)\}, \mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{seed\_rule}(Q)\} \rangle$ coincides with the least fixpoint of $T^s_{\mathcal{P}}$ with respect to a soft partition $\mathcal{P}$ of the magic rewritten rules which contains the fact base $\mathcal{F} \cup \{\mathtt{seed}(Q)\}$ completely.

Theorem 4.2 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable deductive database, $\mathtt{ms}(\mathcal{R}^Q)$ the corresponding set of Magic Sets transformed rules with respect to a given query Q, and λ^s a soft stratification on $\mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{seed\_rule}(Q)\}$ inducing the soft partition $\mathcal{P} = P_1 \uplus \ldots \uplus P_n$. Evaluating these rules using the soft consequence operator $T^s_{\mathcal{P}}$ yields the well-founded model $\mathcal{M}_{\mathcal{D}^m}$ of $\mathcal{D}^m = \langle \mathcal{F} \cup \{\mathtt{seed}(Q)\}, \mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{seed\_rule}(Q)\} \rangle$. Thus, for $\mathcal{M}_{\mathcal{D}^m} = \mathcal{M}^+_{\mathcal{D}^m} \uplus \neg \cdot \overline{\mathcal{M}^+_{\mathcal{D}^m}}$ the following holds:*

$$\mathtt{lfp}\ (T^s_{\mathcal{P}}, \mathcal{F} \cup \{\mathtt{seed}(Q)\}) = \mathcal{M}^+_{\mathcal{D}^m}.$$

Proof: The theorem is shown by induction on the number l of components in the partition $\mathcal{P}$ induced by λ^s on $\mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{seed\_rule}(Q)\}$. Without loss of generality we can assume that $\mathtt{seed\_rule}(Q) \in P_1$ because no other rule may derive facts unless this rule has been fired first. The application of $T^s_{\mathcal{P}}$ then starts

again with the first component P_1 of partition $\mathcal{P}$.

Suppose that $l = 1$: All negative literals in P_1 have empty sets of required rules as they refer to base relations only. Hence, the true portion of the well-founded model of the semi-positive rule set P_1 for an arbitrary fact base X is given by

$$\begin{aligned}\mathcal{M}^+_{\langle P_1, X\rangle} &= \text{lfp } (T^\star_{P_1}, X)\\ &=_{def} T^\star_{P_1}(T^\star_{P_1}(\ldots T^\star_{P_1}(X)\ldots)\\ &=_{def} \text{lfp } (T^s_{P_1}, X)\end{aligned}$$

This holds in particular for the fact base $X = \mathcal{F} \cup \{\texttt{seed}(Q)\}$.

Suppose that $l > 1$: Assuming

$$\texttt{lfp } (T^s_{P_1 \uplus \ldots \uplus P_{l-1}}, X) = \mathcal{M}^+_{\langle P_1 \uplus \ldots \uplus P_{l-1}, X\rangle}$$

holds for any fact base X, we have to show that

$$\texttt{lfp } (T^s_{P_1 \uplus \ldots \uplus P_l}, X) = \mathcal{M}^+_{\langle P_1 \uplus \ldots \uplus P_l, X\rangle}.$$

According to Definition 3.4 the least fixpoint computation of $T^s_{P_1 \uplus \ldots \uplus P_l}$ with respect to the fact base $\mathcal{F} \cup \{\texttt{seed}(Q)\}$ corresponds to the following sequence of separate fixpoint computations

$$\begin{aligned}&F_1 := T^\star_{P_l}[\text{lfp } (T^s_{P_1 \uplus \ldots \uplus P_{l-1}}, \mathcal{F} \cup \{\texttt{seed}(Q)\})]\\ &F_2 := T^\star_{P_l}[\text{lfp } (T^s_{P_1 \uplus \ldots \uplus P_{l-1}}, F_1\})]\\ &\ldots\\ &F_m := T^\star_{P_l}[\text{lfp } (T^s_{P_1 \uplus \ldots \uplus P_{l-1}}, F_{m-1}\})],\end{aligned}$$

performed until no more new facts can be derived; that is $F_m = F_{m+1}$. Using the induction hypothesis, the fixpoint computations of partition $P_1 \uplus \ldots \uplus P_{l-1}$ with respect to the different base facts F_i $(1 \leq i \leq m)$ are correct and therefore coincide with the corresponding well-founded models. Thus, we have

$$\begin{aligned}&F_1 := T^\star_{P_l}(\mathcal{M}^+_{\langle P_1 \uplus \ldots \uplus P_{l-1}, \mathcal{F} \cup \{\texttt{seed}(Q)\}\rangle})\\ &F_2 := T^\star_{P_l}(\mathcal{M}^+_{\langle P_1 \uplus \ldots \uplus P_{l-1}, F_1\rangle})\\ &\ldots\\ &F_m := T^\star_{P_l}(\mathcal{M}^+_{\langle P_1 \uplus \ldots \uplus P_{l-1}, F_{m-1}\rangle}).\end{aligned}$$

Let us suppose that $\texttt{lfp } (T^s_{P_1 \uplus \ldots \uplus P_l}, \mathcal{F} \cup \{\texttt{seed}(Q)\}) \subseteq \mathcal{M}^+_{\langle P_1 \uplus \ldots \uplus P_l, \mathcal{F} \cup \{\texttt{seed}(Q)\}\rangle}$ does not hold. Then there must be a fact $f \equiv p(\vec{c})$ and a set of base facts F_j with $j \in \{1, \ldots, m\}$ such that $f \in F_j$ and $f \notin \mathcal{M}^+_{\langle P_1 \uplus \ldots \uplus P_l, \mathcal{F} \cup \{\texttt{seed}(Q)\}\rangle}$. As the computations of the well-founded models with respect to partition $P_1 \uplus \ldots \uplus P_{l-1}$ are correct, there must be a rule $R \in P_l$ with $\texttt{pred}(f) = \texttt{pred}(R)$ such that the application of R leads to the erroneous derivation of f. On the one hand, any negative

literal in the body of R is evaluated correctly because its corresponding req-set is complete (see [KSS95, Ram91]) and consists of rules only located in components $P_1 \ldots P_{l-1}$ (because of the soft stratification property) whose corresponding well-founded model is determined correctly according to the induction hypothesis. On the other hand, any positive literal is also evaluated correctly, as there is only one application of $T^{\star}_{P_l}$ and therefore every substitution must have come from the previously determined true portion of the well-founded model $\mathcal{M}^{+}_{\langle P_1 \cup \ldots \cup P_{l-1}, F_{j-1}\rangle}$. But then it can be concluded that the erroneous derivation f may only be due to an erroneous fact base F_{j-1}. Analogously, it can be followed that F_1 must have been an erroneous fact base and because of the correct application of $T^{\star}_{P_l}$ the true portion of the well-founded model $\mathcal{M}^{+}_{\langle P_1 \cup \ldots \cup P_{l-1}, \mathcal{F} \cup \{\mathtt{seed}(Q)\}\rangle}$ must have been incorrect which is a contradiction to the induction hypothesis.

Let us suppose that $\mathtt{lfp}\ (T^{s}_{P_1 \cup \ldots \cup P_l}, \mathcal{F} \cup \{\mathtt{seed}(Q)\}) \supseteq \mathcal{M}^{+}_{\langle P_1 \cup \ldots \cup P_l, \mathcal{F} \cup \{\mathtt{seed}(Q)\}\rangle}$ does not hold. Then there must be a fact $f \equiv p(\vec{c})$ such that $f \notin F_m$ and $f \in \mathcal{M}^{+}_{\langle P_1 \cup \ldots \cup P_l, \mathcal{F} \cup \{\mathtt{seed}(Q)\}\rangle}$. As the computations of the well-founded models with respect to partition $P_1 \cup \ldots \cup P_{l-1}$ are correct, there must be a rule $R \in P_l$ with $\mathtt{pred}(f) = \mathtt{pred}(R)$ such that the final application of $T^{\star}_{P_l}$ with respect to $\mathcal{M}^{+}_{\langle P_1 \cup \ldots \cup P_{l-1}, F_{m-1}\rangle}$ does not derive f (which must be erroneous because of $F_m = F_{m+1}$). Analogously to the previous case, we can assume that all positive as well as all negative literals within the body of R are correctly evaluated over $\mathcal{M}^{+}_{\langle P_1 \cup \ldots \cup P_{l-1}, F_{m-1}\rangle}$. Therefore, the previously determined fact base F_{m-1} cannot be correct. Analogously it can be followed again that the computation of F_1 must have been erroneous and subsequently the true portion of the well-founded model $\mathcal{M}^{+}_{\langle P_1 \cup \ldots \cup P_{l-1}, \mathcal{F} \cup \{\mathtt{seed}(Q)\}\rangle}$ must have been incorrect. This again contradicts our induction hypothesis.

Thus, we conclude $\mathtt{lfp}\ (T^{s}_{P_1 \cup \ldots \cup P_l}, \mathcal{F} \cup \{\mathtt{seed}(Q)\}) = \mathcal{M}^{+}_{\langle P_1 \cup \ldots \cup P_l, \mathcal{F} \cup \{\mathtt{seed}(Q)\}\rangle}$.

□

From Proposition 4.10 and Theorem 4.2 follows that the concepts soft stratification and soft consequence operator together provide a sound and complete evaluation method for determining the well-founded semantics of a Magic Sets transformed database. This simple approach is easy to implement on top of an existing relational database system and allows for further (algebraic) optimizations in contrast to the structured query evaluation method by Balbin et al., for example. In the following section we argue that the soft stratification approach represents indeed a more efficient query evaluation method in comparison to the alternative bottom-up approaches mentioned above.

4.2.3 Comparison to Other Approaches

In this section we compare the soft stratification method to other fixpoint-based query evaluation methods by means of an example. When considering the quality of query evaluation methods in this context, we usually take the number of derived facts as well as the number of iteration rounds as cost measurements into account. We refrain from presenting a formal complexity analysis as all discussed methods require costs polynomial in the size of the corresponding Herbrand universe. Nevertheless, the soft stratification approach always performs asymptotically better.

For illustrating this, let us consider the following example of a stratifiable database $\mathcal{D} = \langle \mathcal{R}, \mathcal{F} \rangle$ with

$\underline{\mathcal{R}:}$

$$\begin{array}{l} \mathtt{i(X)} \leftarrow \neg \mathtt{s(X)} \wedge \mathtt{j(X,Y)} \wedge \mathtt{i(Y)} \\ \mathtt{i(X)} \leftarrow \mathtt{k(X)} \\ \mathtt{s(X)} \leftarrow \mathtt{b(X,Y)} \wedge \mathtt{s(Y)} \\ \mathtt{s(X)} \leftarrow \mathtt{g(X)} \end{array}$$

$\underline{\mathcal{F}:}$

$$\begin{array}{l} \mathtt{k(8)},\ \mathtt{k(9)} \\ \mathtt{j(6,4)},\ \mathtt{j(7,4)},\ \mathtt{j(4,8)} \\ \mathtt{g(3)},\ \mathtt{g(5)} \\ \mathtt{b(1,2)},\ \mathtt{b(2,3)},\ \mathtt{b(4,5)}. \end{array}$$

and the query $Q \equiv i(6)$. After applying Magic Sets, the transformed rules are

$$\begin{array}{l} \mathtt{i_b(X)} \leftarrow \mathtt{m\_i_b(X)} \wedge \neg \mathtt{s_b(X)} \wedge \mathtt{j(X,Y)} \wedge \mathtt{i_b(Y)} \\ \mathtt{i_b(X)} \leftarrow \mathtt{m\_i_b(X)} \wedge \mathtt{k(X)} \\ \mathtt{s_b(X)} \leftarrow \mathtt{m\_s_b(X)} \wedge \mathtt{b(X,Y)} \wedge \mathtt{s_b(Y)} \\ \mathtt{s_b(X)} \leftarrow \mathtt{m\_s_b(X)} \wedge \mathtt{g(X)} \\ \\ \mathtt{m\_i_b(X)} \leftarrow \mathtt{m\_s\_i_b(X)} \\ \mathtt{m\_i_b(Y)} \leftarrow \mathtt{m\_i_b(X)} \wedge \neg \mathtt{s_b(X)} \wedge \mathtt{j(X,Y)} \\ \mathtt{m\_s_b(X)} \leftarrow \mathtt{m\_i_b(X)} \\ \mathtt{m\_s_b(Y)} \leftarrow \mathtt{m\_s_b(X)} \wedge \mathtt{b(X,Y)}. \end{array}$$

The following negative cycle can be found in the corresponding dependency graph:

$$s_b \xrightarrow{neg} m\_i_b \xrightarrow{pos} m\_s_b \xrightarrow{pos} s_b$$

For computing the well-founded model of the Magic Set transformed database $\mathcal{D}^m := \langle \mathcal{F} \cup \{\mathtt{seed}(Q)\}, \mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{seed\_rule}(Q)\} \rangle$ we compare the alternating fixpoint computation by Van Gelder [vG89] and the structured bottom-up method by Balbin et al. [BPRM91] with our approach. First let us trace the alternating fixpoint computation using Algorithm 2 from Section 3.3.2. We begin with initializing the set of definitely true facts DT^0 with the empty set $DT^0 = \emptyset$. Afterwards the largest overestimation of positive conclusions is computed with respect to the empty set of true positive conclusions DT^0 implying that all negative literals are assumed to be true. The least fixpoint of $\widehat{T}_{\mathcal{R}}$ then yields the first set of not definitely false facts

$$\begin{aligned} NDF^1 := \mathcal{F} &\cup \{\mathtt{seed}(Q)\} \\ &\cup \{\mathtt{m\_i_b}(4),\ \mathtt{m\_i_b}(6),\ \mathtt{m\_i_b}(8)\} \\ &\cup \{\mathtt{i_b}(4),\ \mathtt{i_b}(6),\ \mathtt{i_b}(8)\} \\ &\cup \{\mathtt{m\_s_b}(4),\ \mathtt{m\_s_b}(5),\ \mathtt{m\_s_b}(6),\ \mathtt{m\_s_b}(8)\} \\ &\cup \{\mathtt{s_b}(4), \mathtt{s_b}(5)\}. \end{aligned}$$

The first set of true negative conclusions DF^1 is implicitly given by complementing NDF^1, e.g.:

$$DF^1 := \neg \cdot (\mathcal{H}_{\mathcal{D}^m} \setminus NDF^1)$$

The subsequent computation of definitely true facts DT^1 is performed by employing the previously obtained set of not definitely false facts NDF^1 for evaluating negative literals using the negation as failure principle

$$\begin{aligned} DT^1 := \mathcal{F} &\cup \{\mathtt{seed}(Q)\} \\ &\cup \{\mathtt{m\_i_b}(4),\ \mathtt{m\_i_b}(6)\} \\ &\cup \{\mathtt{m\_s_b}(4),\ \mathtt{m\_s_b}(5),\ \mathtt{m\_s_b}(6)\} \\ &\cup \{\mathtt{s_b}(4), \mathtt{s_b}(5)\}. \end{aligned}$$

As the sets DT^0 and DT^1 are not equal, the iteration continues producing the following sets of positive conclusions

$$\begin{aligned} NDF^2 &:= DT^1 \\ DT^2 &:= DT^1. \end{aligned}$$

As no more true positive conclusions can be derived, a fixpoint has been reached. The alternating fixpoint $\mathcal{M}^+_{\mathcal{D}^m} = \mathcal{F} \cup \{\mathtt{seed}(Q)\} \cup \{m\_i_b(4),\ m\_i_b(6),\ m\_s_b(4),\ m\_s_b(5), m\_s_b(6),\ s_b(4), s_b(5)\}$ coincides with the positive portion of the corresponding total well-founded model $\mathcal{M}_{\mathcal{D}^m}$ such that $\mathcal{M}_{\mathcal{D}^m} = \mathcal{M}^+_{\mathcal{D}^m} \cup \neg \cdot \overline{\mathcal{M}^+_{\mathcal{D}^m}}$.

Let us compare this result with the application of the soft consequence operator. The set of required rules for the body literal $\neg s_b(X)$ within the answer rule for defining i_b is

$$\begin{aligned} req(\neg s_b(X)) := \{\ &\mathtt{m\_s_b(X)} \leftarrow \mathtt{m\_i_b(X)}, \\ &\mathtt{s_b(X)} \leftarrow \mathtt{m\_s_b(X)} \wedge \mathtt{b(X,Y)} \wedge \mathtt{s_b(Y)}, \\ &\mathtt{s_b(X)} \leftarrow \mathtt{m\_s_b(X)} \wedge \mathtt{g(X)}, \\ &\mathtt{m\_s_b(Y)} \leftarrow \mathtt{m\_s_b(X)} \wedge \mathtt{b(X,Y)}\ \} \end{aligned}$$

which coincides with the required rule set of the corresponding negative body literal within the sub-query rule for defining m_i_b. The following partition $\mathcal{P} = P_1 \cup P_2 \cup P_3$ of the Magic Sets transformed rule set $\mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{rule\_seed}(Q)\}$ satisfies the condition of soft stratification:

$\underline{P_1}$:

$$\begin{aligned}
&\mathtt{m_s_b(X)} \leftarrow \mathtt{m_i_b(X)}\\
&\mathtt{s_b(X)} \leftarrow \mathtt{m_s_b(X)} \wedge \mathtt{b(X,Y)} \wedge \mathtt{s_b(Y)}\\
&\mathtt{s_b(X)} \leftarrow \mathtt{m_s_b(X)} \wedge \mathtt{g(X)}\\
&\mathtt{m_s_b(Y)} \leftarrow \mathtt{m_s_b(X)} \wedge \mathtt{b(X,Y)}
\end{aligned}$$

$\underline{P_2}$:

$$\begin{aligned}
&\mathtt{i_b(X)} \leftarrow \mathtt{m_i_b(X)} \wedge \neg\mathtt{s_b(X)} \wedge \mathtt{j(X,Y)} \wedge \mathtt{i_b(Y)}\\
&\mathtt{m_i_b(Y)} \leftarrow \mathtt{m_i_b(X)} \wedge \neg\mathtt{s_b(X)} \wedge \mathtt{j(X,Y)}
\end{aligned}$$

$\underline{P_3}$:

$$\begin{aligned}
&\mathtt{m_i_b(X)} \leftarrow \mathtt{m_s_i_b(X)}\\
&\mathtt{i_b(X)} \leftarrow \mathtt{m_i_b(X)} \wedge \mathtt{k(X)}
\end{aligned}$$

The computation of lfp $(T^s_{\mathcal{P}}, \mathcal{F})$ induces the following sequence of sets:

$$\begin{aligned}
F_1 &:= \mathcal{F} \cup \{\mathtt{seed}(Q)\}\\
F_2 &:= T^{\star}_{P_3}(F_1) = F_1 \cup \{\mathtt{m_i_b(6)}\}\\
F_3 &:= T^{\star}_{P_1}(F_2) = F_2 \cup \{\mathtt{m_s_b(6)}\}\\
F_4 &:= T^{\star}_{P_2}(F_3) = F_3 \cup \{\mathtt{m_i_b(4)}\}\\
F_5 &:= T^{\star}_{P_1}(F_4) = F_4 \cup \{\mathtt{m_s_b(4)}\}\\
F_6 &:= T^{\star}_{P_1}(F_5) = F_5 \cup \{\mathtt{m_s_b(5)}\}\\
F_7 &:= T^{\star}_{P_1}(F_6) = F_6 \cup \{\mathtt{s_b(5)}\}\\
F_8 &:= T^{\star}_{P_1}(F_7) = F_7 \cup \{\mathtt{s_b(4)}\}\\
F_9 &= F_8
\end{aligned}$$

This result coincides with the alternating fixpoint determined above. However, as this computation is strictly monotonic, any overestimations are avoided. That is, in contrast to the alternating fixpoint computation, the facts $m\_i_b(8), m\_s_b(8)$, $i_b(4)$, $i_b(6)$, $i_b(8)$ are not derived. In addition, it is possible to apply a semi-naive evaluation method in order to avoid the recomputation of certain facts. A possible drawback of our approach could be the expensive search for the next partition set to be applied which might require testing all 'lower' partitions. This can be partly avoided by providing additional information on literal dependencies in order to avoid the consideration of partition sets which cannot be affected by newly derived facts.

The structured bottom-up method by Balbin et al. [BPRM91] uses a function $eval/2$ for evaluating the Magic Set transformed rule set. Every time a negative literal is considered, the function $eval/2$ is recursively called for performing a

local fixpoint computation over the relevant portion of the Magic Set transformed rules. The nested fixpoint computations terminate as soon as no more facts can be added to the global database state M_e. In our example, the evaluation process starts with $M_e := \mathcal{F} \cup \{m\_i_b(6)\}$ and the function call $eval(i_b, M_e)$. The overall evaluation process then looks as follows:

$eval(i_b, \mathcal{F} \cup \{\mathtt{m\_i_b}(6)\})$
$M_e := M_e \cup \mathcal{F} \cup \{\mathtt{m\_i_b}(6)\}$

$eval(s_b, \{\mathtt{m\_s_b}(6)\})$
$M_e := M_e \cup \{\mathtt{m\_s_b}(6)\}$
...
$M_e := M_e \cup \emptyset$
$eval(s_b, \{\mathtt{m\_s_b}(6)\})$
$M_e := M_e \cup \{\mathtt{m\_s_b}(6)\}$
...
$M_e := M_e \cup \emptyset$
$M_e := M_e \cup \{\mathtt{m\_i_b}(4)\}$

} I

$eval(s_b, \{\mathtt{m\_s_b}(4)\})$
$M_e := M_e \cup \{\mathtt{m\_s_b}(4)\}$
...
$M_e := M_e \cup \{\mathtt{m\_s_b}(5),\ \mathtt{s_b}(4),\ \mathtt{s_b}(5)\}$
$eval(s_b, \{\mathtt{m\_s_b}(4)\})$
$M_e := M_e \cup \{\mathtt{m\_s_b}(4)\}$
...
$M_e := M_e \cup \{\mathtt{m\_s_b}(5),\ \mathtt{s_b}(4),\ \mathtt{s_b}(5)\}$
$M_e := M_e \cup \emptyset$

} II

While evaluating the top level function call, two iteration rounds `I` and `II` can be identified, each performing a separate fixpoint computation for the two 'negative' queries against s_b. Note that the evaluation basically coincides with the one induced by the soft stratification approach. This means that for each negative derived literal the corresponding separate 'negative' query evaluation is performed in the same order as in the soft stratification approach. Nevertheless, as each negative (derived) literal causes a separate function call, many facts are repeatedly computed in the example above. In addition, this separation of context makes it difficult or even impossible to apply further rule optimization techniques as proposed, e.g., in [NRSU95, Ros91, RS91, Sud92, LMSS95, LS92, LS95, SMK97, GSSS91, SSS90, Beh00]. Therefore, the soft stratification approach performs at least as good as the structured query evaluation method, but because of the deficiencies mentioned above asymptotically better in the general case.

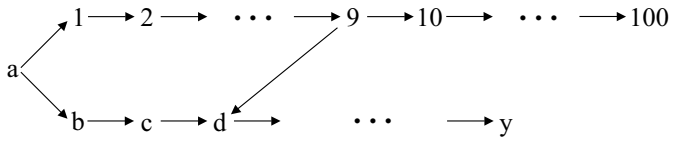


Figure 4.1: Connections represented by relation `edge/2`.

Of course, for finite Herbrand universes and fixed rule sets any of the above mentioned approaches requires time polynomial in the size of the Herbrand universe. The actual efficiency therefore strongly depends on the chosen implementation and the applied optimization techniques. The discussion above, however, already indicates some principle problems of the alternative approaches in comparison to our proposed method.

We will now turn our attention to an orthogonal optimization problem by means of existential query optimization. This problem is particularly interesting in the context of softly stratifiable rules as every negative derived literal leads to a set of existential queries.

4.3 Existential Query Optimization

Interestingly, the problem of optimizing the evaluation of existential queries has drawn little attention in the deductive database literature up till now, e.g. [RBK88, NRSU89, Aze97, Beh00], and there exists no general solution yet. In this section, however, we will present a transformation-based approach which solves this problem at least for a certain class of existential queries. Basically, an existential query is a query which contains no free variables at the time of its evaluation, i.e., all variables occurring in the query literal are bound to certain constants or the query literal is a 0-ary predicate. For answering an existential query in a set-oriented language like Datalog it is sufficient to find just one appropriate answer fact while all other possible derivations of the same answer fact are not needed and ought to be avoided. As an example, let us consider the following positive database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ with $\mathcal{R}$ consisting of the well-known transitive closure rules for defining the derived relation `path/2`

$\underline{\mathcal{R}}$:

$$\mathtt{p(X,Y)} \leftarrow \mathtt{e(X,Y)}$$
$$\mathtt{p(X,Y)} \leftarrow \mathtt{e(X,Z)} \wedge \mathtt{p(Z,Y)}$$

$\underline{\mathcal{F}}$:

```
e(a,1), e(1,2), e(2,3), ...e(99,100), e(9,d)
e(a,b), e(b,c), e(c,d), ..., e(x,y)
```

and the query $Q \equiv p(a, d)$ asking for a path between the nodes a and d. A graphical illustration of the facts $\mathcal{F}$ is presented in Figure 4.1. Apparently there are two different derivation paths for the fact $\mathtt{p}(\mathtt{a}, \mathtt{d})$ such that the query can be successfully answered. The Magic DB transformation yields the deductive rule set $\mathcal{R}^m := \mathtt{ms}(\mathcal{R}^Q) \cup \{\mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}) \leftarrow \mathtt{m\_s\_p_{bb}}(\mathtt{X}, \mathtt{Y})\}$ with

$\underline{\mathtt{ms}(\mathcal{R}^Q)}$:

$$
\begin{aligned}
\mathtt{p_{bb}}(\mathtt{X}, \mathtt{Y}) &\leftarrow \mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}) \wedge \mathtt{e}(\mathtt{X}, \mathtt{Y}) \\
\mathtt{p_{bb}}(\mathtt{X}, \mathtt{Y}) &\leftarrow \mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}) \wedge \mathtt{e}(\mathtt{X}, \mathtt{Z}) \wedge \mathtt{p_{bb}}(\mathtt{Z}, \mathtt{Y}) \\
\mathtt{m\_p_{bb}}(\mathtt{Z}, \mathtt{Y}) &\leftarrow \mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}) \wedge \mathtt{e}(\mathtt{X}, \mathtt{Z})
\end{aligned}
$$

as well as the magic seed fact $\mathtt{m\_s\_p_{bb}}(\mathtt{a}, \mathtt{d})$. Let us consider the partition $\mathcal{P} := P_1 \cup P_2$ of $\mathcal{R}^m$ with P_1 consisting of all answer rules in $\mathcal{R}^m$, while P_2 comprises all sub-query rules occurring in $\mathcal{R}^m$ including the seed rule. Separating answer and sub-query rules in this way allows for computing answer facts as early as possible. From Lemma 3.3 follows that the soft consequence operator can be applied for computing the well-founded model of the Magic DB transformed database $\mathcal{D}^m = \langle \mathcal{F} \cup \{\mathtt{m\_s\_p_{bb}}(\mathtt{a}, \mathtt{d})\}, \mathcal{R}^m \rangle$. The computation of $\mathtt{lfp}(T^s_{\mathcal{P}}, \mathcal{F} \cup \{\mathtt{m\_s\_p_{bb}}(\mathtt{a}, \mathtt{d})\})$ then induces the following sequence of sets:

$$
\begin{aligned}
F_1 &:= \mathcal{F} \cup \{\mathtt{m\_s\_p_{bb}}(\mathtt{a}, \mathtt{d})\} \\
F_2 &:= T^\star_{P_2}(F_1) = F_1 \cup \{\mathtt{m\_p_{bb}}(\mathtt{a}, \mathtt{d})\} \\
F_3 &:= T^\star_{P_2}(F_2) = F_2 \cup \{\mathtt{m\_p_{bb}}(\mathtt{1}, \mathtt{d}), \mathtt{m\_p_{bb}}(\mathtt{b}, \mathtt{d})\} \\
F_4 &:= T^\star_{P_2}(F_3) = F_3 \cup \{\mathtt{m\_p_{bb}}(\mathtt{2}, \mathtt{d}), \mathtt{m\_p_{bb}}(\mathtt{c}, \mathtt{d})\} \\
F_5 &:= T^\star_{P_1}(F_4) = F_4 \cup \{\mathtt{p_{bb}}(\mathtt{c}, \mathtt{d})\} \\
F_6 &:= T^\star_{P_1}(F_5) = F_5 \cup \{\mathtt{p_{bb}}(\mathtt{b}, \mathtt{d})\} \\
F_7 &:= T^\star_{P_1}(F_6) = F_6 \cup \{\mathtt{p_{bb}}(\mathtt{a}, \mathtt{d})\} \\
F_8 &:= T^\star_{P_2}(F_7) = F_7 \cup \{\mathtt{m\_p_{bb}}(\mathtt{3}, \mathtt{d}), \mathtt{m\_p_{bb}}(\mathtt{d}, \mathtt{d})\} \\
&\ldots \\
F_{105} &:= T^\star_{P_2}(F_{104}) = F_{104} \cup \{\mathtt{m\_p_{bb}}(\mathtt{100}, \mathtt{d})\} \\
F_{106} &:= F_{105}.
\end{aligned}
$$

The evaluation stops after computing the last fact set F_{106}, which means visiting all nodes in the graph corresponding to relation e by generating corresponding sub-query facts. The total well-founded model of the Magic DB transformed database $\mathcal{D}^m = \langle \mathcal{F} \cup \{\mathtt{m\_s\_p_{bb}}(\mathtt{a}, \mathtt{d})\}, \mathcal{R}^m \rangle$ is given by $\mathcal{M}_{\mathcal{D}^m} = \mathcal{M}^+_{\mathcal{D}^m} \cup \neg \cdot \overline{\mathcal{M}^+_{\mathcal{D}^m}}$ with

$$
\begin{aligned}
\mathcal{M}^+_{\mathcal{D}^m} = &\{m\_p_{bb}(X, d) \mid X \in \mathcal{U}_{\mathcal{D}^m}\} \cup \\
&\{p_{bb}(X, d) \mid X \in \{a, b, c, 1, 2, 3, 4, 5, 6, 7, 8, 9\}\} \cup \\
&\{\mathtt{m\_s\_p_{bb}}(\mathtt{a}, \mathtt{d})\} \cup \mathcal{F}
\end{aligned}
$$

while $\mathcal{U}_{\mathcal{D}^m}$ denotes the set of all constants occurring in $\mathcal{D}^m$, i.e., $\mathcal{U}_{\mathcal{D}^m}$ is the Herbrand universe of $\mathcal{D}^m$. It is obvious, however, that the computation could already have been stopped with the computation of F_7 after generating the answer fact $\mathtt{p_{bb}}(\mathtt{a},\mathtt{d})$. We therefore propose to slightly modify the Magic Sets transformed rules by incorporating a criterion for restricting the sub-query generation to those facts which are really necessary for answering an existential query. Consider again the above example but with the following modified magic rules

$$\begin{array}{ll}
\mathtt{p_{bb}}(\mathtt{X},\mathtt{Y}) & \leftarrow \mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{U},\mathtt{V}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Y}) \\
\mathtt{p_{bb}}(\mathtt{X},\mathtt{Y}) & \leftarrow \mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{U},\mathtt{V}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p_{bb}}(\mathtt{Z},\mathtt{Y}) \\
\mathtt{m_p_{bb}}(\mathtt{Z},\mathtt{Y},\mathtt{U},\mathtt{V}) & \leftarrow \mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{U},\mathtt{V}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \neg\mathtt{p_{bb}}(\mathtt{U},\mathtt{V}) \\
\mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{X},\mathtt{Y}) & \leftarrow \mathtt{m_s_p_{bb}}(\mathtt{X},\mathtt{Y}).
\end{array}$$

Within the derived sub-queries represented by magic literals, two additional parameters are used for storing the information about the top-query Q. Adding the negative answer literal $\neg\mathtt{p_{bb}}(\mathtt{U},\mathtt{V})$ to the third sub-query rule allows the generation of new sub-queries only as long as the top-query has not been successfully answered. Evaluating these rules using the soft consequence operator together with a similar partition as proposed above then induces the following sequence of sets:

$$\begin{array}{ll}
F_1 & := \mathcal{F} \cup \{\mathtt{m_s_p_{bb}}(\mathtt{a},\mathtt{d})\} \\
F_2 & := T^{\star}_{P_2}(F_1) = F_1 \cup \{\mathtt{m_p_{bb}}(\mathtt{a},\mathtt{d},\mathtt{a},\mathtt{d})\} \\
F_3 & := T^{\star}_{P_2}(F_2) = F_2 \cup \{\mathtt{m_p_{bb}}(\mathtt{1},\mathtt{d},\mathtt{a},\mathtt{d}), \mathtt{m_p_{bb}}(\mathtt{b},\mathtt{d},\mathtt{a},\mathtt{d})\} \\
F_4 & := T^{\star}_{P_2}(F_3) = F_3 \cup \{\mathtt{m_p_{bb}}(\mathtt{2},\mathtt{d},\mathtt{a},\mathtt{d}), \mathtt{m_p_{bb}}(\mathtt{c},\mathtt{d},\mathtt{a},\mathtt{d})\} \\
F_5 & := T^{\star}_{P_1}(F_4) = F_4 \cup \{\mathtt{p_{bb}}(\mathtt{c},\mathtt{d})\} \\
F_6 & := T^{\star}_{P_1}(F_5) = F_5 \cup \{\mathtt{p_{bb}}(\mathtt{b},\mathtt{d})\} \\
F_7 & := T^{\star}_{P_1}(F_6) = F_6 \cup \{\mathtt{p_{bb}}(\mathtt{a},\mathtt{d})\} \\
F_8 & := F_7.
\end{array}$$

The computation indeed shows the desired behavior as the existential query evaluation process is restricted to the generation of relevant (sub-)queries and answers, only. Note that the evaluation of the modified magic rules does not yield their corresponding well-founded model in which all answer facts are undefined. However, it is easy to see that applying the modified magic rules in the way described above represents an approach which is at least complete and sound with respect to the original existential query.

Instead of a single top-query Q, this shortened computation would also work for a set of existential queries stored in the binary seed relation $\mathtt{m\_s\_p_{bb}}$. This observation is important since we want to provide a solution to optimizing derived existential queries as well. As an example consider the two queries $Q_1 \equiv p(d, y)$ and $Q_2 \equiv p(9, y)$ which may be stored in the seed relation for initiating the query evaluation process. Using again the modified magic rules for evaluating these

queries would also avoid the generation of unnecessary sub-query facts, i.e., in this case the facts $\mathtt{m\_p_{bb}}(32,\mathtt{y},9,\mathtt{y}), \mathtt{m\_p_{bb}}(33,\mathtt{y},9,\mathtt{y}), \ldots \mathtt{m\_p_{bb}}(100,\mathtt{y},9,\mathtt{y})$ are not computed, after successfully deriving the answer fact $\mathtt{p_{bb}}(9,\mathtt{y})$. However, this example already indicates one of the deficiencies of this approach as required derived sub-query facts are possibly duplicated. For instance, the original sub-query fact $\mathtt{m\_p_{bb}}(\mathtt{e},\mathtt{y})$ is now represented twice by the facts $\mathtt{m\_p_{bb}}(\mathtt{e},\mathtt{y},\mathtt{d},\mathtt{y})$ and $\mathtt{m\_p_{bb}}(\mathtt{e},\mathtt{y},9,\mathtt{y})$ according to the two top-queries $Q_1 \equiv p(d,y)$ and $Q_2 \equiv p(9,y)$.

Before we discuss the optimization of derived existential queries in more detail, we will formally define the optimized magic rules for a set of existential (top-)queries by means of the *existential magic sets rewriting*. Similar to the presentation of Magic Templates in Section 4.1.2 we begin by specifying how magic literals are constructed and how the seed is derived from a set of queries in this context.

Definition 4.11 (Existential Magic Literal) *Let Qs be a set of existential queries with respect to a derived relation q, $A \equiv p_{ad}(\vec{x})$ be a positive literal with adornment* `ad` *and* $\mathtt{bd}(\vec{x})$ *the sequence of variables within $\vec{x}$ indicated as bound in the adornment* `ad`*. Then the existential magic literal of A is defined as*

$$\mathtt{exist\_magic}(A,Qs) := m\_p_{ad}(\mathtt{bd}(\vec{x}),\vec{y}).$$

where $\vec{y}$ is a vector of distinct variables $y_1,\ldots,y_n$ and n is the arity of q. If $A \equiv \neg p_{ad}(\vec{x})$ is a negative literal, then the magic literal of A is defined as $\mathtt{exist\_magic}(A,Qs) := m\_p_{ad}(\mathtt{bd}(\vec{x}),\vec{y})$.

Definition 4.12 (Existential Seed/Seed Rule) *Let $Qs = \{q_{ad}(\vec{c_1}), q_{ad}(\vec{c_2}), \ldots\}$ be a set of existential queries with respect to a derived relation q. Then the set of existential seeds for Qs is defined as*

$$\mathtt{exist\_seed}(Qs) := \{m\_s\_q_{ad}(\vec{c}) \mid q_{ad}(\vec{c}) \in Qs\}$$

and the corresponding existential seed rule is defined as

$$\mathtt{exist\_seed\_rule}(Qs) := m\_q_{ad}(\vec{x},\vec{x}) \leftarrow m\_s\_q_{ad}(\vec{x})$$

where $\vec{x}$ is a vector of distinct variables $x_1,\ldots,x_n$ and n is the arity of q.

Using the two definitions above, we can now specify the modified magic rules optimized with respect to a given set of existential (top-)queries.

Definition 4.13 (Existential Magic Rules) *Let $\mathcal{R}$ be a stratifiable deductive rule set, Qs a set of existential queries with respect to a derived relation $q \in \mathtt{pred}(\mathcal{R})$, and $\mathcal{R}^{Qs}$ the adorned rule set of $\mathcal{R}$ with respect to Qs. The Existential Magic Sets rewriting of $\mathcal{R}^{Qs}$ yields the magic rules $\mathtt{ems}(\mathcal{R}^{Qs})$ defined as the smallest set satisfying the following conditions:*

1. *For each deductive rule* $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^{Qs}$ *an answer rule of the form*

   $$A \leftarrow \texttt{exist\_magic}(A, Qs) \wedge L_1 \wedge \ldots \wedge L_n$$

   is in $\texttt{ems}(\mathcal{R}^{Qs})$.

2. *For each deductive rule* $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^{Qs}$ *and each derived body literal* L_i $(1 \leq i \leq n)$ *a sub-query rule of the form*

   $$\texttt{exist\_magic}(L_i, Qs) \leftarrow \texttt{exist\_magic}(A, Qs) \wedge L_1 \wedge \ldots \wedge L_{i-1} \wedge \neg q_{ad}(\vec{y})$$

   is in $\texttt{ems}(\mathcal{R}^{Qs})$ *where* $q_{ad}(\vec{y})$ *is an adorned q literal and* $\vec{y}$ *is a vector of variables which are used in* $\texttt{exist\_magic}(L_i, Qs)$ *wit respect to the set* Qs *as well.*

The following lemma shows the correctness of the existential magic sets rewriting according to a given set of existential queries.

Lemma 4.1 *Let* $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ *be a stratifiable database,* Qs *a set of existential queries with respect to a derived relation* $q \in \texttt{pred}(\mathcal{R})$*, and* $\texttt{ems}(\mathcal{R}^{Qs})$ *the existential magic rules of* $\mathcal{R}$ *with respect to* Qs*. Then the rule set* $\texttt{ems}(\mathcal{R}^{Qs}) \cup \texttt{exist\_seed\_rule}(Qs)$ *is always softly stratifiable and a soft partition* $\mathcal{P}$ *in which all answer rules are assigned to a smaller component of* $\mathcal{P}$ *than the sub-query rules derived from them always exists. Additionally, evaluating* $\texttt{lfp}(T^s_{\mathcal{P}}, \mathcal{F} \cup \texttt{exist\_seed}(Qs))$ *using the soft consequence operator* $T^s_{\mathcal{P}}$ *always yields the correct answer set with respect to* Qs*.*

Since it is immediate that the propositions of this lemma are true we refrain from presenting a formal proof of their correctness. Instead, we turn our attention to derived existential queries and show how the results from above can be used for optimizing them as well. As an example, let us consider the following additional rules for defining a relation r

$$\texttt{r(X, Y)} \leftarrow \texttt{p(X, Y)}$$
$$\texttt{r(X, Y)} \leftarrow \texttt{p(Y, X)}$$

and an existential query $Q \equiv r(c_1, c_2)$ with some constants c_1 and c_2. The corresponding (top-)query is given by the magic fact $m\_r_{bb}(c_1, c_2)$ inducing the two existential derived queries $m\_p_{bb}(c_1, c_2)$ and $m\_p_{bb}(c_2, c_1)$. However, applying the same technique as presented above for optimizing their evaluation leads to the following problem: As soon as one of them is successfully answered leading to an answer to the top-query, the evaluation of the other sub-query ought to be terminated. Instead, our approach fully evaluates the two optimized sub-queries

although one successful computation would suffice. A straightforward solution to this problem would be to allow the generation of sub-queries only as long as none of the existential queries from which they have been derived has been answered. In our example, this would lead to the following sub-query rules:

$$\begin{aligned}
&\mathtt{m_r_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{X},\mathtt{Y}) &&\leftarrow \mathtt{m_s_r_{bb}}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{X},\mathtt{Y},\mathtt{X},\mathtt{Y}) &&\leftarrow \mathtt{m_r_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{X},\mathtt{Y}) \wedge \neg \mathtt{r_{bb}}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_p_{bb}}(\mathtt{Y},\mathtt{X},\mathtt{Y},\mathtt{X},\mathtt{X},\mathtt{Y}) &&\leftarrow \mathtt{m_r_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{X},\mathtt{Y}) \wedge \neg \mathtt{r_{bb}}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_p_{bb}}(\mathtt{Z},\mathtt{X},\mathtt{O},\mathtt{P},\mathtt{U},\mathtt{V}) &&\leftarrow \mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{O},\mathtt{P},\mathtt{U},\mathtt{V}) \wedge \mathtt{e_{bb}}(\mathtt{X},\mathtt{Z}) \wedge \neg \mathtt{p_{bb}}(\mathtt{O},\mathtt{P}) \wedge \neg \mathtt{r_{bb}}(\mathtt{U},\mathtt{V})
\end{aligned}$$

Within the derived sub-queries for relation p, four additional parameters are used, the first two for storing the information about the existential queries against p, i.e., represented by the facts $m\_p_{bb}(c_1, c_2)$ and $m\_p_{bb}(c_2, c_1)$, and the second two arguments for storing the information about the existential query with respect to r, i.e., represented by the magic fact $m\_r_{bb}(c_1, c_2)$. Again it is quite obvious that an evaluation using the soft consequence operator as proposed above would be well-optimized with respect to the three existential queries. In general, however, adding new parameters and new negative answer literals in this way can really "blow-up" the rule set such that its evaluation becomes more expensive despite of existential query optimization effects. Therefore, we suggest to optimize only certain derived existential queries using the existential magic sets rewriting independent of other existential queries. We propose to solely optimize existential derived queries with respect to recursive or negatively referenced relations as this promises to be most effective. In our example, this would lead to the optimization of the derived existential queries with respect to the recursive relation p, resulting in the following sub-query rules

$$\begin{aligned}
&\mathtt{m_r_{bb}}(\mathtt{X},\mathtt{Y}) &&\leftarrow \mathtt{m_s_r_{bb}}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{X},\mathtt{Y}) &&\leftarrow \mathtt{m_r_{bb}}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_p_{bb}}(\mathtt{Y},\mathtt{X},\mathtt{Y},\mathtt{X}) &&\leftarrow \mathtt{m_r_{bb}}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_p_{bb}}(\mathtt{Z},\mathtt{X},\mathtt{O},\mathtt{P}) &&\leftarrow \mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y},\mathtt{O},\mathtt{P}) \wedge \mathtt{e_{bb}}(\mathtt{X},\mathtt{Z}) \wedge \neg \mathtt{p_{bb}}(\mathtt{O},\mathtt{P})
\end{aligned}$$

where the existential (top-)query $m\_r_{bb}(c_1, c_2)$ is not considered for optimization. Note that the existential magic sets rewriting of the rules for defining relation p is done with respect to the 'abstract' queries represented by the two magic sub-query rules

$$\begin{aligned}
&\mathtt{m_p_{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m_r_{bb}}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_p_{bb}}(\mathtt{Y},\mathtt{X}) \leftarrow \mathtt{m_r_{bb}}(\mathtt{X},\mathtt{Y})
\end{aligned}$$

in the original Magic Sets transformed rule set. However, these rules together with the corresponding answer rules are replaced by the existential magic rules. The correctness of the existential magic sets rewriting with respect to a certain subset of (derived) queries directly follows from Lemma 4.1.

4.4 Discussion

The Magic Set rewriting technique seems to be the most promising approach to evaluating database queries for database systems with a powerful view concept. This is in particular the case for systems which will implement the new SQL:1999 standard, and hence will allow the definition of stratifiable recursive views. The attractiveness of this method lies in its generality and efficiency. Additionally, in [GM92, MFPR90, MP94] it has been shown that the Magic Set method can improve the performance of nonrecursive queries as well. Thus, the Magic Set transformation and an appropriate fixpoint-based evaluation mechanism seem to be well-suited for being implemented on top of a (non)recursive relational database system in order to extend its functionality.

In this chapter we have introduced a new fixpoint-based evaluation method for computing the well-founded model of Magic Set transformed deductive databases. The main focus was to provide a simple method which allows for further refinement during a subsequent relational optimization phase and is at least as efficient as comparable fixpoint-based approaches. Therefore, it is a crucial point that the concept soft stratification restricts the evaluation of Magic Set transformed rules as little as necessary in order to be most flexible for the application of additional (orthogonal) rule optimization techniques. As an example we have shown how the incorporation of existential query optimization techniques may further enhance the evaluation of softly stratifiable rules. We will return to this issue in Section 5.2.3 as it represents an important optimization in the context of update propagation as well.

Soft stratification can serve as a basic evaluation mechanism applicable to other transformation-based approaches in deductive databases as well, e.g., soft update propagation in Chapter 5, efficient computation of general logic programs in Chapter 6 or methods for view updating. Although this approach plays an important role throughout this work, certain aspects are left undiscussed as they are beyond the scope of this thesis. These include, e.g., the applicability of the soft consequence operator in case of general unstratifiable deductive databases which do not result from a Magic Set transformation but are always guaranteed to have a two-valued well-founded model. Other questions concerning an efficient implementation would be how to incorporate a semi-naive evaluation technique and how efficient this approach performs for special classes of databases, e.g., linear ones [NRSU89], illustrating its advantages in more detail. Finally, in the SQL context it is necessary to consider this approach under bag semantics.

Chapter 5

Soft Update Propagation

In the field of deductive databases, a considerable amount of research has been devoted to the efficient computation of induced changes by means of update propagation. This technique has been mainly studied in order to provide methods for efficient incremental view maintenance and integrity checking in stratifiable databases. Additionally, update propagation methods based on bottom-up materialization seem to be particularly well-suited for updating distributed databases (e.g. [LMSS95]) or in the context of WWW applications for signaling changes of data sources to mediators (e.g. [GSUW94]).

The aim of update propagation is the computation of implicit changes of derived relations in a deductive database resulting from an explicitly performed update of its extensional fact base. As in most cases an update will affect only a small portion of the database, it is rarely reasonable to compute the induced changes by comparing the entire old and new database state. Instead, the implicit modifications should be iteratively computed by propagating the individual updates through the possibly affected rules and computing their consequences.

Within the last two decades, plenty of update propagation methods have been proposed, and the list [Dec86, LST87, BDM88, SK88, DW89, MB88, GL90, Wüt90, CW91, Oli91, VBK91, Küc91, GMS93, Man94, GL95, GM95, Gri97, BKR$^+$99, LR01, Pie01] is still not exhaustive. Although all these approaches essentially apply the same propagation techniques, they mainly differ in the way they are implemented and in the granularity of the computed induced updates. With respect to implementation, authors either propose propagation algorithms of their own or the application of deductive or active propagation rules. The different granularities considered result from integrity checking methods in which the propagation of induced changes may be simplified with respect to integrity rules. As this restricts the range of applications of update propagation, we will consider the smallest granularity of updates, so called true updates [Gri97], only. True updates correspond to real database changes excluding redundant or even false

induced updates. Moreover, we will use *deductive propagation rules*[1] allowing the bottom-up computation of induced true updates by means of fixpoint-based evaluation methods as proposed in Chapter 3.

Generally, for computing true updates, evaluations on both the old and new database state are necessary. In [Oli91], Olivé introduces the Internal Events Method which performs update propagation on one state only and derives the other state from the given one as well as the induced updates by means of *deductive transition rules*. A major advantage of such state simulation is that the underlying database system need not provide a mechanism allowing deduction on two different states. Similar transition rules have also been used by Bry et al. in [BDM88] and by Griefahn in [Gri97]. Propagation rules and transition rules together represent the *update propagation rules* which specify induced updates with respect to an extensional update and a given database state. In the context of pure bottom-up materialization, the benefit of these update propagation rules is that the evaluation of their rule bodies can be restricted to the values of the currently propagated update such that the entire propagation process is very naturally limited to the actually affected derived relations.

On the other hand, similar bottom-up approaches require to materialize the simulated state of derived relations completely in order to determine true updates. By contrast, if update propagation were based on a pure top-down approach, as proposed by Olivé [Oli91] and Küchenhoff [Küc91], the simulation of the opposite state can be easily restricted to the relevant part by querying the relevant portion of the database only. A pure top-down approach, however, has the disadvantage that the induced changes can only be determined by querying all existing derived relations, although most of them will probably not be affected by the update.

The structured update propagation method in [Gri97] combines the advantages of top-down and bottom-up propagation by applying the Magic rewriting to the update propagation rules mentioned above leading to potentially unstratifiable magic propagation rules. Therefore, structured update propagation is based on the alternating fixpoint computation [vG93] in order to determine the well-founded model of the possibly unstratifiable magic propagation rules correctly. The application of the alternating fixpoint computation, however, is not really efficient as the specific reason for unstratifiability (namely the application of the magic sets transformation to a stratified rule set) is not taken into account. For this reason, we will propose different update propagation rules which allow a more efficient evaluation based on the soft stratification approach from Chapter 4. The overall result of this chapter then is the soft update propagation approach for efficiently computing induced true updates.

Section 5.1 deals with the generation of update propagation rules which allow the incremental computation of true updates. Section 5.2 presents the soft update

[1] Deductive propagation rules have been used in [VBK91, Küc91, Oli91, UO92, Man94, Gri97] as well.

propagation approach. In Section 5.3 its application to incremental maintenance of materialized views and integrity checking is described. Section 5.4 concludes this chapter with a discussion of our proposed update propagation method.

5.1 Incremental Update Propagation

In Section 2.4 we introduced the notions *update* $u_D = \langle u_D^+, u_D^- \rangle$ and *induced update* $u_{D \to D'} = \langle u_{D \to D'}^+, u_{D \to D'}^- \rangle$ with respect to a given deductive database $\mathcal{D}$ for specifying modifications of extensional relations in $\mathcal{D}$ and the overall modifications of $\mathcal{D}$, respectively. The following lemma summarizes the most essential properties of true induced updates.

Lemma 5.1 (Properties of Induced Updates) *Let $\mathcal{D}$ be a stratifiable deductive database, $\mathcal{M}_\mathcal{D}$ the semantics of $\mathcal{D}$, $u_\mathcal{D}$ an update and $u_{D \to D'} = \langle u_{D \to D'}^+, u_{D \to D'}^- \rangle$ the corresponding true induced update. As $u_{D \to D'}$ represents the exact difference between the two database states, the sets of induced insertions and deletions are disjoint and both the new and old database state can be constructed from the other one, respectively, and the true induced update sets:*

$$\begin{array}{lcl} u_{D \to D'}^+ \cap u_{D \to D'}^- & = & \emptyset \\ \mathcal{M}_{\mathcal{D}'}^+ & = & (\mathcal{M}_\mathcal{D}^+ \setminus u_{D \to D'}^-) \uplus u_{D \to D'}^+ \\ \mathcal{M}_\mathcal{D}^+ & = & (\mathcal{M}_{\mathcal{D}'}^+ \setminus u_{D \to D'}^+) \uplus u_{D \to D'}^- \end{array}$$

Proof: The properties immediately follow from Definitions 2.22 and 2.23. □

The task of update propagation is to constructively determine the overall effect of the update $u_{D \to D'}$. This is achieved by determining a set of delta facts for every affected relation which may be stored in corresponding delta relations (cf. Definition 2.24). In the following, we develop deductive rules for defining such delta relations by providing transformations for deriving propagation rules in Section 5.1.1 and transition rules in Section 5.1.2 from a given base update and database.

5.1.1 Propagation Rules for True Updates

In this section we develop propagation rules for defining delta relations which represent the changes of the original relations induced by a certain base update. Delta relations reflect the original database schema in such a way that the explicitly performed base updates are represented by extensional delta relations while derived updates are described by rule-defined ones. For efficiency reasons we allow to reference delta relations in the body of propagation rules as well such that their evaluation is restricted to already computed induced updates. In order to abstract from negative and positive occurrences of atoms in rule bodies, we

use the superscripts "$^+$" and "$^-$" for indicating what kind of delta relation is to be used. For a positive literal $A \equiv p(t_1, \ldots, t_n)$ we define $A^+ \equiv \Delta^+p(t_1, \ldots, t_n)$ and $A^- \equiv \Delta^-p(t_1, \ldots, t_n)$. For a negative literal $L \equiv \neg A$, we use $L^+ := A^-$ and $L^- := A^+$.

In the sequel, we call a delta relation Δ^+p *positive* while Δ^-p is denoted *negative*. Additionally, a literal L which references a delta relation is called *delta literal*. If $\texttt{pred}(L) = \Delta^+p$, then L is called a *positive delta literal*, and if $\texttt{pred}(L) = \Delta^-p$ it is denoted a *negative delta literal*.

For computing the derived delta relations, the explicit changes caused by a base update have to be represented by the extensional delta relations. Thus, quite similar to query seeds used in the Magic Sets method, we generate a set of delta facts called *propagation seeds* for an explicitly performed base update.

Definition 5.1 (Propagation Seeds) *Let $\mathcal{D}$ be a stratifiable deductive database and $u_D = \langle u_D^+, u_D^- \rangle$ a base update. The set of propagation seeds* $\texttt{prop\_seeds}(u_D)$ *with respect to u_D is*

$$\texttt{prop\_seeds}(u_D) := \{\ \Delta^\pi p(c_1, \ldots, c_n) \mid p(c_1, \ldots, c_n) \in u_D^\pi \text{ and } \pi \in \{+, -\}\}.$$

The extensional delta relations represent the starting point from which induced updates are to be computed. An update propagation method can only be efficient if most derived facts eventually rely on at least one fact in an extensional delta relation.

Before defining propagation rules, we still have to introduce one more notion. As already mentioned above, within the propagation rules references to both the old and new database state are necessary. We will use corresponding meta predicates `old` and `new` for these references in the bodies of propagation rules, and assume that evaluations on both states are correctly performed by the underlying database system. When transition rules for state simulation are considered in Section 5.1.2, we no longer assume these predicates to be meta predicates but mappings which syntactically transform the predicate symbols of the literals they are applied to. We can now introduce incremental propagation rules for true updates as proposed in [Gri97]. Since an induced insertion or induced deletion can be simply represented by the difference between the two consecutive database states, the propagation rules may be defined as follows:

Definition 5.2 (Propagation Rules) *Let $\mathcal{R}$ be a stratifiable deductive rule set. The set of propagation rules for true updates with respect to $\mathcal{R}$ is denoted $\varphi(\mathcal{R})$ and is defined as follows:*

1. *For each rule $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}$ and each body literal L_i $(i = 1, \ldots, n)$ two propagation rules of the form*

$$A^+ \leftarrow L_i^+ \wedge \texttt{new}(L_1 \wedge \ldots \wedge L_{i-1} \wedge L_{i+1} \wedge \ldots \wedge L_n) \wedge \texttt{old}\,\neg A$$
$$A^- \leftarrow L_i^- \wedge \texttt{old}(L_1 \wedge \ldots \wedge L_{i-1} \wedge L_{i+1} \wedge \ldots \wedge L_n) \wedge \texttt{new}\,\neg A$$

are in $\varphi(\mathcal{R})$*. The literals* `new` L_j *and* `old` L_j $(j = 1, \ldots, i-1, i+1, \ldots, n)$ *are called side literals of* L_i^+ *and* L_i^-*, respectively.*

2. *No other rules are in* $\varphi(\mathcal{R})$*.*

The propagation rules basically perform a comparison of the `old` and `new` database state while providing a focus on individual updates by applying the delta literals L_i^π with $\pi \in \{+, -\}$. Each propagation rule body may be divided into two parts:

1. The *derivability test* $(L_i^\pi \wedge \{\texttt{new}|\texttt{old}\}\ (L_1 \wedge \ldots \wedge\ L_{i-1} \wedge\ L_{i+1} \wedge \ldots \wedge\ L_n))$ is performed in order to determine whether A is derivable in the new or old state, respectively. Basically, it is responsible for calculating potential updates [Gri97].

2. The *effectiveness test*[2] ($\{\texttt{new} \mid \texttt{old}\}(\neg A)$) checks whether the fact obtained by the derivability test is not derivable in the opposite state. Hence, it checks whether the potential updates obtained by the derivability test are effective.

Semantically, however, two other tasks can be identified depending on the database state a literal refers to. The *safeness test* (`new`-derivations) takes care that only safe updates are derived while the *trueness test* (`old`-derivations) takes care that only true updates are propagated by checking whether a safe update is not redundant with respect to the old database. For more details on different granularities of updates and their relation to deducible tests we refer to [Gri97].

As the derivability test for defining a delta relation `pred`(A^π) refers to one delta literal L_i^+ or L_i^- respectively, a fact for relation `pred`(A) is considered potentially inserted if an update adds a derivation path for it and possibly deleted if it removes one. In contrast to the derivability test, the effectiveness test solely refers to the derived relation `pred`(A) ensuring that the fact inserted (respectively deleted) is not derivable in the old (respectively new) state. In general, this test cannot be further specialized, as it is needed to detect alternative derivations caused by other rules defining the respective relation.

The obtained propagation rules and seeds can be added to the original database yielding a safe and stratifiable database which is called *augmented database* in the following[3]. The safeness of propagation rules immediately follows from the safeness of the original rules. The only negative literal newly introduced corresponds to the head of the transformed rule, so that the respective variables are guaranteed to be bound by the remaining positive body literals. Furthermore, the propagation rules cannot jeopardize stratifiability, as delta relations are always positively referenced and hence cannot participate in any cycle involving negation.

[2]The effectiveness test is called *derivability test* in [VBK91] and *redundancy test* in [Küc91].
[3]The notion *augmented database* has been coined in [Oli91]

Example 5.1 *Let us consider again the rules from example 2.1 for defining the derived relations* `path` *and* `one_way`*:*

1. $\texttt{one\_way}(\texttt{X}) \leftarrow \texttt{path}(\texttt{X},\texttt{Y}) \wedge \neg\texttt{path}(\texttt{Y},\texttt{X})$
2. $\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{edge}(\texttt{X},\texttt{Y})$
3. $\texttt{path}(\texttt{X},\texttt{Y}) \leftarrow \texttt{edge}(\texttt{X},\texttt{Z}) \wedge \texttt{path}(\texttt{Z},\texttt{Y})$

The corresponding propagation rules are as follows (In the sequel, the relation symbols will be abbreviated by their first letter.):

1. $\begin{aligned}
\Delta^{+}\texttt{o}(\texttt{X}) &\leftarrow \Delta^{+}\texttt{p}(\texttt{X},\texttt{Y}) \wedge \texttt{new}\ \neg\,\texttt{p}(\texttt{Y},\texttt{X}) \wedge \texttt{old}\ \neg\texttt{o}(\texttt{X})\\
\Delta^{+}\texttt{o}(\texttt{X}) &\leftarrow \Delta^{-}\texttt{p}(\texttt{Y},\texttt{X}) \wedge \texttt{new}\ \ \texttt{p}(\texttt{X},\texttt{Y}) \wedge \texttt{old}\ \neg\texttt{o}(\texttt{X})\\
\Delta^{-}\texttt{o}(\texttt{X}) &\leftarrow \Delta^{-}\texttt{p}(\texttt{X},\texttt{Y}) \wedge \texttt{old}\ \neg\,\texttt{p}(\texttt{Y},\texttt{X}) \wedge \texttt{new}\ \neg\texttt{o}(\texttt{X})\\
\Delta^{-}\texttt{o}(\texttt{X}) &\leftarrow \Delta^{+}\texttt{p}(\texttt{Y},\texttt{X}) \wedge \texttt{old}\ \ \texttt{p}(\texttt{X},\texttt{Y}) \wedge \texttt{new}\ \neg\texttt{o}(\texttt{X})
\end{aligned}$

2. $\begin{aligned}
\Delta^{+}\texttt{p}(\texttt{X},\texttt{Y}) &\leftarrow \Delta^{+}\texttt{e}(\texttt{X},\texttt{Y}) \wedge \texttt{old}\ \neg\texttt{p}(\texttt{X},\texttt{Y})\\
\Delta^{-}\texttt{p}(\texttt{X},\texttt{Y}) &\leftarrow \Delta^{-}\texttt{e}(\texttt{X},\texttt{Y}) \wedge \texttt{new}\ \neg\texttt{p}(\texttt{X},\texttt{Y})
\end{aligned}$

3. $\begin{aligned}
\Delta^{+}\texttt{p}(\texttt{X},\texttt{Y}) &\leftarrow \Delta^{+}\texttt{e}(\texttt{X},\texttt{Z}) \wedge \texttt{new}\ \ \texttt{p}(\texttt{Z},\texttt{Y}) \wedge \texttt{old}\ \neg\texttt{p}(\texttt{X},\texttt{Y})\\
\Delta^{+}\texttt{p}(\texttt{X},\texttt{Y}) &\leftarrow \Delta^{+}\texttt{p}(\texttt{Z},\texttt{Y}) \wedge \texttt{new}\ \ \texttt{e}(\texttt{X},\texttt{Z}) \wedge \texttt{old}\ \neg\texttt{p}(\texttt{X},\texttt{Y})\\
\Delta^{-}\texttt{p}(\texttt{X},\texttt{Y}) &\leftarrow \Delta^{-}\texttt{e}(\texttt{X},\texttt{Z}) \wedge \texttt{old}\ \ \texttt{p}(\texttt{Z},\texttt{Y}) \wedge \texttt{new}\ \neg\texttt{p}(\texttt{X},\texttt{Y})\\
\Delta^{-}\texttt{p}(\texttt{X},\texttt{Y}) &\leftarrow \Delta^{-}\texttt{p}(\texttt{Z},\texttt{Y}) \wedge \texttt{old}\ \ \texttt{e}(\texttt{X},\texttt{Z}) \wedge \texttt{new}\ \neg\texttt{p}(\texttt{X},\texttt{Y})
\end{aligned}$

Note that the upper indices π of the delta literal $\Delta^{\pi}\texttt{p}(\texttt{Y},\texttt{X})$ in the propagation rules for defining $\Delta^{\overline{\pi}}\texttt{o}(\texttt{X})$ are inverted as p is negatively referenced by the corresponding literal in the original rule.

Each propagation rule in Example 5.1 includes one delta literal for restricting the evaluation to the changes induced by the respective body literal. Thus, we obtain one propagation rule for each possible update (i.e., insertion or deletion) of each body literal. For each original rule $2n$ propagation rules are generated if n is the number of body literals. However, quite similar to the delta rules for differential fixpoint computation (cf. Section 3.1) it is possible to substitute not only a single body literal but any subset of them by a corresponding delta literal. This approach can provide a much better focus on propagated updates but would lead to an exponential number of propagation rules in the augmented database.

Another deficiency of the propagation rules from the example above is that a bottom-up materialization, as discussed in Chapter 3, will nevertheless determine both the new as well as the old state of the relations `path` and `one_way` completely. The reason is that the supposed evaluation over the two consecutive database states is performed using deductive rules which are not specialized with respect to the particular updates that are propagated. If propagation was based on a top-down evaluation technique, this problem would not occur. Then the bindings of

delta literals could be easily passed to the remaining literals in the rule bodies such that their evaluation is restricted to the affected part of the database only. This obvious weakness of propagation rules in view of a bottom-up materialization is cured by incorporating Magic Sets optimizations as proposed in Griefahn's structured update propagation approach [Gri97].

The following proposition shows that if propagation rules are generated according to Definition 5.2, the delta relations will correctly represent the corresponding induced update. Note that the proof of Proposition 5.3 is basically adopted from [Gri97], but it is included in order to make this chapter self-contained.

Proposition 5.3 (Correctness of Propagation Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable database, $u_\mathcal{D}$ an update and $u_{D \to D'} = \langle u^+_{D \to D'}, u^-_{D \to D'} \rangle$ the corresponding induced update from $\mathcal{D}$ to $\mathcal{D}'$. Let $\mathcal{D}^p = \langle \mathcal{F} \cup \texttt{prop\_seeds}(u_\mathcal{D}), \mathcal{R} \cup \varphi(\mathcal{R}) \rangle$ be the augmented deductive database of $\mathcal{D}$. Then the delta relations defined by the propagation rules $\varphi(\mathcal{R})$ correctly represent the induced update $u_{D \to D'}$. Hence, for each relation $p \in \texttt{pred}(\mathcal{D})$ the following conditions hold:*

$$\Delta^+ p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^p} \iff p(\vec{t}\,) \in u^+_{D \to D'}$$
$$\Delta^- p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^p} \iff p(\vec{t}\,) \in u^-_{D \to D'} \ .$$

Proof: The proposition is shown by induction on the depth of proof trees[4] for A (respectively A^+) with respect to $\mathcal{D}'$ (respectively $\mathcal{D}^a$). Additionally, the proof is solely performed for insertions, as the corresponding result for deletions can be shown by analogy. We assume that the meta predicates `old` and `new` are correctly evaluated with respect to the database states $\mathcal{M}_\mathcal{D}$ and $\mathcal{M}_{\mathcal{D}'}$, respectively.

1) We show by induction on the depth d of proof trees with respect to $\mathcal{D}'$ that the implication $A \in u^+_{D \to D'} \Rightarrow A^+ \in \mathcal{M}_{\mathcal{D}^p}$ holds. Suppose that $A \in u^+_{D \to D'}$. Then there exists a proof tree for A with respect to $\mathcal{D}'$, but none with respect to $\mathcal{D}$.

   Suppose that $d = 0$: In this case, A refers to an extensional relation and hence $A \in u^+_\mathcal{D}$. From Definition 5.1 follows that $A^+ \in \texttt{prop\_seeds}(u_\mathcal{D}) \subseteq \mathcal{M}_{\mathcal{D}^p}$.

   Suppose that $d > 0$: We assume that the implication holds for all atoms in $u^+_{D \to D'}$ having a proof tree with respect to $\mathcal{D}'$ of depth less than d. As $d > 0$, A has children $L_1\sigma, \ldots, L_n\sigma$ and $A \equiv B\sigma$ is derived via

   $$R \equiv B \leftarrow L_1 \wedge \ldots \wedge L_n$$

[4] A proof tree for a ground literal L is a tree of ground literals where each internal node $L' \equiv A'$ has children $L_i\sigma$ (if there exists a ground instance $R\sigma$ of a rule $R \equiv A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}$ such that $A' \equiv A\sigma$) or is a leaf (if $A' \in \mathcal{F}$).

where σ is a ground substitution for all variables in R. As no proof tree exists for A with respect to $\mathcal{D}$, at least one of the $L_i\sigma$ is not derivable in $\mathcal{D}$. If $L_i\sigma$ is a positive literal, then $L_i\sigma \in u^+_{D\to D'}$. As $L_i\sigma$ has a proof tree of depth $< d$ with respect to $\mathcal{D}'$, $L_i^+\sigma \in \mathcal{M}_{\mathcal{D}^p}$ follows from the induction hypothesis. If $L_i\sigma \equiv \neg C_i\sigma$ is a negative literal, then $C_i\sigma \in u^-_{D\to D'}$. It can be shown by induction on the depth of proof trees with respect to $\mathcal{D}$ that then $L_i^+\sigma \equiv C_i^-\sigma \in \mathcal{M}_{\mathcal{D}^p}$ holds, but this part of the proof is omitted since it can be performed by analogy to the proof for deletions. From these two cases follows that $L_i^+\sigma \in \mathcal{M}_{\mathcal{D}^p}$. According to Definition 5.2, for each body literal of R a positive propagation rule is generated such that the rule

$$B^+ \leftarrow L_i^+ \wedge \texttt{new}(L_1 \wedge \ldots \wedge L_n) \wedge \texttt{old}\neg B$$

is in $\varphi(\mathcal{R})$. As evaluations over `new` and `old` are correct, it follows that $A^+ \equiv B^+\sigma$ is derivable in $\mathcal{D}^p$, i.e., $A^+ \in \mathcal{M}_{\mathcal{D}^p}$.

2) We show by induction on the depth d of proof trees with respect to $\mathcal{D}^p$ that the implication $A^+ \in \mathcal{M}_{\mathcal{D}^p} \Rightarrow A \in u^+_{D\to D'}$ holds. Suppose that $A^+ \in \mathcal{M}_{\mathcal{D}^p}$. Then there must be a proof tree for A^+ with respect to $\mathcal{D}^p$, whose depth shall be denoted d.

Suppose that $d = 0$: In this case, A^+ refers to an extensional delta relation and $A \in u^+_{\mathcal{D}} \subseteq u^+_{D\to D'}$ directly follows from Definition 5.1 of propagation seeds.

Suppose that $d > 0$: We assume that the implication holds for all delta facts which represent induced insertions having a proof tree with respect to $\mathcal{D}^p$ the depth of which is less than d. As $d > 0$, A^+ has children $L_i^+\sigma$, $\texttt{new}L_1\sigma$, ..., $\texttt{new}L_{i-1}\sigma$, $\texttt{new}L_{i+1}\sigma$, ... , $\texttt{new}L_n\sigma$, $\texttt{old}\neg B\sigma$ and $A^+ \equiv B^+\sigma$ is derived via the propagation rule

$$B^+ \leftarrow L_i^+ \wedge \texttt{new}(L_1 \wedge \ldots \wedge L_n) \wedge \texttt{old}\neg B.$$

The child $L_i^+\sigma$ of A^+ has a proof tree with respect to $\mathcal{D}^p$ of depth $< d$. If $L_i^+\sigma$ is a positive delta literal, then $L_i^+\sigma \in \mathcal{M}_{\mathcal{D}^p}$ and from the induction hypothesis follows that $L_i^+\sigma \in u^+_{D\to D'}$. If $L_i^+\sigma C_i^-\sigma$ is a negative delta literal, then $C_i^-\sigma \in \mathcal{M}_{\mathcal{D}^p}$ and it can be shown by induction on the depth of proof trees with respect to $\mathcal{D}^p$ that $C_i\sigma \in u^-_{D\to D'}$. (This part of the proof is omitted for the same reasons as above.) This shows that $L_i\sigma \in \mathcal{M}_{\mathcal{D}'}$. As the side literals $\texttt{new}(L_1 \wedge \ldots \wedge L_{i-1} \wedge L_{i+1} \wedge \ldots L_n)$ are correctly evaluated, it additionally follows that

$$\mathcal{M}_{\mathcal{D}'} \models (L_1 \wedge \ldots L_{i-1} \wedge L_{i+1} \wedge \ldots \wedge L_n)\sigma$$

and thus $B\sigma \in \mathcal{M}_{D'}$ due to $B \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}'$. The effectiveness test proves that $B\sigma \notin \mathcal{M}_{\mathcal{D}}$ which finally shows that $A \equiv B\sigma \in u^+_{D \rightarrow D'}$.

□

The propagation rules can be determined at schema definition time and don't have to be recompiled each time a new base update is applied. The presented transformation-based approach allows the specification of propagation rules for true updates only, though it can be extended to describe the modifications induced by a certain base update at an arbitrary granularity, as proposed in [Gri97, MK88] for example. The consideration of different granularities allows for cutting down the cost of propagation as long as accurate results are not required.

For propagating true updates the truth value of the updated facts in the old as well as in the new state is essential and is determined by the derivability and effectiveness test. However, the propagation rules can be further enhanced by dropping the effectiveness test or by either refining or even omitting the derivability test in some cases. As an example, consider a derived relation which is defined without an implicit union or projection. In this case, no multiple derivations of facts are possible, and thus the effectiveness test in the corresponding propagation rules can be completely omitted. In the sequel, however, we will not consider these specialized propagation rules any further as these optimizations are orthogonal to the following discussion. We will now turn our attention to the rule based simulation of database states by means of transition rules.

5.1.2 Transition Rules for True Updates

Generally, for computing true updates references to both the old and new database state are necessary. Up to now, we have considered propagation rules containing explicit references to both states, and their correct evaluation was assumed to be guaranteed by the underlying database system. The purpose of this section is to investigate the possibility of dropping the explicit references to one of the states by deriving it from the other one and the given updates. The benefit of such a state simulation is that the database system is not required to store both states explicitly but may work on one state only. The deductive rules defining the simulated state will be called *transition rules* according to the naming in [Oli91].

Although both directions are possible, we will concentrate on a somehow pessimistic approach, the simulation of the new state while the old one is actually given. The following discussion, however, can be easily transferred to the case of simulating the old state. In principle, transition rules can be differentiated by the way how far induced updates are considered for simulating the other database state. We start with the definition of *naive transition rules* which derive the new state from the physically present old fact base and the explicitly given updates. The disadvantage of these transition rules is, however, that each derivation with

respect to the new state has to go back to the extensional delta relations and hence makes no use of the implicit updates already derived during the course of propagation. In the Internal Events Method [Oli91] as well as in [Man94] it has been proposed to improve state simulation by employing not only the extensional delta relations but the derived ones as well. However, the union of the original, the propagation and this kind of transition rules is not stratifiable, if the database includes recursively defined relations, and may even not represent the true induced update anymore under the well-founded semantics [Gri97]. In [Gri97] such stratification problems are avoided by introducing so-called *incremental transition rules* containing references to certain derived updates only. Naive as well as incremental transition rules can be applied to the update propagation methods presented in subsequent sections. As both kinds of transition rules allow a complete and sound propagation of updates having a distinct influence on the efficiency of the underlying propagation process.

We start by considering the simulation of the new state by means of naive transition rules. To this end, we assume that the base updates are not yet physically performed on the database but are only represented in the extensional delta relations. From Lemma 5.1 we know that the new state can be computed from the old one and the true induced update $u_{D\rightarrow D'} = \langle u^+_{D\rightarrow D'}, u^-_{D\rightarrow D'}\rangle$:

$$\mathcal{M}^+_{\mathcal{D}'} \;=\; (\mathcal{M}^+_{\mathcal{D}} \setminus u^-_{D\rightarrow D'}) \uplus u^+_{D\rightarrow D'}.$$

This equation directly leads to an equivalence on the level of tuples

$$\texttt{new}\ A \iff (\texttt{old}\ A \wedge \neg(A^-)) \vee A^+.$$

which holds if the referenced delta relations correctly describe the induced update $u_{D\rightarrow D'}$. Note that we assume the precedence of the superscripts "$^+$" and "$^-$" to be higher than the one of $\neg$. Thus, we can omit the brackets in $\neg(A^-)$ and simply write $\neg A^-$.

According to Definition 5.1 the delta relations of the propagation seeds correctly correspond to the base update. Thus, using the equivalence above the deductive rules for inferring the new state of extensional relations can be easily derived. For instance, for the extensional relation `edge/2` of our Example 5.1 the new state is specified by the rules

$$\texttt{new e(X,Y)} \leftarrow \texttt{old e(X,Y)} \wedge \neg\Delta^-\texttt{e(X,Y)}$$
$$\texttt{new e(X,Y)} \leftarrow \Delta^+\texttt{e(X,Y)},$$

where the first rule specifies the unchanged portion of edge and the second one the facts that are added. In the following, such rules will be denoted *direct transition rules* according to the naming in [Gri97] as they directly define the new state of a relation by means of its old state and its own delta relations.

From the new states of extensional relations we can successively infer the new states of derived relations using the dependencies given by the original rule set. To this end, the original rules are duplicated and a `new` mapping is applied to all predicate symbols occurring in the new rules. For instance, the rules

$$\begin{aligned}
&\texttt{new o(X)} &&\leftarrow \texttt{new p(X,Y)} \wedge \neg\texttt{new p(Y,X)}\\
&\texttt{new p(X,Y)} &&\leftarrow \texttt{new e(X,Y)}\\
&\texttt{new p(X,Y)} &&\leftarrow \texttt{new e(X,Z)} \wedge \texttt{new p(Z,Y)}
\end{aligned}$$

specify the new state of the relations `path/2` and `one_way/2`. As transition rules of this structure solely infer the new state of a derived relation from the new states of the underlying relations, they will be denoted *indirect transition rules*. Again this denotation has been adopted from [Gri97].

In order to provide a homogeneous view on the propagation rules presented in Section 5.1.1 and the transition rules introduced in the sequel, we stick to the usage of `new` and `old` literals. However, we no longer assume them to be meta predicates but mappings which syntactically transform the relation symbols of literals they are applied to. As we consider the simulation of the new state only, the `old` mapping is assumed to be the identity on literals such that their evaluation is performed with respect to the original relations. As derivations on the new state are to be done with respect to the relations specified by the transition rules, the `new` mapping actually replaces the predicate symbols by corresponding new ones. Note that the application of $\neg$ and the mappings `new` respectively `old` are orthogonal, i.e., $\texttt{new}\neg A \equiv \neg\texttt{new}\ A$ and $\texttt{old}\neg A \equiv \neg\texttt{old}\ A$. Hence, the negative referenced path literal $\texttt{new}\neg\texttt{p(Y,X)}$ from the example above may be replaced by $\neg\texttt{new p(Y,X)}$.

We will now define naive transition rules using direct transition rules for extensional relations and indirect ones for derived relations as proposed above.

Definition 5.4 (Naive Transition Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable deductive database. Then the set of naive transition rules for true updates and new state simulation with respect to $\mathcal{R}$ is denoted $\tau_n(\mathcal{R})$ and is defined as follows:*

1. *For each n-ary extensional predicate symbol $p \in \texttt{pred}(\mathcal{F})$, the direct transition rules*

   $$\begin{aligned}
   &\texttt{new}\ A \leftarrow \texttt{old}\ A \wedge \neg A^-\\
   &\texttt{new}\ A \leftarrow A^+
   \end{aligned}$$

   are in $\tau_n(\mathcal{R})$ where $A \equiv p(x_1, \ldots, x_n)$, and the x_i $(i = 1, \ldots, n)$ are distinct variables.

2. *For each rule $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}$, an indirect transition rule of the form*

$$\texttt{new}\ A \leftarrow \texttt{new}\ (L_1 \wedge \ldots \wedge L_n)$$

is in $\tau_n(\mathcal{R})$.

3. *No other rules are in* $\tau_n(\mathcal{R})$.

It is obvious that if $\mathcal{R}$ is stratifiable, the rule set $\mathcal{R} \uplus \varphi(\mathcal{R}) \uplus \tau_n(\mathcal{R})$ must be stratifiable as well. The following proposition shows that if a deductive database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ is augmented with the naive transition rules $\tau_n(\mathcal{R})$ constructed from $\mathcal{R}$, the propagation rules $\varphi(\mathcal{R})$ as well as the propagation seeds $\texttt{prop\_seeds}(u_D)$ with respect to a base update u_D, then the transition rules correctly define the new database state, and the delta relations correctly represent the induced update.

Proposition 5.5 (Correctness of Naive Transition Rules) *Let* $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ *be a stratifiable database,* $u_\mathcal{D}$ *an update and* $u_{D \to D'} = \langle u^+_{D \to D'}, u^-_{D \to D'} \rangle$ *the corresponding induced update from* $\mathcal{D}$ *to* $\mathcal{D}'$. *Let* $\mathcal{D}^p = \langle \mathcal{F} \cup \texttt{prop\_seeds}(u_\mathcal{D}), \mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau_n(\mathcal{R}) \rangle$ *be the augmented deductive database of* $\mathcal{D}$. *Then* $\mathcal{D}^p$ *correctly represents the implicit state of* $\mathcal{D}'$, *i.e., for all atoms* $A \in \mathcal{H}_{D'}$

$$A \in \mathcal{M}_{D'} \iff \texttt{new}\ A \in \mathcal{M}_{\mathcal{D}^p},$$

and all delta relations defined by the propagation rules $\varphi(\mathcal{R})$ *correctly represent the induced update* $u_{D \to D'}$, *i.e., for* $A \equiv p(\vec{t}\,)$:

$$\Delta^+ p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^p} \iff p(\vec{t}\,) \in u^+_{D \to D'}$$
$$\Delta^- p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^p} \iff p(\vec{t}\,) \in u^-_{D \to D'}\ .$$

Proof: The correctness of the new state simulation follows from the fact that the propagation seeds truly represent the given base update and that the new state of the extensional relations in $\mathcal{D}$ is correctly simulated using the properties from Lemma 5.1. As the remaining transition rules are a copy of the original rules in $\mathcal{R}$ with the predicate symbols consistently replaced by new ones, their evaluation is based on the correctly simulated new states of the extensional relations in $\mathcal{D}$ and hence, must be correct as well. Thus, for a database $\mathcal{D}^* = \langle \mathcal{F} \cup \texttt{prop\_seeds}(u_\mathcal{D}), \mathcal{R} \cup \tau_n(\mathcal{R}) \rangle$ and all atoms $A \in \mathcal{H}_{D'}$ the following holds:

$$A \in \mathcal{M}_{D'} \iff \texttt{new}\ A \in \mathcal{M}_{\mathcal{D}^*}.$$

The correctness of delta relations follows from the fact that the propagation rules $\varphi(\mathcal{R})$ soundly represent the induced update (Proposition 5.3). Thus, the rule sets $\varphi(\mathcal{R})$ and $\mathcal{R} \cup \tau_n(\mathcal{R})$ correctly represent the induced update and the new state, respectively.

The only question left is whether the evaluation of the entire rule set, i.e., the union of all rules $\mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau_n(\mathcal{R})$, still remains correct. This follows from the fact that every derived relation is solely defined by one of the three rule sets, i.e., $\texttt{pred}(\mathcal{R}) \cap \texttt{pred}(\varphi(\mathcal{R})) = \emptyset$, $\texttt{pred}(\mathcal{R}) \cap \texttt{pred}(\tau_n(\mathcal{R})) = \emptyset$ and $\texttt{pred}(\tau_n(\mathcal{R})) \cap \texttt{pred}(\varphi(\mathcal{R})) = \emptyset$, and that the union is still stratifiable such that the following holds:

$$\mathcal{M}_{\langle \mathcal{M}_{\mathcal{D}^*}, \mathcal{R} \cup \varphi(\mathcal{R}) \rangle} = \mathcal{M}_{\langle \mathcal{F} \cup \texttt{prop\_seeds}(u_\mathcal{D}), \mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau_n(\mathcal{R}) \rangle}.$$

Thus, the evaluation of the rule sets $\varphi(\mathcal{R})$ and $\mathcal{R} \cup \tau_n(\mathcal{R})$ remains correct if the union of the entire rule set $\mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau_n(\mathcal{R})$ is considered.

□

Although it seems to be obvious to simulate the new database state by means of naive transition rules, only base updates are used and thus, induced updates computed by the propagation rules $\varphi(\mathcal{R})$ cannot enhance the evaluation of transition rules. Adding direct transition rules to the set $\tau_n(\mathcal{R})$, however, may lead to an unstratifiable rule set which may even not represent the induced update in any case anymore. Therefore, in [Gri97] it has been proposed to consider only a certain combination of indirect and direct transition rules such that the resulting rule set remains stratifiable. The basic idea is to consider direct and indirect transition rules for all derived predicates while indirect rules do not solely depend on other indirect rules anymore but may also contain references to direct transition rules as long as the entire rule set remains stratifiable. Hence, the new state of a derived relation is defined by two different kinds of transition rules, each of them dedicated to a specific task. Direct transition rules are used for computing induced insertions while indirect rules are employed for computing induced deletions. Transition rules of this structure preserve stratifiability and will be denoted *incremental transition rules* in the following.

Definition 5.6 (Incremental Transition Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a stratifiable deductive database. Then the set of incremental transition rules for true updates and new state simulation with respect to $\mathcal{R}$ is denoted $\tau_i(\mathcal{R})$ and is defined as follows:*

1. *For each n-ary predicate symbol $p \in \texttt{pred}(\mathcal{F} \cup \mathcal{R})$, the direct transition rules*

   $$\texttt{new}^d\ A \leftarrow \texttt{old}\ A \wedge \neg A^-$$
   $$\texttt{new}^d\ A \leftarrow A^+$$

   are in $\tau_i(\mathcal{R})$ where $A \equiv p(x_1, \ldots, x_n)$, and the x_i $(i = 1, \ldots, n)$ are distinct variables.

2. *For each rule $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}$, an indirect transition rule of the form*

   $$\texttt{new}^i\ A \leftarrow \nu_1 L_1 \wedge \ldots \wedge \nu_n L_n$$

   is in $\tau_i(\mathcal{R})$ where

   $$\nu_i := \begin{cases} \texttt{new}^i & \text{if } \texttt{pred}(L_i) \approx \texttt{pred}(A) \\ \texttt{new}^d & \text{otherwise} \end{cases}$$

   for $i = 1, \ldots, n$.

3. No other rules are in $\tau_i(\mathcal{R})$.

The $\texttt{new}^i$ state relation will be applied in the effectiveness test of negative propagation rules, and the $\texttt{new}^d$ state relations in the derivability test of negative propagation rules. In positive propagation rules the $\texttt{new}^d$ state relations are used in both, the derivability and effectiveness test. Under this assumption the following proposition holds:

Proposition 5.7 (Correctness of Incremental Transition Rules) *Let* $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ *be a stratifiable database,* $u_\mathcal{D}$ *an update and* $u_{D \to D'} = \langle u^+_{D \to D'}, u^-_{D \to D'} \rangle$ *the corresponding induced update from* $\mathcal{D}$ *to* $\mathcal{D}'$. *Let* $\mathcal{D}^p = \langle \mathcal{F} \cup \texttt{prop\_seeds}(u_\mathcal{D}), \mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau_i(\mathcal{R}) \rangle$ *be the augmented deductive database of* $\mathcal{D}$ *and the mapping* new *used in* $\varphi(\mathcal{R})$ *defined by*

$$\texttt{new}\ L := \begin{cases} \texttt{new}^i L & \text{if } \texttt{new}\ L \text{ occurs in the effectiveness test of a negative propagation rule} \\ \texttt{new}^d L & \text{otherwise.} \end{cases}$$

Then $\mathcal{D}^p$ *correctly represents the implicit state of* $\mathcal{D}'$, *i.e., for all atoms* $A \in \mathcal{H}_{D'}$

$$A \in \mathcal{M}_{D'} \iff \texttt{new}\ A \in \mathcal{M}_{\mathcal{D}^p},$$

and all delta relations defined by the propagation rules $\varphi(\mathcal{R})$ *correctly represent the induced update* $u_{D \to D'}$, *i.e., for* $A \equiv p(\vec{t}\,)$:

$$\begin{aligned} \Delta^+ p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^p} &\iff p(\vec{t}\,) \in u^+_{D \to D'} \\ \Delta^- p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^p} &\iff p(\vec{t}\,) \in u^-_{D \to D'}\ . \end{aligned}$$

Proof: cf. [Gri97, p. 179-180]. □

For illustrating the definitions above, consider again the deductive rules from Example 5.1 for defining the relations `path/2` and `one_way/2`. Let the mappings $\texttt{new}^i$, $\texttt{new}^d$ and `old` for a literal $A \equiv r(t_1, \ldots, t_n)$ be defined as follows:

$$\begin{aligned} \texttt{new}^i A &:= r^{new^i}(t_1, \ldots, t_n) & \texttt{new}^i \neg A &:= \neg \texttt{new}^i A \\ \texttt{new}^d A &:= r^{new^d}(t_1, \ldots, t_n) & \texttt{new}^d \neg A &:= \neg \texttt{new}^d A \\ \texttt{old}\ A &:= A & \texttt{old} \neg A &:= \neg \texttt{old}\ A \end{aligned}$$

The corresponding propagation rules $\varphi(\mathcal{R})$ will be

1. $\Delta^{+}\mathtt{o}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{Y}) \wedge \neg\, \mathtt{p}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{Y},\mathtt{X}) \wedge \neg\mathtt{o}(\mathtt{X})$
   $\Delta^{+}\mathtt{o}(\mathtt{X}) \leftarrow \Delta^{-}\mathtt{p}(\mathtt{Y},\mathtt{X}) \wedge \mathtt{p}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{o}(\mathtt{X})$
   $\Delta^{-}\mathtt{o}(\mathtt{X}) \leftarrow \Delta^{-}\mathtt{p}(\mathtt{X},\mathtt{Y}) \wedge \neg\, \mathtt{p}(\mathtt{Y},\mathtt{X}) \wedge \neg\mathtt{o}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{X})$
   $\Delta^{-}\mathtt{o}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{p}(\mathtt{Y},\mathtt{X}) \wedge \mathtt{p}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{o}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{X})$

2. $\Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{p}(\mathtt{X},\mathtt{Y})$
   $\Delta^{-}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{-}\mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{p}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{X},\mathtt{Y})$

3. $\Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{Z},\mathtt{Y}) \wedge \neg\mathtt{p}(\mathtt{X},\mathtt{Y})$
   $\Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{p}(\mathtt{Z},\mathtt{Y}) \wedge \mathtt{e}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Z}) \wedge \neg\mathtt{p}(\mathtt{X},\mathtt{Y})$
   $\Delta^{-}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{-}\mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}(\mathtt{Z},\mathtt{Y}) \wedge \neg\mathtt{p}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{X},\mathtt{Y})$
   $\Delta^{-}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{-}\mathtt{p}(\mathtt{Z},\mathtt{Y}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \neg\mathtt{p}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{X},\mathtt{Y})$

while the incremental transition rules $\tau_i(\mathcal{R})$ are given by

1. $\mathtt{o}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{X}) \leftarrow \mathtt{p}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Y}) \wedge \neg\, \mathtt{p}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{Y},\mathtt{X})$

   $\mathtt{o}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X}) \leftarrow \mathtt{o}(\mathtt{X}) \wedge \neg\, \Delta^{-}\mathtt{o}(\mathtt{X})$
   $\mathtt{o}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{o}(\mathtt{X})$

2. $\mathtt{p}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Y})$
   $\mathtt{p}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}^{\mathtt{new}^{\mathtt{i}}}(\mathtt{Z},\mathtt{Y})$

   $\mathtt{p}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{p}(\mathtt{X},\mathtt{Y}) \wedge \neg\, \Delta^{-}\mathtt{p}(\mathtt{X},\mathtt{Y})$
   $\mathtt{p}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{Y})$

3. $\mathtt{e}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \neg\, \Delta^{-}\mathtt{e}(\mathtt{X},\mathtt{Y})$
   $\mathtt{e}^{\mathtt{new}^{\mathtt{d}}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y}).$

Similar to propagation rules, transition rules can be determined at schema definition time as well and don't have to be recompiled each time a new update is applied. Since we work on the old database state, the mapping old(A) simply yields A. The effectiveness test makes sure that only true updates are computed by the propagation rules; that is, only insertions are derived with respect to facts which were not derivable in the old database state while only those deletions are derived with respect to facts which were derivable in the old database state. Although the application of delta literals indeed restricts the computation of induced updates, the side literals and effectiveness test within the propagation rules as well as the transition rules of this example require the entire new and old state of relation e, p and o to be derived. In addition, when employing incremental transition rules, the situation becomes even worse as the simulated new state of a relation has to be materialized twice (via $\mathtt{new}^{\mathtt{d}}$ and $\mathtt{new}^{\mathtt{i}}$) if evaluated using a

pure bottom-up approach.

In order to avoid this drawback, in [Gri97] the evaluation of transition rules is limited by using the Magic Set method. The reason for the application of Magic Sets is twofold: On the one hand, this method is used to restrict the evaluation to those old and new state facts only which are really needed for satisfying the effectiveness and derivability test of a propagated update. On the other hand, if incremental transition rules are used, the top-down evaluation simulated by Magic Sets basically divides the derivation of relevant new state facts into those facts derivable using direct transition rules and those derivable by indirect transition rules. Thus, despite of using two kinds of transition rules, no new state fact is redundantly derived twice by direct and indirect transition rules.

In the following section we will show how the Magic Set rewriting can be used for enhancing the update propagation rules introduced above. Since the considered update propagation rules are stratifiable, the resulting Magic Updates rules must be softly stratifiable and thus, may be evaluated using the soft stratification method from Chapter 4. This approach for computing true updates will be denoted *soft update propagation*.

5.2 Update Propagation via Soft Stratification

In Section 5.1 we already pointed out the obvious inefficiency of update propagation, if performed by a pure bottom-up materialization of the augmented database. In fact, simply applying iterated fixpoint computation to an augmented database as proposed in Section 3.2 implies that at least all derived relations which are relevant for showing the effectiveness of derived delta facts in propagation and transition rules will be entirely computed. Thus, the implicit state of both, the old and new database state of these relations will be materialized, although in most cases only a small portion of each are relevant for computing the induced changes. The only benefit of incremental propagation rules is that the evaluation of their bodies is restricted to the values of the currently propagated updates and thus can be completely avoided if delta relations are empty. In a pure top-down procedure, on the other hand, the values of the propagated updates can be passed to the side literals and effectiveness tests automatically restricting their evaluation to the relevant part of the database. However, a pure top-down approach must query all existing delta relations in order to check whether they are affected by an induced update, although for most of them this will not be the case.

In this section we develop an update propagation approach which combines the advantages of the two strategies discussed above. In this way, update propagation is automatically limited to the affected delta relations and the evaluation of side literals and effectiveness tests is restricted to the updates currently propagated. We will use the Magic Sets approach for incorporating a top-down evaluation

strategy by considering the currently propagated updates in the dynamic body literals as abstract queries on the remainder of the respective propagation rule bodies. Evaluating these queries (in the following called *propagation queries*) has the advantage that the respective state relations will only be partially materialized. Moreover, later evaluations of propagation queries can benefit from the state facts already derived in previous iteration rounds.

5.2.1 Soft Update Propagation by Example

Before formally presenting the soft update propagation approach, we will illustrate the main ideas by means of an example.

Example 5.2 *Let us consider the following stratifiable deductive database* $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ *with* $\mathcal{R}$ *consisting again of the well-known transitive closure rules for defining the derived relation* `path/2`*:*

$\underline{\mathcal{R}:}$

$$\mathtt{p(X,Y)} \leftarrow \mathtt{e(X,Y)}$$
$$\mathtt{p(X,Y)} \leftarrow \mathtt{e(X,Z)} \wedge \mathtt{p(Z,Y)}$$

$\underline{\mathcal{F}:}$

```
e(1,2), e(1,4), e(3,4)
e(10,11), e(11,12), ..., e(98,99), e(99,10),e(99,100)
```

The positive portion $\mathcal{M}^+_{\mathcal{D}^m}$ *of the corresponding total well-founded model* $\mathcal{M}_{\mathcal{D}^m} = \mathcal{M}^+_{\mathcal{D}^m} \uplus \neg \cdot \overline{\mathcal{M}^+_{\mathcal{D}^m}}$ *consists of* 8193 *p-facts, i.e.,* $|\mathcal{M}^+_{\mathcal{D}^m}| = 8193 + |e| = 8287$ *facts.*

For maintaining readability we restrict our attention to the propagation of true insertions. In addition, we assume the new state to be simulated by means of naive transition rules although incremental transition rules could be applied as well. Let the mappings `new` and `old` for a literal $A \equiv r(t_1, \ldots, t_n)$ be defined as `new` $A := r^{new}(t_1, \ldots, t_n)$ and `old` $A := A$. The corresponding propagation rules $\varphi(\mathcal{R})$ are then given by

$$\Delta^+\mathtt{p(X,Y)} \leftarrow \Delta^+\mathtt{e(X,Y)} \wedge \neg\mathtt{p(X,Y)}$$
$$\Delta^+\mathtt{p(X,Y)} \leftarrow \Delta^+\mathtt{e(X,Z)} \wedge \mathtt{p}^{\mathtt{new}}\mathtt{(Z,Y)} \wedge \neg\mathtt{p(X,Y)}$$
$$\Delta^+\mathtt{p(X,Y)} \leftarrow \Delta^+\mathtt{p(Z,Y)} \wedge \mathtt{e}^{\mathtt{new}}\mathtt{(X,Z)} \wedge \neg\mathtt{p(X,Y)}$$

while the naive transition rules $\tau_n(\mathcal{R})$ are

$$\mathtt{p}^{\mathtt{new}}\mathtt{(X,Y)} \leftarrow \mathtt{e}^{\mathtt{new}}\mathtt{(X,Y)}$$
$$\mathtt{p}^{\mathtt{new}}\mathtt{(X,Y)} \leftarrow \mathtt{e}^{\mathtt{new}}\mathtt{(X,Z)} \wedge \mathtt{p}^{\mathtt{new}}\mathtt{(Z,Y)}$$

$$\mathtt{e}^{\mathtt{new}}\mathtt{(X,Y)} \leftarrow \mathtt{e(X,Y)} \wedge \neg\Delta^-\mathtt{e(X,Y)}$$
$$\mathtt{e}^{\mathtt{new}}\mathtt{(X,Y)} \leftarrow \Delta^+\mathtt{e(X,Y)}.$$

Let $u_{\mathcal{D}}$ be an update consisting of the new edge fact $e(2,3)$ to be inserted into $\mathcal{D}$, i.e., $u_{\mathcal{D}}^{+} = \{e(2,3)\}$. The resulting augmented database $\mathcal{D}^p$ is $\mathcal{D}^p = \langle \mathcal{F} \cup \{\Delta^{+}\mathtt{e}(2,3)\}, \mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau_n(\mathcal{R})\rangle$. Computing the induced update by evaluating the stratifiable database $\mathcal{D}^p$ leads to the generation of 95 new state facts for relation e, 8193 old state facts for p and $8193+3$ new state facts for p. The entire number of generated facts is 16487 for computing the three induced insertions $\Delta^{+}\mathtt{p}(1,3)$, $\Delta^{+}\mathtt{p}(2,3)$, and $\Delta^{+}\mathtt{p}(2,4)$ with respect to relation p.

We will now apply the Magic Sets rewriting with respect to the abstract (propagation) queries Q^u represented by the predicates $\Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y})$, $\Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Z})$ and $\Delta^{+}\mathtt{p}(\mathtt{Z},\mathtt{Y})$ in the propagation rule bodies. In order to emphasize the analogy to the Magic Sets approach this transformation is denoted *Magic Updates rewriting*. Let $\mathcal{R}^p = \mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau_n(\mathcal{R})$ be the set of update rules used in the augmented database $\mathcal{D}^p$ and $\mathcal{R}^p_{Q^u}$ the adorned rule set of $\mathcal{R}^p$ with respect to the abstract propagation queries Q^u. The rule set resulting from the application of the Magic Updates rewriting will be denoted $\mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{Q^u}})$ and consists of the following answer rules for our example

$$\Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \neg\, \mathtt{p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y})$$
$$\Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z},\mathtt{Y}) \wedge \neg\, \mathtt{p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y})$$
$$\Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{p}(\mathtt{Z},\mathtt{Y}) \wedge \mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{X},\mathtt{Z}) \wedge \neg\, \mathtt{p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y})$$

$$\mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X}) \wedge \mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y})$$
$$\mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X}) \wedge \mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z},\mathtt{Y})$$

$$\mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \neg\, \Delta^{-}\mathtt{e}(\mathtt{X},\mathtt{Y})$$
$$\mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X}) \wedge \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y})$$
$$\mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{Y}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \neg\, \Delta^{-}\mathtt{e}(\mathtt{X},\mathtt{Y})$$
$$\mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{Y}) \wedge \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y})$$

$$\mathtt{p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Y})$$
$$\mathtt{p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}_{\mathtt{bb}}(\mathtt{Z},\mathtt{Y})$$

as well as the following sub-query rules

$$\mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Z})$$
$$\mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z}) \leftarrow \mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X}) \wedge \mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Z})$$

$$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{Z}) \leftarrow \Delta^{+}\mathtt{p}(\mathtt{Z},\mathtt{Y})$$

$$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X}) \leftarrow \mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X})$$

$\mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}$	$\mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}$	$\mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}$	$\mathtt{p}_{\mathtt{bb}}$	$\mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}$	$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{bf}}$	$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{fb}}$	$\mathtt{m\_p}_{\mathtt{bb}}$
$\mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}(3,4)$	$\mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(3,4)$	$\mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}(1,2)$	$\mathtt{p}_{\mathtt{bb}}(1,4)$	$\mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(3)$	$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{bf}}(3)$	$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{fb}}(1)$	$\mathtt{m\_p}_{\mathtt{bb}}(1,3)$
				$\mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(4)$	$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{bf}}(4)$	$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{fb}}(2)$	$\mathtt{m\_p}_{\mathtt{bb}}(1,4)$
							$\mathtt{m\_p}_{\mathtt{bb}}(2,3)$
							$\mathtt{m\_p}_{\mathtt{bb}}(2,4)$
							$\mathtt{m\_p}_{\mathtt{bb}}(4,3)$
							$\mathtt{m\_p}_{\mathtt{bb}}(4,4)$

Table 5.1: Generated state relation facts using soft update propagation

$$\mathtt{m\_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y})$$
$$\mathtt{m\_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z},\mathtt{Y})$$
$$\mathtt{m\_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{p}(\mathtt{Z},\mathtt{Y}) \wedge \mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{X},\mathtt{Z})$$
$$\mathtt{m\_p}_{\mathtt{bb}}(\mathtt{Z},\mathtt{Y}) \leftarrow \mathtt{m\_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Z}).$$

Quite similar to the Magic sets approach, the Magic Updates rewriting may result in an unstratifiable rule set. This is also the case for the rule set of our example where the following negative cycle can be found in the corresponding dependency graph of $\mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{Q}^{\mathtt{u}}})$:

$$\Delta^{+}\mathtt{p} \xrightarrow{pos} \mathtt{m\_p}_{\mathtt{bb}} \xrightarrow{pos} \mathtt{p}_{\mathtt{bb}} \xrightarrow{neg} \Delta^{+}\mathtt{p}$$

We will show, however, that the resulting rule set must be at least softly stratifiable such that the soft consequence operator could be used for determining the corresponding well-founded model. Computing the induced update by evaluating $\mathcal{D}^{mp} = \langle \mathcal{F} \cup \{\Delta^{+}\mathtt{e}(2,3)\}, \mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{Q}^{\mathtt{u}}})\rangle$ leads to the generation of two new state facts for relation e, one old state fact and one new state fact for p. The entire number of generated facts is 19 in contrast to 16487 for computing the three induced insertions with respect to relation p. Table 5.1 summarizes the generated state relation facts with respect to the corresponding answer and sub-query rules in $\mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{Q}^{\mathtt{u}}})$. The reason for the small number of facts is that only relevant state relation facts are derived. In the example above, this excludes the set of edge facts $\{\mathtt{e}(10,11), \mathtt{e}(11,12), \ldots, \mathtt{e}(98,99), \mathtt{e}(99,10), \mathtt{e}(99,100)\}$ and the corresponding p-facts as they are not affected by the insertion $\Delta^{+}\mathtt{e}(2,3)$ and thus, do not have to be considered during the update propagation process.

Although this example already shows the advantages of applying the Magic Sets transformation to the propagation rules from Section 5.1, the application of a rule set resulting from the Magic Updates rewriting does not necessarily improve the performance of the update propagation process. This is due to the fact that there are cases where the relevant part of a database represented by Magic Sets transformed rules together with the necessary sub-queries exceeds the amount of derivable facts using the original rule set. However, these cases are 'theoretically'

constructed examples. In general, the Magic Sets approach indeed leads to a well-optimized rule evaluation, and so does the Magic Updates approach.

Note that we have covered so far the application of naive transition rules only, although incremental transition rules could have been used in the sample database as well. The advantage of incremental transition rules is that at least the computation of induced insertions is partially based on previously derived induced deletions and insertions by using direct transition rules (via $\texttt{new}^{\texttt{d}}$). Induced deletions, however, are based on indirect transition rules (via $\texttt{new}^{\texttt{i}}$) which almost correspond to naive transition rules and thus, make no use of induced updates already computed in previous iteration rounds. The application of the Magic Sets rewriting now restricts the new state relations defined by indirect and direct transition rules to those portions which are relevant for computing induced deletions and induced insertions, respectively. The only remaining disadvantage when applying incremental transition rules then is the consideration of two different relations $\texttt{new}^{\texttt{i}}$ and $\texttt{new}^{\texttt{d}}$ for simulating the new state of a relation which are not necessarily disjoint in spite of using Magic Sets. Thus, additional joins have to be performed and identical new state facts could be derived by the two rule sets, separately.

We will now formally introduce the Magic Updates rewriting and prove it to be always softly stratifiable. Afterwards we present a comparison to the related structured update propagation approach by Griefahn in [Gri97] and argue that soft stratification indeed represents an efficient update propagation method for stratifiable deductive databases.

5.2.2 The Soft Update Propagation Approach

In this section we formally introduce the soft update propagation approach. To this end, we define the Magic Updates rewriting which, applied to an augmented rule set, results in a set of propagation rules that contains references to relevant portions of state relations only. After proving its correctness, it is shown that the resulting rule set is softly stratifiable and that its evaluation using the soft consequence operator from Section 3.2.2 yields the induced updates defined by the underlying augmented database.

Definition 5.8 (Magic Updates Rewriting) *Let $\mathcal{R}$ be a stratifiable rule set, $\mathcal{R}^p = \mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau(\mathcal{R})$ an augmented rule set of $\mathcal{R}$, and Q^u the set of abstract propagation queries given by all delta literals occurring in rule bodies of propagation rules in $\varphi(\mathcal{R})$. The Magic Updates rewriting of $\mathcal{R}^p$ yields the magic rule set $\texttt{mu}(\mathcal{R}^p_{Q^u}) := \mathcal{R}^u_P \cup \mathcal{R}^u_Q \cup \mathcal{R}^u_M$ where $\mathcal{R}^u_P$, $\mathcal{R}^u_Q$ and $\mathcal{R}^u_M$ are defined as follows:*

1. *From $\varphi(\mathcal{R})$ we derive the two deductive rule sets $\mathcal{R}^u_P$ and $\mathcal{R}^u_Q$: For each propagation rule $A^\pi \leftarrow \Delta^{\dot{\pi}}\texttt{e} \wedge L^1 \wedge \ldots \wedge L^n \in \varphi(\mathcal{R})$ with $\Delta^{\dot{\pi}}\texttt{e} \in Q^u$ is a dynamic literal and $\pi, \dot{\pi} \in \{+, -\}$, an adorned answer rule of the form*

$$A^{\pi} \leftarrow \Delta^{\vec{\pi}}\mathtt{e} \wedge L^1_{ad_1} \wedge \ldots \wedge L^n_{ad_n}$$

is in $\mathcal{R}^u_P$ *where each non-dynamic body literal* L^i $(1 \leq i \leq n)$ *is replaced by the corresponding adorned literal* $L^i_{ad_i}$ *while assuming the body literals* $\Delta^{\vec{\pi}}\mathtt{e} \wedge L^1 \wedge \ldots \wedge L^{i-1}$ *have been evaluated in advance. Note that the adornment of each non-derived literal consists of the empty string. For each derived adorned body literal* $L^i_{ad_i}$ $(1 \leq i \leq n)$ *a sub-query rule of the form*

$$\mathtt{magic}(L^i_{ad_i}) \leftarrow \Delta^{\vec{\pi}}\mathtt{e} \wedge L^1_{ad_1} \wedge \ldots \wedge L^{i-1}_{ad_{i-1}}$$

is in $\mathcal{R}^u_Q$.

No other rules are in $\mathcal{R}^u_P$ *and* $\mathcal{R}^u_Q$.

2. *From the set* $\mathcal{R}_{state} := \mathcal{R} \uplus \tau(\mathcal{R})$ *we derive the rule set* $\mathcal{R}^u_M$*: For each relation symbol* $\mathtt{magic}(L_{ad}) \in \mathtt{pred}(\mathcal{R}^u_Q)$ *the corresponding Magic Set transformed rule set* $\mathtt{ms}(\mathcal{R}^Q_{state})$ *is in* $\mathcal{R}^u_M$ *where* $Q \equiv L_{ad}$ *represents an adorned query with* $\mathtt{pred}(L) \in \mathtt{pred}(\mathcal{R}_{state})$ *and* $\mathcal{R}^Q_{state}$ *is the adorned rule set of* $\mathcal{R}_{state}$ *with respect to* Q.

3. *No other rules are in* $\mathcal{R}^u_M$.

The following Theorem 5.1 shows that a rule set resulting from the Magic Updates rewriting is always softly stratifiable and correctly represents the induced updates defined by the underlying augmented database.

Theorem 5.1 *Let* $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ *be a stratifiable database,* $u_{\mathcal{D}}$ *an update,* $u_{D \to D'} = \langle u^+_{D \to D'}, u^-_{D \to D'} \rangle$ *the corresponding induced update from* $\mathcal{D}$ *to* $\mathcal{D}'$*,* Q^u *the set of all abstract queries in* $\varphi(\mathcal{R})$*, and* $\mathcal{R}^p = \mathcal{R} \cup \varphi(\mathcal{R}) \cup \tau(\mathcal{R})$ *an augmented rule set of* $\mathcal{R}$*. Let* $\mathtt{mu}(\mathcal{R}^p_{Q^u})$ *be the result of applying Magic Updates rewriting to* $\mathcal{R}^p$ *and* $\mathcal{D}^{mp} = \langle \mathcal{F} \cup \mathtt{prop\_seeds}(u_{\mathcal{D}}), \mathtt{mu}(\mathcal{R}^p_{Q^u}) \rangle$ *the corresponding augmented deductive database of* $\mathcal{D}$*. Then* $\mathcal{D}^{mp}$ *is softly stratifiable and all delta relations in* $\mathcal{D}^{mp}$ *defined by the propagation rules* $\varphi(\mathcal{R})$ *correctly represent the induced update* $u_{D \to D'}$*, i.e., for all atoms* $A \in \mathcal{H}_{D'}$ *with* $A \equiv p(\vec{t}\,)$*:*

$$\Delta^+ p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^{mp}} \iff p(\vec{t}\,) \in u^+_{D \to D'}$$
$$\Delta^- p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^{mp}} \iff p(\vec{t}\,) \in u^-_{D \to D'} \ .$$

Proof: The correctness of the Magic Updates rewriting with respect to an augmented rule set $\mathcal{R}^p$ is shown by proving it to be equivalent to a specific Magic Set transformation of $\mathcal{R}^p$ which is known to be sound and complete. In Chapter 4 the Magic Sets transformation has been introduced by starting with the adornment

phase which basically depicts information flow between literals in a database according to a chosen sip strategy. Up till now, we have considered full sip strategies only in which all captured variable bindings are passed to the literal considered next. In [BPRM91] it is shown, however, that the Magic Sets approach is also sound for so-called *partial sip strategies* which may pass on only a certain subset of captured variable bindings, or even no bindings at all. Let us assume we have chosen such a sip strategy which passes no bindings to dynamic literals such that their adornments are strings solely consisting of $'f'$ symbols representing unbounded attributes. Additionally, let $\mathcal{R}_{\acute{p}} = \mathcal{R}^p \cup \{h \leftarrow \Delta^{\pi 1} p1(\vec{x}_1)\} \cup \ldots \cup \{h \leftarrow \Delta^{\pi n} pn(\vec{x}_n)\}$ be an extended augmented rule set with rules for defining an auxiliary 0-ary relation h with $h \notin \mathtt{pred}(\varphi(\mathcal{R}))$, $\{\Delta^{\pi 1} p1, \ldots, \Delta^{\pi n} pn\} = \mathtt{pred}(\varphi(\mathcal{R}))$ distinct predicates, and $\vec{x}_i$ $(i = 1, \ldots, n)$ vectors of pairwise distinct variables with a length according to the arity of the corresponding predicates $\Delta^{\pi i} pi$. Relation h references all derived delta relations in $\varphi(\mathcal{R})$ as they are potentially affected by a given base update. Note that since $\mathcal{R}^p$ is assumed to be stratifiable, $\mathcal{R}_{\acute{p}}$ must be stratifiable as well. The Magic Sets rewriting of $\mathcal{R}_{\acute{p}}$ with respect to the query $H \equiv h$ using a partial sip strategy as proposed above yields the rule set $\mathtt{ms}(\mathcal{R}^H_{\acute{p}})$ which is basically equivalent to the rule set $\mathtt{mu}(\mathcal{R}^p_{Q^u})$ resulting from the Magic Updates rewriting. The rule set $\mathtt{ms}(\mathcal{R}^H_{\acute{p}})$ differs from $\mathtt{mu}(\mathcal{R}^p_{Q^u})$ by the answer rules of the form

$$h \leftarrow m\_h, \Delta^{\pi 1} p1_{ff\ldots}(\vec{x}_1), \ \ldots \ , h \leftarrow m\_h, \Delta^{\pi n} pn_{ff\ldots}(\vec{x}_n)$$

for the additional relation h, by sub-query rules of the form

$$m\_\Delta^{\pi 1} p1_{ff\ldots} \leftarrow m\_h, \ \ldots \ , m\_\Delta^{\pi n} pn_{ff\ldots} \leftarrow m\_h,$$

by sub-query rules of the form

$$m\_\Delta^{\pi i} pi_{ff\ldots} \leftarrow m\_\Delta^{\pi j} pj_{ff\ldots} \text{ with } i, j \in \{1, \ldots, n\},$$

and by the usage of $m\_\Delta^{\pi i} pi_{ff\ldots}$ literals in propagation rule bodies for defining a corresponding delta relation $\Delta^{\pi i} pi_{ff\ldots}$. It is obvious that these rules and literals can be removed from $\mathtt{ms}(\mathcal{R}^H_{\acute{p}})$ without changing the semantics of the derived delta relations in $\mathtt{ms}(\mathcal{R}^H_{\acute{p}})$. The remaining rules coincide with the magic updates rules $\mathtt{mu}(\mathcal{R}^p_{Q^u})$. Using Propositions 5.3, 5.5 and 5.7, it can be concluded that the rule set $\mathcal{R}^H_{\acute{p}}$ is stratifiable, and all delta relations defined in it correctly represent the induced update $u_{D \rightarrow D'}$. Thus, the Magic Set transformed rule set $\mathtt{ms}(\mathcal{R}^H_{\acute{p}})$ must be sound and complete as well. As the magic updates rules $\mathtt{mu}(\mathcal{R}^p_{Q^u})$ can be derived from $\mathtt{ms}(\mathcal{R}^H_{\acute{p}})$ in the way described above, they must correctly represent the induced update $u_{D \rightarrow D'}$ as well. In addition, since $\mathtt{ms}(\mathcal{R}^H_{\acute{p}})$ is softly stratifiable, the magic updates rules $\mathtt{mu}(\mathcal{R}^p_{Q^u})$ must be softly stratifiable, too. □

From Theorem 5.1 follows that the soft stratification approach from Section 4.2.2 can be applied for efficiently computing the induced changes represented by the augmented database $\mathcal{D}^{mp}$. For instance, the partition $\mathcal{P} := P_1 \cup P_2$ of the Magic Updates transformed rule set $\mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{Q^u}})$ of our running example with

$\underline{P_1}$:

$$
\begin{array}{ll}
\mathtt{p_{bb}(X,Y)} & \leftarrow \mathtt{m\_p_{bb}(X,Y)} \wedge\ \mathtt{e(X,Y)} \\
\mathtt{p_{bb}(X,Y)} & \leftarrow \mathtt{m\_p_{bb}(X,Y)} \wedge\ \mathtt{e(X,Z)}\ \ \wedge \mathtt{p_{bb}(Z,Y)}
\end{array}
$$

$$
\begin{array}{l}
\mathtt{m\_p_{bb}(X,Y)} \leftarrow \Delta^{+}\mathtt{e(X,Y)} \\
\mathtt{m\_p_{bb}(X,Y)} \leftarrow \Delta^{+}\mathtt{p(Z,Y)} \wedge\ \mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}\mathtt{(X,Z)} \\
\mathtt{m\_p_{bb}(X,Y)} \leftarrow \Delta^{+}\mathtt{e(X,Z)} \wedge\ \mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}\mathtt{(Z,Y)} \\
\mathtt{m\_p_{bb}(Z,Y)} \leftarrow \mathtt{m\_p_{bb}(X,Y)} \wedge\ \mathtt{e(X,Z)}.
\end{array}
$$

and with partition P_2 consisting of all other magic updates rules, i.e., $P_2 := \mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{Q^u}}) \setminus P_1$, satisfies the condition of soft stratification. Using the soft consequence operator for determining $\mathtt{lfp}(T^s_{\mathcal{P}}, \mathcal{F} \cup \{\Delta^{+}\mathtt{e(2,3)}\})$ yields the correct well-founded model, the state relation facts of which were already presented in Table 5.1.

Before we compare soft update propagation with the related structured update propagation approach by Griefahn [Gri97], we briefly consider again the problem of optimizing existential queries from Section 4.3 for improving the evaluation of the effectiveness test in our Magic Updates transformed propagation rules.

5.2.3 Efficient Evaluation of the Effectiveness Test

In Section 4.3 we presented an approach for optimizing (derived) existential queries in a Magic Sets transformed rule set. The basic idea was to apply the existential magic sets rewriting to certain existentially queried relations instead of the original Magic Sets transformation. As this rewriting may lead to repeated computations of facts, we proposed to solely optimize existential derived queries with respect to recursively defined relations or negatively referenced relations.

In principle, the efficient evaluation of (derived) existential queries represents an orthogonal optimization problem which can be considered after the Magic Updates rewriting has been applied. However, because of the special structure of magic updates rules, this optimization technique turns out to be very important in this context. Therefore, we will briefly discuss the optimization effects that can be achieved when applying the existential magic sets rewriting with respect to the effectiveness tests in propagation rules.

Let us consider again the transitive closure rules as well as the fact base from Example 5.2 in Section 5.2.1 and let $u_{\mathcal{D}}$ be an update consisting of the new edge fact $e(98,100)$ to be inserted into $\mathcal{D}$, i.e., $u^{+}_{\mathcal{D}} = \{e(98,100)\}$. Because of the facts $e(98,99)$ and $e(99,100)$ in $\mathcal{F}$ it is known that no additional p-facts are derivable after applying this update. The augmented database $\mathcal{D}^{mp}$ resulting from the magic updates rewriting is given by $\mathcal{D}^{mp} = \langle \mathcal{F} \cup \{\Delta^{+}\mathtt{e(98,100)}\}, \mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{Q^u}})\rangle$. Computing the induced update by evaluating the softly stratifiable database $\mathcal{D}^{mp}$ leads to the generation of 91 sub-query facts with respect to m_p_{bb} and 90 old

state facts with respect to p despite of using magic sets. The entire number of generated answer and sub-query facts is 183 for showing that the insertion $u^+_{\mathcal{D}} = \{e(98, 100)\}$ does not affect relation p.

Originally, we used the Magic Sets approach to limit the evaluation of side literals and effectiveness tests in propagation rules to the relevant part of the database. However, it is already clear that one successful derivation with respect to the negative literals of the effectiveness test is sufficient to show that a potential update is ineffective. For our example this implies that after computing the sub-query facts $\mathtt{m\_p_{bb}}(\mathtt{98}, \mathtt{100})$ and $\mathtt{m\_p_{bb}}(\mathtt{99}, \mathtt{100})$ together with the corresponding answer facts $\mathtt{p_{bb}}(\mathtt{99}, \mathtt{100})$ and $\mathtt{p_{bb}}(\mathtt{98}, \mathtt{100})$ the evaluation could have been stopped. Therefore, we propose to apply the existential magic sets rewriting with respect to the effectiveness tests leading to the following modified answer rules R_a in our running example:

$$\mathtt{p_{bb}}(\mathtt{X}, \mathtt{Y}) \leftarrow \mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}, \mathtt{U}, \mathtt{V}) \wedge \mathtt{e}(\mathtt{X}, \mathtt{Y})$$
$$\mathtt{p_{bb}}(\mathtt{X}, \mathtt{Y}) \leftarrow \mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}, \mathtt{U}, \mathtt{V}) \wedge \mathtt{e}(\mathtt{X}, \mathtt{Z}) \wedge \mathtt{p_{bb}}(\mathtt{Z}, \mathtt{Y}).$$

In addition, the following modified sub-query rules R_m are generated:

$$\mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}, \mathtt{X}, \mathtt{Y}) \leftarrow \Delta^+\mathtt{e}(\mathtt{X}, \mathtt{Y})$$
$$\mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}, \mathtt{X}, \mathtt{Y}) \leftarrow \Delta^+\mathtt{p}(\mathtt{Z}, \mathtt{Y}) \wedge \mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{X}, \mathtt{Z})$$
$$\mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}, \mathtt{X}, \mathtt{Y}) \leftarrow \Delta^+\mathtt{e}(\mathtt{X}, \mathtt{Z}) \wedge \mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z}, \mathtt{Y})$$
$$\mathtt{m\_p_{bb}}(\mathtt{Z}, \mathtt{Y}, \mathtt{U}, \mathtt{V}) \leftarrow \mathtt{m\_p_{bb}}(\mathtt{X}, \mathtt{Y}, \mathtt{U}, \mathtt{V}) \wedge \mathtt{e}(\mathtt{X}, \mathtt{Z}) \wedge \neg\mathtt{p}(\mathtt{U}, \mathtt{V}).$$

All other rules in $\mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{q^u}})$ remain unchanged. The partition $\mathcal{P} := P_1 \uplus P_2 \uplus P_3$ with $P_1 := R_a$, $P_2 := R_m$, and P_3 consisting of all other rules in $\mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{q^u}})$ satisfies the condition of soft stratification and additionally separates answer and sub-query rules with respect to relation $\mathtt{p_{bb}}$ as necessary for existential query optimization. Using the soft consequence operator for evaluating the soft partition $\mathcal{P}$ then induces the following sequence of facts:

$$\begin{aligned}
F_1 &:= \mathcal{F} \cup \{\Delta^+\mathtt{e}(\mathtt{98}, \mathtt{100})\} \\
F_2 &:= T^\star_{P_2}(F_1) = F_1 \cup \{\mathtt{m_p_{bb}}(\mathtt{98}, \mathtt{100}, \mathtt{98}, \mathtt{100})\} \\
F_3 &:= T^\star_{P_2}(F_2) = F_2 \cup \{\mathtt{m_p_{bb}}(\mathtt{99}, \mathtt{100}, \mathtt{98}, \mathtt{100})\} \\
F_4 &:= T^\star_{P_1}(F_3) = F_3 \cup \{\mathtt{p_{bb}}(\mathtt{99}, \mathtt{100})\} \\
F_5 &:= T^\star_{P_1}(F_4) = F_4 \cup \{\mathtt{p_{bb}}(\mathtt{98}, \mathtt{100})\} \\
F_6 &:= T^\star_{P_3}(F_5) = F_5 \cup \{\mathtt{m_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{100})\} \\
F_7 &:= T^\star_{P_3}(F_6) = F_6 \cup \{\mathtt{m_e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{100})\} \\
F_8 &:= F_7.
\end{aligned}$$

The computation with respect to the old state of relation p shows the desired behavior and stops after successfully deriving the fact $p_{bb}(98, 100)$.

Although this example is rather simple, the general positive impact of existential query optimization for evaluating the effectiveness tests in propagation

rules is evident. Since the state simulation via transition rules is usually considered to be very expensive, the optimized evaluation of effectiveness tests for a limited evaluation of simulated state relations becomes especially important. Therefore, the application of existential query optimization for an enhanced new state simulation, i.e., needed for computing induced deletions, is essential as well.

5.2.4 Comparison to Structured Update Propagation

As already mentioned above, the idea of combining the advantages of bottom-up and top-down approaches to update propagation using the Magic Sets method has been first published in [Gri97] resulting in the structured update propagation method. In this section, we will briefly compare soft update propagation with this related approach by means of our running example. For finite Herbrand universes and fixed rule sets, both approaches require time polynomial in the size of the Herbrand universe. However, since structured update propagation is based on the alternating fixpoint computation, it can be shown again that soft update propagation performs at least asymptotically better as any overestimation of facts when applying the alternating fixpoint is avoided. In addition, by using the soft consequence operator as basic evaluation technique it is possible to further simplify the underlying Magic Updates transformation leading to a smaller number of derived relations and rules. Thus, less joins have to be performed and less facts are generated during the fixpoint evaluation process.

For illustrating the differences, let us consider again the stratifiable deductive database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ from Example 5.2 with the well-known transitive closure rules

$\underline{\mathcal{R}}$:

$\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}(\mathtt{X},\mathtt{Y})$
$\mathtt{p}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}(\mathtt{Z},\mathtt{Y})$

and the fact base $\mathcal{F}$

$\underline{\mathcal{F}}$:

```
e(1,2), e(1,4), e(3,4)
e(10,11), e(11,12), ..., e(98,99), e(99,10),e(99,100)
```

For sake of readability we again restrict our attention to the propagation of true insertions and to the application of naive transition rules only. In order to implement structured update propagation, at first a transformation is applied to the augmented rule set, i.e., the union of propagation rules $\varphi(\mathcal{R})$, transition rules $\tau(\mathcal{R})$, and original rules $\mathcal{R}$. The aim of this rewriting is to clearly separate the so-called *propagation rule set* and *query rule set* from the augmented rules, the first responsible for deriving the delta facts, the latter used for evaluating propagation queries. To this end, in each propagation rule the conjunction of side

literals together with the effectiveness test is replaced by a new literal which is constructed from a new predicate symbol as well as the predicate symbol the corresponding propagation rule refers to. Each new predicate is supplied with an adornment indicating the bound arguments. Assuming the new predicate symbols to be $\texttt{i1\_p}_{\texttt{bb}}$, $\texttt{i2\_p}_{\texttt{bbf}}$, and $\texttt{i3\_p}_{\texttt{bbf}}$, the resulting propagation rule set $\mathcal{R}_P$ is as follows

$\underline{\mathcal{R}_P}$:

$$\begin{aligned}
&\Delta^{+}\texttt{p}(\texttt{X},\texttt{Y}) \leftarrow \Delta^{+}\texttt{e}(\texttt{X},\texttt{Y}) \wedge \texttt{i1_p}_{\texttt{bb}}(\texttt{X},\texttt{Y})\\
&\Delta^{+}\texttt{p}(\texttt{X},\texttt{Y}) \leftarrow \Delta^{+}\texttt{e}(\texttt{X},\texttt{Z}) \wedge \texttt{i2_p}_{\texttt{bbf}}(\texttt{X},\texttt{Z},\texttt{Y})\\
&\Delta^{+}\texttt{p}(\texttt{X},\texttt{Y}) \leftarrow \Delta^{+}\texttt{p}(\texttt{Z},\texttt{Y}) \wedge \texttt{i3_p}_{\texttt{bbf}}(\texttt{Z},\texttt{Y},\texttt{X})
\end{aligned}$$

The substituted side literals and effectiveness tests are used to build the adorned allowed rules

$$\begin{aligned}
\texttt{i1_p}_{\texttt{bb}}(\texttt{X},\texttt{Y}) &\leftarrow \neg\texttt{p}(\texttt{X},\texttt{Y})\\
\texttt{i2_p}_{\texttt{bbf}}(\texttt{X},\texttt{Z},\texttt{Y}) &\leftarrow \texttt{p}^{\texttt{new}}(\texttt{Z},\texttt{Y}) \wedge \neg\texttt{p}(\texttt{X},\texttt{Y})\\
\texttt{i3_p}_{\texttt{bbf}}(\texttt{Z},\texttt{Y},\texttt{X}) &\leftarrow \texttt{e}^{\texttt{new}}(\texttt{X},\texttt{Z}) \wedge \neg\texttt{p}(\texttt{X},\texttt{Y})
\end{aligned}$$

which are now transformed together with the original rules $\mathcal{R}$ as well as the transition rules $\tau(\mathcal{R})$ using the Magic Sets rewriting with respect to the abstract propagation queries represented by the newly introduced predicates $\texttt{i1\_p}_{\texttt{bb}}$, $\texttt{i2\_p}_{\texttt{bbf}}$, and $\texttt{i3\_p}_{\texttt{bbf}}$. The resulting rule set then represents the query rules $\mathcal{R}_Q$ and consist of the following answer rules

$$\begin{aligned}
\texttt{i1_p}_{\texttt{bb}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_i1_p}_{\texttt{bb}}(\texttt{X},\texttt{Y}) \wedge \neg\,\texttt{p}_{\texttt{bb}}(\texttt{X},\texttt{Y})\\
\texttt{i2_p}_{\texttt{bbf}}(\texttt{X},\texttt{Z},\texttt{Y}) &\leftarrow \texttt{m_i2_p}_{\texttt{bbf}}(\texttt{X},\texttt{Z}) \wedge \texttt{p}^{\texttt{new}}_{\texttt{bf}}(\texttt{Z},\texttt{Y}) \wedge \neg\,\texttt{p}_{\texttt{bb}}(\texttt{X},\texttt{Y})\\
\texttt{i3_p}_{\texttt{bbf}}(\texttt{Z},\texttt{Y},\texttt{X}) &\leftarrow \texttt{m_i3_p}_{\texttt{bbf}}(\texttt{Z},\texttt{Y}) \wedge \texttt{e}^{\texttt{new}}_{\texttt{fb}}(\texttt{X},\texttt{Z}) \wedge \neg\,\texttt{p}_{\texttt{bb}}(\texttt{X},\texttt{Y})
\end{aligned}$$

$$\begin{aligned}
\texttt{p}^{\texttt{new}}_{\texttt{bf}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_p}^{\texttt{new}}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{e}^{\texttt{new}}_{\texttt{bf}}(\texttt{X},\texttt{Y})\\
\texttt{p}^{\texttt{new}}_{\texttt{bf}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_p}^{\texttt{new}}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{e}^{\texttt{new}}_{\texttt{bf}}(\texttt{X},\texttt{Z}) \wedge \texttt{p}^{\texttt{new}}_{\texttt{bf}}(\texttt{Z},\texttt{Y})
\end{aligned}$$

$$\begin{aligned}
\texttt{e}^{\texttt{new}}_{\texttt{bf}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_e}^{\texttt{new}}_{\texttt{bf}}(\texttt{X}) \wedge \texttt{e}(\texttt{X},\texttt{Y}) \wedge \neg\,\Delta^{-}\texttt{e}(\texttt{X},\texttt{Y})\\
\texttt{e}^{\texttt{new}}_{\texttt{bf}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_e}^{\texttt{new}}_{\texttt{bf}}(\texttt{X}) \wedge \Delta^{+}\texttt{e}(\texttt{X},\texttt{Y})\\
\texttt{e}^{\texttt{new}}_{\texttt{fb}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_e}^{\texttt{new}}_{\texttt{fb}}(\texttt{Y}) \wedge \texttt{e}(\texttt{X},\texttt{Y}) \wedge \neg\,\Delta^{-}\texttt{e}(\texttt{X},\texttt{Y})\\
\texttt{e}^{\texttt{new}}_{\texttt{fb}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_e}^{\texttt{new}}_{\texttt{fb}}(\texttt{Y}) \wedge \Delta^{+}\texttt{e}(\texttt{X},\texttt{Y})
\end{aligned}$$

$$\begin{aligned}
\texttt{p}_{\texttt{bb}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_p}_{\texttt{bb}}(\texttt{X},\texttt{Y}) \wedge \texttt{e}(\texttt{X},\texttt{Y})\\
\texttt{p}_{\texttt{bb}}(\texttt{X},\texttt{Y}) &\leftarrow \texttt{m_p}_{\texttt{bb}}(\texttt{X},\texttt{Y}) \wedge \texttt{e}(\texttt{X},\texttt{Z}) \wedge \texttt{p}_{\texttt{bb}}(\texttt{Z},\texttt{Y})
\end{aligned}$$

as well as the following sub-query rules

$$\begin{aligned}
&\mathtt{m_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z}) \leftarrow \mathtt{m_i2_p}_{\mathtt{bbf}}(\mathtt{X},\mathtt{Z})\\
&\mathtt{m_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z}) \leftarrow \mathtt{m_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X}) \wedge \mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Z})
\end{aligned}$$

$$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{Z}) \leftarrow \mathtt{m\_i3\_p}_{\mathtt{bbf}}(\mathtt{Z},\mathtt{Y})$$

$$\mathtt{m\_e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X}) \leftarrow \mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X})$$

$$\begin{aligned}
&\mathtt{m_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m_i1_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m_i2_p}_{\mathtt{bbf}}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{Z},\mathtt{Y})\\
&\mathtt{m_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m_i3_p}_{\mathtt{bbf}}(\mathtt{Z},\mathtt{Y}) \wedge \mathtt{e}^{\mathtt{new}}_{\mathtt{fb}}(\mathtt{X},\mathtt{Z})\\
&\mathtt{m_p}_{\mathtt{bb}}(\mathtt{Z},\mathtt{Y}) \leftarrow \mathtt{m_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Z})
\end{aligned}$$

as well as the following propagation seeds

$$\begin{aligned}
&\mathtt{m_i1_p}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_i2_p}_{\mathtt{bbf}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y})\\
&\mathtt{m_i3_p}_{\mathtt{bbf}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y}).
\end{aligned}$$

In principle, this rule set coincides with the Magic Updates rewritten rules $\mathtt{mu}(\mathcal{R}^{\mathtt{p}}_{\mathtt{Q}^{\mathtt{u}}})$ as presented in Section 5.2.1. The only difference lies in the application of the newly introduced predicates $\mathtt{i1\_p}_{\mathtt{bb}}$, $\mathtt{i2\_p}_{\mathtt{bbf}}$, and $\mathtt{i3\_p}_{\mathtt{bbf}}$ representing the conditions under which a delta fact induces a new delta fact to be derived by using the propagation rules in $\mathcal{R}_P$. The reason for separating the magic rewritten rules into the propagation rules $\mathcal{R}_P$ and the set of query rules $\mathcal{R}_Q$ is that the complete alternating fixpoint computation of the entire (and possibly unstratifiable) rule set can be avoided this way. Instead, all negative cycles involving delta literals are cut such that alternating fixpoint computation is solely needed for correctly evaluating $\mathcal{R}_Q$. For evaluating $\mathcal{R}_P$, however, the simple computation of the least fixpoint of $T^{\star}_{\mathcal{R}_P}$ is already sufficient. In structured update propagation the two rule sets $\mathcal{R}_Q$ and $\mathcal{R}_P$ are now mutually applied each time employing the correctly determined facts of the other rule set until no more new delta facts can be derived with $\mathcal{R}_P$.

Note that the application of alternating fixpoint computation is still imperative, as the Magic Sets transformed query rules may be partly unstratifiable. Although all unstratifiable cycles in the query rules $\mathcal{R}_Q$ of our running example are cut, adding the following rule

$$\mathtt{no\_cycle}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{p}(\mathtt{Y},\mathtt{Y}) \wedge \neg\mathtt{p}(\mathtt{X},\mathtt{X})$$

for defining the relation `no_cycle` would result in an unstratifiable query rule set. In this case, $\mathcal{R}_Q$ would additionally contain the query rules

$$\mathtt{m\_i4\_no\_cycle}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \Delta^{+}\mathtt{e}(\mathtt{X},\mathtt{Y}) \qquad (\mathrm{R}_1)$$

$$\mathtt{p}^{\mathtt{new}}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \wedge \ \mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Y}) \tag{R_2}$$
$$\mathtt{p}^{\mathtt{new}}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \leftarrow \mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \wedge \ \mathtt{e}^{\mathtt{new}}_{\mathtt{bf}}(\mathtt{X},\mathtt{Z}) \wedge \mathtt{p}^{\mathtt{new}}_{\mathtt{bb}}(\mathtt{Z},\mathtt{Y}) \tag{R_3}$$

$$\mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bb}}(\mathtt{Y},\mathtt{Y}) \leftarrow \mathtt{m\_i4\_no\_cycle}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \tag{R_4}$$
$$\mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bb}}(\mathtt{X},\mathtt{X}) \leftarrow \mathtt{m\_i4\_no\_cycle}_{\mathtt{bb}}(\mathtt{X},\mathtt{Y}) \wedge \neg \mathtt{p}^{\mathtt{new}}_{\mathtt{bb}}(\mathtt{Y},\mathtt{Y}) \tag{R_5}$$

for propagating true insertions with respect to `no_cycle`, leading to the following negative cycle in the corresponding dependency graph:

$$\mathtt{p}^{\mathtt{new}}_{\mathtt{bb}} \xrightarrow{neg} \mathtt{m\_p}^{\mathtt{new}}_{\mathtt{bb}} \xrightarrow{pos} \mathtt{p}^{\mathtt{new}}_{\mathtt{bb}}.$$

Let $u_{\mathcal{D}}$ be an update consisting of the new edge facts $e(2,1)$ and $e(10,2)$ to be inserted into $\mathcal{D}$, i.e., $u^{+}_{\mathcal{D}} = \{e(2,1), e(10,2)\}$ inducing no insertions with respect to *no_cycle* but the fact *no_cycle*$(1,2)$ to be deleted from $\mathcal{D}$. During the alternating fixpoint evaluation of the above rule set, however, the application of the rule R_5 which negatively references relation p^{new}_{bb} leads to the overestimated sub-query fact $m\_p^{new}_{bb}(10,10)$. This in turn results in the redundant generation of further sub-query facts $\{m\_p^{new}_{bb}(10,1), m\_p^{new}_{bb}(10,2)\} \cup \{m\_p^{new}_{bb}(10,11), \ldots, m\_p^{new}_{bb}(10,100)\}$ as well as corresponding answer facts $\{p^{new}_{bb}(10,1), p^{new}_{bb}(10,2)\} \cup \{p^{new}_{bb}(10,10), \ldots, p^{new}_{bb}(10,100)\}$ which would not have been derived in the soft update propagation approach. This is due to the fact that the softly stratifiable rules cannot derive the sub-query fact $m\_p^{new}_{bb}(10,10)$ because of the answer facts $p^{new}_{bb}(1,1)$ and $p^{new}_{bb}(2,2)$ generated before, and thus, avoiding any overestimations.

It is obvious that the application of the soft consequence operator for evaluating the query rules is more efficient than the application of alternating fixpoint computation. The reason is that the former approach avoids any overestimation of facts as already discussed in Section 4.2.3. The desire to apply the expensive alternating fixpoint to the smallest possible rule set has led to the division of the augmented rules into two sub-rule sets $\mathcal{R}_P$ and $\mathcal{R}_Q$. The advantage is that only definitely true facts with respect to delta literals are computed as it is known that no not definitely false facts have to be generated during the course of alternating fixpoint computation. A disadvantage is that new relations have to be introduced for copying results between the rule sets $\mathcal{R}_P$ and $\mathcal{R}_Q$ leading to the generation of additional facts. Another deficiency of this approach is that the alternating evaluation of both rule sets makes relational optimization hard or even impossible. Thus, it can be concluded that soft update propagation indeed represents an improved evaluation method in comparison to the related structured update propagation approach.

5.3 Applications of Update Propagation

The computation of the implicit changes of derived relations caused by a base update is helpful to provide a deeper understanding of a complex database schema

and interdependencies within it. In addition, databases with a powerful trigger component may allow the definition of triggers not only with respect to changes in base relations but also with respect to changes in derived relations, for which the computation of the implicit changes then becomes necessary. In the literature, however, incremental methods to update propagation have been mainly investigated for providing implementations to integrity checking (e.g. [Dec86, LST87, DW89, KSS87, QS87, CW90, Küc91, Oli91, BMM91, LL96]) and materialized view maintenance (e.g. [CW91, BMM91, GMS93, CW94, GL95, CGL+96, Gri97, BDD+98, DS00, SBLC00]). As incremental update propagation represents an essential sub-task common to integrity checking and view maintenance, propagation methods for these two database services have been proposed as well, e.g. [UO92, TO95, RSS96, CKL+97, MT00]. These two classical database tasks obviously represent the most relevant applications for update propagation. Therefore, we will briefly discuss in this section how soft update propagation can be used as a basic evaluation technique for each of them.

5.3.1 Integrity Checking

The enforcement of integrity is a crucial issue, as the quality of a database essentially depends on the quality in which an application is presented. Hence, integrity constraints have to be verified each time a database is updated, i.e., at the end of each transaction. In case of a violation of at least one integrity constraint, consistency can be maintained by either rolling back the transaction or by applying further updates in order to repair the violated constraints.

Generally, static constraints are closed first order formulas describing admissible database states. In our database from Example 5.2, we may require that for each cycle in the path relation at least one direct cycle between the underlying edge facts exists:

$$\forall\ \mathtt{X}\ \mathtt{p}(\mathtt{X},\mathtt{X}) \Rightarrow \exists \mathtt{Y}\ \mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{e}(\mathtt{Y},\mathtt{X}).$$

Practically, it is more convenient to specify integrity constraints by means of ground atoms derivable in every database state. In our example, the same integrity constraint can be expressed using the deductive rules

$$\begin{aligned}
&\mathtt{ic}_1 \leftarrow \neg \mathtt{aux}_1\\
&\mathtt{aux}_1 \leftarrow \mathtt{p}(\mathtt{X},\mathtt{X}) \wedge \neg \mathtt{aux}_2(\mathtt{X})\\
&\mathtt{aux}_2(\mathtt{X}) \leftarrow \mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{e}(\mathtt{Y},\mathtt{X}).
\end{aligned}$$

The proper integrity constraint then is just $\mathtt{ic}_1$. Note that it is always possible to transform integrity constraints into such a rule-based representation (cf. [Llo87, CGH94]). Specifying constraints via derived or base facts has the advantage that integrity checking is reduced to the question whether an update leads to the

deletion of one of these ground atoms from the implicit state of a given database. Hence, in our example a constraint violation would be indicated by the derivation of the delta fact $\Delta^{-}\mathtt{ic}_1$ during an update propagation process.

The Magic Updates rewriting presented so far is performed with respect to all derived relations in a given database. Although the evaluation of Magic Updates transformed propagation rules is limited to the actually affected delta relations, it is possible to further refine the propagation rules if it is known for certain (delta) relations that they will always be empty. For instance, integrity constraints as defined in Section 2.3 have to be solely checked for induced deletions if an originally consistent database is updated. The construction used in the proof of Theorem 5.1 already implies the possibility of further enhancing the magic update propagation process. In the proof, an auxiliary 0-ary relation h has been used for referencing delta relations which are possibly affected by a given base update. If integrity constraints are taken into account, we can exclude rules for defining h which contain references to induced insertions, as these delta relations must be empty. In addition, it is often possible to further refine the resulting rules using the same argumentation as above. As an example consider the propagation rules for induced deletions with respect to the above introduced constraint $\mathtt{ic}_1$:

$$\Delta^{-}\mathtt{ic}_1 \leftarrow \Delta^{+}\mathtt{aux}_1 \wedge \neg\mathtt{ic}_1^{\mathtt{new}}$$

$$\Delta^{+}\mathtt{aux}_1 \leftarrow \Delta^{+}\mathtt{p}(\mathtt{X},\mathtt{X}) \wedge \neg\,\mathtt{aux}_2^{\mathtt{new}}(\mathtt{X}) \wedge \neg\mathtt{aux}_1$$
$$\Delta^{+}\mathtt{aux}_1 \leftarrow \Delta^{-}\mathtt{aux}_2(\mathtt{X}) \wedge \mathtt{p}^{\mathtt{new}}(\mathtt{X},\mathtt{X}) \wedge \neg\mathtt{aux}_1$$

$$\Delta^{-}\mathtt{aux}_2(\mathtt{X}) \leftarrow \Delta^{-}\mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \mathtt{e}(\mathtt{Y},\mathtt{X}) \wedge \neg\mathtt{aux}_2^{\mathtt{new}}(\mathtt{X})$$
$$\Delta^{-}\mathtt{aux}_2(\mathtt{X}) \leftarrow \Delta^{-}\mathtt{e}(\mathtt{Y},\mathtt{X}) \wedge \mathtt{e}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{aux}_2^{\mathtt{new}}(\mathtt{X}).$$

Assuming that the relations $\mathtt{aux}_1/0$ and $\mathtt{aux}_2/1$ are solely needed for defining the constraint $\mathtt{ic}_1$, it is not necessary to consider rules for defining the delta relations $\Delta^{-}\mathtt{aux}_1/0$ and $\Delta^{+}\mathtt{aux}_2/1$ in the set of optimized propagation rules. This is due to the fact that only delta facts in $\Delta^{+}\mathtt{aux}_1/0$ and $\Delta^{-}\mathtt{aux}_2/1$ may induce the derivation of $\Delta^{-}\mathtt{ic}_1$. Additionally, it is possible to drop the effectiveness tests in the first three rules as no alternative derivation paths with respect to $\mathtt{ic}_1$ exist and the old state relation $\mathtt{aux}_1/0$ will always be empty in a consistent database state. Note that is possible to automatically enhance the transformed rule set in the described way although the general problem of finding an optimized rule set (which needs less rules or joins but is equivalent to the original one) is undecidable.

First approaches to incremental integrity checking have been proposed using closed first order formulas as constraints (e.g. [Nic82, BBC80, BB82]). The basic idea was to simplify a given constraint in such a way that the resulting conditions directly refer to base relation updates. This simplification basically coincides with our transformation-based approach which uses optimized propagation rules.

In general, universally and existentially quantified formulas in conjunctive normal form have been distinguished, each of them being simplified in a different way. The simplification of universally quantified constraints leads to specialized constraints which are well-optimized with respect to a given base update. This is also the case for our example in which the resulting propagation rules provide a focus on the relevant changes with respect to the constraint $\mathtt{ic}_1$. Simplifying existentially quantified formulas, however, turned out to be much more complicated as it is not always possible to specialize them with respect to induced updates. As an example consider the following constraint which requires that at least one cycle in path has to exist:

$$\exists \mathtt{X}\ \mathtt{p}(\mathtt{X},\mathtt{X}).$$

This condition can be expressed by the following deductive rule

$$\mathtt{ic}_2 \leftarrow \mathtt{p}(\mathtt{X},\mathtt{X})$$

which defines the integrity constraint $\mathtt{ic}_2$. The corresponding propagation and transition rule for defining $\Delta^{-}\mathtt{ic}_2$ are

$$\begin{aligned}\Delta^{-}\mathtt{ic}_2 &\leftarrow \Delta^{-}\mathtt{p}(\mathtt{X},\mathtt{X}) \wedge \neg \mathtt{ic}_2^{\mathtt{new}}\\ \mathtt{ic}_2^{\mathtt{new}} &\leftarrow \mathtt{p}^{\mathtt{new}}(\mathtt{X},\mathtt{X}).\end{aligned}$$

Note that it is not possible to avoid the effectiveness test in the propagation rule because of alternative cycle facts in p. Hence, it is necessary to determine the new state relation $\mathtt{ic}_2^{\mathtt{new}}/0$ which leads to the complete materialization of $\mathtt{p}^{\mathtt{new}}/2$. This is due to the fact that no bindings from the induced updates in $\Delta^{-}\mathtt{p}/2$ can be passed to the new state relation $\mathtt{p}^{\mathtt{new}}/2$ when applying the Magic Updates transformation. Thus, the complete new state of p has to be determined during the update propagation process and the computed induced updates cannot be used at all.

Solutions for the problem of optimizing so-called non-complacent assertions have been already proposed in [BB82]. The authors suggest to generate pretests which are easier to evaluate than the original integrity constraint. If a pretest is successful, i.e., it can be evaluated to TRUE, the corresponding integrity constraint must be satisfied as well. If a pretest is evaluated to FALSE, the entire integrity constraint must be checked again for a possible violation. A quite similar optimization effect in our approach can be achieved by applying the existential query optimization technique proposed in Section 4.3 and Section 5.2.3. As already mentioned in these sections, however, our proposed optimization of existential queries does not represent a complete solution for the general case. Therefore, both approaches, existential magic sets rewriting or the application of pretests, cannot provide a complete solution to the problem of incremental integrity checking with respect to existential constraints yet.

5.3.2 Materialized Views

The motivation for materializing derived relations (or views) is to provide fast access to frequently queried relations having such a complex definition that they should not be recomputed for every query. Once a relation has been materialized, this relation can be treated like a base relation and query evaluation can be further supported by building up index structures [GM95].

However, a modification of the fact base may induce changes of derived relations such that the current materialization no longer coincides with its definition. Hence, for each affecting base update the materialized relation has to be adapted. As in most cases, only a small portion of the derived relation will be changed by a base update, it is rarely expedient to entirely recompute the new state of the relation by means of transition rules. Instead, only the particular changes represented by delta facts ought to be computed in order to incrementally maintain the materialized relation.

Although update propagation is an essential step towards the incremental maintenance of materialized relations, the case that such relations are referred to during the process of update propagation has not been considered yet. As materialized relations will only be adapted according to the result of update propagation, they remain in their old state during the entire propagation process. Hence, for materialized relations the new state has to be simulated only. In case of new state simulation, this causes no problems. References to the old state can be performed by accessing the materialized relation and the new state can be simulated as for any other derived relation.

However, if the old state simulation is considered, for all extensional and virtual derived relations the old state has to be simulated while for materialized relations the new state is to be computed. References to the old state of such a relation are again evaluated by accessing the materialized relation like any other base relation. Evaluations on the new state, however, have to be performed via the deductive rules originally defining the relation, as done for any other derived relation.

It can be concluded that the soft update propagation approach needs only slight modifications when materialized views are considered due to the above mentioned particularities with respect to state simulation. Query evaluation, however, is performed by considering materialized views as ordinary base relations. Thus, during the application of Magic Updates rewriting as presented in Definition 5.8 all body literals referring to the old state of a materialized relation are considered to be non-derived literals for which no sub-query rules are generated. If the soft update propagation approach is modified in this way, the results of the propagation process can be used for maintaining the materialized views by simply applying the corresponding delta facts.

5.4 Discussion

In this chapter, we have presented a new bottom-up evaluation method for computing the implicit changes of derived relations resulting from explicitly performed updates of the extensional fact base. The proposed transformation-based approach derives deductive propagation rules by means of range-restricted $Datalog^{\neg}$ rules which can be automatically generated from a given stratifiable database schema. We use the Magic Sets method to combine the advantages of top-down and bottom-up propagation approaches in order to restrict the computation of true updates to the affected part of the database only. To do so has been first proposed in [Gri97], where structured update propagation has been introduced as computation method for the potentially unstratifiable magic propagation rules. Structured update propagation is based on the alternating fixpoint computation proposed by Van Gelder [vG93] in order to determine the well-founded model of the possibly unstratifiable magic propagation rules correctly. The application of the alternating fixpoint computation, however, is not really efficient as the specific reason for unstratifiability (namely the application of the magic sets transformation to a stratified rule set) is not taken into account. Therefore, we propose a less complex magic updates transformation resulting in a set of rules which is not only smaller but may in addition be efficiently evaluated using the soft stratification approach. Thus, less joins have to be performed and less facts are generated.

Incremental methods for update propagation have been mainly studied in the context of Datalog (e.g. [Dec86, KSS87, LST87, BDM88, Küc91, Oli91, BMM91, UO92, GMS93, CW94, Man94, UO94, TO95, LL96, MT99, MT00]), relational algebra (e.g. [QW91, Man94, GL95, CGL+96, CKL+97, BDD+98, DS00, SBLC00]), and SQL (e.g. [CW90, CW91]). Methods in Datalog are often based on SLDNF resolution and thus cannot guarantee termination for recursively defined predicates. In addition, a set-oriented evaluation technique is preferred in the database context. Approaches formulated in relational algebra or SQL so far are also not capable of handling recursion, the latter usually based on transformed views or specialized triggers. Transformed SQL-views directly correspond to our proposed method in the non-recursive case. The application of triggers (e.g. as proposed by Ceri/Widom) does not allow the reuse of intermediate results obtained by querying the derivability and effectiveness tests.

Summarizing the benefits of our approach: Soft update propagation can handle recursion and may also propagate updates at arbitrary granularity (cf. [Gri97]). It combines the advantages of top-down and bottom-up approaches by using a simple set-oriented fixpoint process which is well-suited for being transferred into the SQL context. In contrast, a pure top-down approach would need a more elaborated control in order to implement the propagation of base updates to recursive views. The incorporation of query evaluation via Magic Sets allows for

reusing intermediate query results. Additionally, the fixpoint evaluation can be easily combined with other relational optimization techniques.

The propagation rules proposed in this chapter are restricted to the propagation of insertions and deletions of base facts in stratifiable databases. However, up till now, several approaches have been proposed dealing with further kinds of updates or additional language concepts. As far as the latter are concerned, update propagation in presence of built-ins and (numerical) constraints has been discussed in [Wüt90], while views possibly containing duplicates are considered in [CW91, GL95]. Aggregates and updates have been investigated in [BW93, GMS93]. As for the various types of updates, methods have been introduced for dealing with the modification of individual tuples, e.g. [CW91, UO92], the insertion and deletion of rules (respectively view definitions) and constraints, e.g. [MB88, SK88], and even changes of view and constraint definitions, e.g. [GMR95]. A taxonomy of view maintenance problems based upon the maintenance of materialized views is given in [GM95]. All these techniques may allow for enhancing the propagation rules introduced in this section in order to provide a more elaborate approach to update propagation. However, there is still the question to be answered to which extent these techniques can be incorporated into our proposed framework.

Chapter 6

Well-founded Model Computation

In Chapter 2, three different database classes have been introduced resulting from certain combinations of recursion and negation among the respective deductive rules. Unstratifiable databases represent the most general class posing no restrictions on the combination of recursion and negation at all. Based on model theory, several proposals for a suitable semantics of unstratifiable negation have been made. The most well-established of these are the stable model semantics and the well-founded semantics, the latter being preferred by many authors because of its unique model [vGRS91]. The reason for dealing with this most general class is that unstratifiable rules are strictly more expressive than stratifiable ones [Kol91]. Hence, a general method for computing the well-founded model subsumes all other rule classes introduced above and can be exploited for any deductive database service which requires the materialization of intensional facts.

Basically, bottom-up approaches to the general computation of well-founded models can be divided into methods using the alternating fixpoint (e.g. [vG93, KSS95, SNV95]) and into those based on the residual program evaluation (e.g. [Bry89, DK89, BD95, BZF96]). The advantage of the residual program approach is that the conditional facts used provide additional information about why certain facts are considered undefined in the resulting three-valued well-founded model. On the other hand, conditional facts and algorithms working with them are not that well-suited for being implemented in a relational database context (see Section 6.3). Therefore, we will concentrate on the efficient implementation of the alternating fixpoint procedure and its well-known drawback of repeated computations.

In Chapter 3, we already introduced alternating fixpoint computation as proposed in [vG89] and based on that a slightly modified version for computing the equivalent KSS Alternating Fixpoint Model suggested in [KSS91, KSS95].

Both approaches, however, suffer from the drawback of repeated computations, as in each iteration round most of the facts from previous iterations are redundantly recomputed. Therefore, we suggest to use the results from Chapter 4 and Chapter 5 for extending alternating fixpoint computation by applying Magic Updates transformed propagation rules. The resulting approach avoids any repeated computations [Beh01] and represents a generalization of the differential fixpoint computation well-known for stratifiable deductive databases [BR87].

In Section 6.1 the doubled program approach is presented for computing the KSS Alternating Fixpoint Model. In Section 6.2 we propose an enhancement of this approach by applying update propagation. To this end, we discuss the generally positive impact of using update propagation rules for evaluating doubled programs in the first Subsection 6.2.1. Afterwards, in Subsection 6.2.2, we introduce a simplified soft consequence operator for evaluating Magic Updates transformed rules in doubled programs. Section 6.3 finally concludes this chapter with a comparison of efficient approaches for evaluating residual programs.

6.1 The Doubled Program Approach

In Section 3.3.2 we argued that the alternating fixpoint approach to constructing the well-founded model of arbitrary databases is not particularly well-suited for being directly implemented, as it works on negative conclusions. Instead, we introduced the algorithm presented in [KSS91] where the sets of not definitely false facts are explicitly stored and only their complement is used to refer to true negative conclusions implicitly. The disadvantage of this method so far is that we have to carefully distinguish between the sets of definitely true and not definitely false facts. As these sets need not be disjoint we cannot simply store them as ordinary facts in the same database or relation.

Therefore, the authors proposed to introduce for each relation referencing definitely true facts a second relation for not definitely false facts. In order to work on these relations, the entire database is doubled, and in each half the deductive rules are rewritten such that negative literals reference relations of the other half. This way, one half is employed for computing definitely true facts, and the other one for determining not definitely false facts. However, rules from the two halves are never applied together. In the following we will call this transformation the *doubled program rewriting*.

In a first step, the entire database, i.e., all facts and rules, are duplicated. Then in one of these copies each positive literal $p(\vec{x})$ is replaced by $dt\_p(\vec{x})$ and each negative literal $\neg q(\vec{y})$ is replaced by $\neg ndf\_q(\vec{y})$ where dt_p and ndf_q are a new predicate symbols. In the other copy all positive atoms $p(\vec{x})$ (occurring in facts and rules) are replaced by $ndf\_p(\vec{x})$ and all negative atoms $\neg q(\vec{y})$ are replaced by $\neg dt\_q(\vec{y})$. As an example consider again the unstratifiable database from Example 3.1 in Chapter 3. Rewriting the single rule

```
e(X) ← succ(X,Y) ∧ ¬e(Y)
```

for defining relation `e` and the sample fact base would lead to

```
dt_e(X)  ← dt_succ(X,Y) ∧ ¬ndf_e(Y)

dt_succ(0,1)
dt_succ(1,2)
dt_succ(2,3)
dt_succ(3,4)
dt_succ(4,5).
```

and

```
ndf_e(X) ← ndf_succ(X,Y) ∧ ¬dt_e(Y)

ndf_succ(0,1)
ndf_succ(1,2)
ndf_succ(2,3)
ndf_succ(3,4)
ndf_succ(4,5).
```

Note that both halves of the doubled database are semi-positive if considered separately. Determining the alternating fixpoint of the doubled database implies to alternately compute the fixpoint of both halves, each time evaluating negative literals with respect to the facts derived from the other half. In our example, the relation dt_e is supposed to hold definitely true facts (DT). Assuming a two-valued semantics, $\neg dt\_e(4)$ is true if $dt\_e(4)$ is not included in dt_e and thus is not known to be definitely true (NDT) at the current stage. It is obvious that this directly corresponds to the conjugate of dt_e. In contrast to this, the relation ndf_e comprises the facts not known to be definitely false (NDF) and hence if $\neg ndf\_e(4)$ is true then $dt\_e(4)$ is known to be definitely false (DF).

Due to the doubling of the database, fixpoint computations for both halves of the database will now derive facts of different relations. This has the positive effect that all facts can be summarized in one common fact base. This furthermore implies that each individual fixpoint can be obtained by differential fixpoint computation as considered in Section 3.1.2.

Before we introduce the algorithm for calculating the well-founded model of a database, we improve and formally define the notion of doubled program rewriting. Note that it is not necessary to double the relations for all predicates of the entire program. Predicates not relying on unstratified negation are known to have a total well-founded model and their state can be uniquely represented by the set of true facts. Hence, our example database may be transformed to

```
dt_e(X)  ← succ(X,Y) ∧ ¬ndf_e(Y)

ndf_e(X) ← succ(X,Y) ∧ ¬dt_e(Y)

succ(0,1)
succ(1,2)
succ(2,3)
succ(3,4)
succ(4,5)
```

leaving the relation `succ` in its original form as it does not rely on unstratified negation. The benefit of restricting the transformation to the unstratified part of the database is that iterated fixpoint computation can be applied as long as no unstratified negation occurs. The following definitions ensure that the stratified part of a database remains in its original form such that model computation for this part coincides with iterated fixpoint computation.

Definition 6.1 (Stratified Layer/Stratified Predicate Symbol) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database and λ a layering on $\mathcal{R}$.*

1. *A layer $\mathcal{R}_l$ ($1 \leq l \leq n$) defined by λ on $\mathcal{R}$ is called stratified with respect to λ if there exist no predicate symbols $p, q \in \mathtt{pred}(\mathcal{D})$ with $\lambda(p) \leq l$ such that $\lambda(p) = \lambda(q)$ and $p \overset{-}{\leftarrow} q$ or $\lambda(p) > \lambda(q)$ and $q \overset{-}{\leftarrow} p$. Otherwise, $\mathcal{R}_l$ is called unstratified with respect to λ.*

2. *A predicate symbol $p \in \mathtt{pred}(\mathcal{D})$ is called stratified with respect to λ if $\lambda(p)$ is a stratified layer of $\mathcal{R}$ or $\lambda(p) = 0$. Otherwise, p is called unstratified with respect to λ.*

In the following the phrase "with respect to λ" is omitted, as the respective layering will always be obvious. Note that Definition 6.1 does not require the given layering to be maximal. Thus, it is possible that even stratifiable rule sets may be partitioned in such a way that unstratified layers are contained. In the worst case, the layering partitions the entire rule set into one layer only such that full alternating fixpoint computation is required for computing the well-founded model. However, if the layering is a stratification, the rule set consists of stratified layers only, each of which can be evaluated by differential fixpoint computation.

In Figure 6.1 the predicate dependency graph of a stratifiable database $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ is presented. The layering, however, partitions the rule set into three layers where $\mathcal{R}_2$ includes a negative dependency such that the layers $\mathcal{R}_2$ and $\mathcal{R}_3$ are unstratified. Thus, full alternating fixpoint computation is required for correctly calculating the well-founded model of them. Note that layer $\mathcal{R}_3$ does not

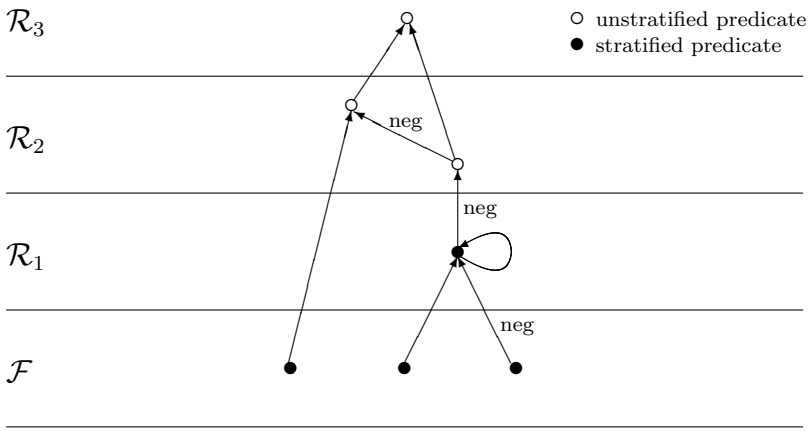


Figure 6.1: Stratified and unstratified layers

include negative dependencies but relies on the result computed for layer $\mathcal{R}_2$. As this is represented by definitely true and not definitely false facts, materialization of $\mathcal{R}_3$ requires alternating fixpoint computation, too. In contrast to this, $\mathcal{R}_1$ may be evaluated by differential fixpoint computation, as it does not rely on any unstratified negation.

In order to homogeneously treat stratified and unstratified predicates in the definition of doubled program rewriting we introduce the definitely true form and not definitely false form for both kinds of predicates. For stratified predicates, both forms coincide with their original form.

Definition 6.2 (Definitely True Form) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database and λ a layering on $\mathcal{R}$. The injective mapping* `dt` *assigns to each literal L with* $\texttt{pred}(L) \in \texttt{pred}(\mathcal{D})$ *its definitely true form such that*

1. *If $L \equiv p(t_1, \ldots, t_n)$ is a positive literal, then*

$$\texttt{dt}(L) \;:=\; \begin{cases} p(t_1, \ldots, t_n) & \text{if } p \text{ is stratified} \\ dt\_p(t_1, \ldots, t_n) & \text{if } p \text{ is unstratified.} \end{cases}$$

2. *If $L \equiv \neg p(t_1, \ldots, t_n)$ is a negative literal, then*

$$\texttt{dt}(L) \;:=\; \begin{cases} \neg p(t_1, \ldots, t_n) & \text{if } p \text{ is stratified} \\ \neg ndf\_p(t_1, \ldots, t_n) & \text{if } p \text{ is unstratified.} \end{cases}$$

3. *The mapping* `dt` *may be applied to conjunctions and sets of literals as follows:*

$$\begin{aligned}\texttt{dt}(L_1 \wedge \ldots \wedge L_n) &:= \bigwedge_{1\leq i\leq n} \texttt{dt}(L_i)\\ \texttt{dt}(\{L_1,\ldots,L_n\}) &:= \bigcup_{1\leq i\leq n} \texttt{dt}(L_i).\end{aligned}$$

Definition 6.3 (Not Definitely False Form) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}\rangle$ be a deductive database and λ a layering on $\mathcal{R}$. The injective mapping* ndf *assigns to each literal L with $\texttt{pred}(L) \in \texttt{pred}(\mathcal{D})$ its not definitely false form such that*

1. *If $L \equiv p(t_1,\ldots,t_n)$ is a positive literal, then*

$$\texttt{ndf}(L) := \begin{cases} p(t_1,\ldots,t_n) & \text{if } p \text{ is stratified}\\ ndf\_p(t_1,\ldots,t_n) & \text{if } p \text{ is unstratified.}\end{cases}$$

2. *If $L \equiv \neg p(t_1,\ldots,t_n)$ is a negative literal, then*

$$\texttt{ndf}(L) := \begin{cases} \neg p(t_1,\ldots,t_n) & \text{if } p \text{ is stratified}\\ \neg dt\_p(t_1,\ldots,t_n) & \text{if } p \text{ is unstratified.}\end{cases}$$

3. *The mapping* ndf *may be applied to conjunctions and sets of literals as follows:*

$$\begin{aligned}\texttt{ndf}(L_1 \wedge \ldots \wedge L_n) &:= \bigwedge_{1\leq i\leq n} \texttt{ndf}(L_i)\\ \texttt{ndf}(\{L_1,\ldots,L_n\}) &:= \bigcup_{1\leq i\leq n} \texttt{ndf}(L_i).\end{aligned}$$

Definition 6.4 (Doubled Program Rewriting) *Let $\mathcal{R}$ be a deductive rule set and λ a layering on $\mathcal{R}$. The doubled program rewriting of $\mathcal{R}$ is the set of rules $\mathcal{R}^{\texttt{dp}} := \mathcal{R}^{\texttt{dt}} \cup \mathcal{R}^{\texttt{ndf}}$ where $\mathcal{R}^{\texttt{dt}}$ and $\mathcal{R}^{\texttt{ndf}}$ are stratifiable rule sets defined as follows:*

$$\begin{aligned}\mathcal{R}^{\texttt{dt}} &:= \{\ \texttt{dt}(A) \leftarrow \texttt{dt}(W)\ \mid A \leftarrow W \in \mathcal{R}\}\\ \mathcal{R}^{\texttt{ndf}} &:= \{\texttt{ndf}(A) \leftarrow \texttt{ndf}(W) \mid A \leftarrow W \in \mathcal{R} \text{ and } \lambda(\texttt{pred}(A)) \text{is unstratified}\}.\end{aligned}$$

Before we define the first algorithm, i.e., AFP materialization, for computing the alternating fixpoint, we still have to introduce one more notion. In order to get access to definitely true and not definitely false facts separately after a fixpoint computation has been applied, we introduce the notion dt- and ndf-restriction. This is because the fact base contains both, and hence each fixpoint may include facts belonging to the other half of the database. Applying the dt- or ndf-restriction to a set of facts, yields the subset of DT- or NDF- facts in this set, respectively.

Definition 6.5 (dt- and ndf-Restriction) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}\rangle$ be a deductive database, $\mathcal{D}^{\texttt{dp}} = \langle \mathcal{F}, \mathcal{R}^{\texttt{dp}}\rangle$ the deductive database derived from $\mathcal{D}$ by applying the doubled program rewriting to $\mathcal{R}$ and $\mathcal{H}_{\mathcal{D}^{\texttt{dp}}}$ the Herbrand base of $\mathcal{D}^{\texttt{dp}}$. For a set of ground atoms $\mathcal{I}^{\texttt{dp}} \subseteq \mathcal{H}_{\mathcal{D}^{\texttt{dp}}}$ we define:*

Algorithm 4: AFP materialization

$i \quad := 0;$
$\mathbb{DT}^0 \quad := \mathtt{lfp}(T^{\star}_{\mathcal{R}^{\mathtt{dt},\circ} \cup \mathcal{R}^{\mathtt{dt},\times}}, \mathcal{F});$
repeat
  $i \quad := i + 1;$
  $\mathbb{NDF}^i \quad := \mathtt{lfp}(T^{\star}_{\mathcal{R}^{\mathtt{ndf}}}, \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^{i-1})) \cup \mathbb{DT}^{i-1})|_{\mathtt{ndf}};$
  $\mathbb{DT}^i \quad := \mathtt{lfp}(T^{\star}_{\mathcal{R}^{\mathtt{dt},\times} \cup \mathcal{R}^{\mathtt{dt},*}}, \mathbb{DT}^{i-1} \cup \mathbb{NDF}^i)|_{\mathtt{dt}};$
until $\mathbb{DT}^i = \mathbb{DT}^{i-1};$
$\mathbb{DT} \quad := \mathbb{DT}^i;$
$\mathbb{NDF} \quad := \mathbb{NDF}^i;$

$$\mathcal{I}^{\mathtt{dp}}|_{\mathtt{dt}} := \{\, \mathtt{dt}(A) \mid A \in \mathcal{H}_{\mathcal{D}} \textit{ and } \mathtt{dt}(A) \in \mathcal{I}^{\mathtt{dp}} \}$$
$$\mathcal{I}^{\mathtt{dp}}|_{\mathtt{ndf}} := \{\, \mathtt{ndf}(A) \mid A \in \mathcal{H}_{\mathcal{D}} \textit{ and } \mathtt{ndf}(A) \in \mathcal{I}^{\mathtt{dp}} \}.$$

The general iteration scheme of AFP materialization based on the doubled program rewriting with respect to one layer is presented in Algorithm 4. This algorithm basically coincides with the one-layered version of the iterated alternating fixpoint computation as presented in Algorithm 3 in Section 3.3.2 and will serve as a starting point for further improvements. Using one layer only implies that all derived relations are considered unstratified if the input rule set $\mathcal{R}$ contains at least one negative derived literal. Otherwise, the rule set for defining not definitely false relations is empty, i.e., $\mathcal{R}^{\mathtt{ndf}} = \emptyset$, and the set for defining definitely true relations is identical with the semi-positive input rules, i.e., $\mathcal{R}^{\mathtt{dt}} = \mathcal{R}$. Note that because of the doubled program rewriting the inner fixpoint computations may use the simple immediate consequence operator again.

As already mentioned above, AFP materialization coincides with the scheme of iterated alternating fixpoint computation as presented in Section 3.3.2 from a structural point of view. The main difference can be discovered in the way the individual fixpoints are computed. However, the results are essentially the same if one layer is considered only. Then for the sets $\mathbb{DT}^i$ and $\mathbb{NDF}^i$ obtained by AFP materialization the following equations would hold

$$DT_1^i = \mathtt{dt}^{-1}(\mathbb{DT}^i)$$
$$NDF_1^i = \mathtt{ndf}^{-1}(\mathbb{NDF}^i)$$

where DT_1^i and NDF_1^i represent the definitely true and the not definitely false sets computed in Algorithm 3 with respect to layer 1. The iteration of AFP materialization terminates if the set of definitely true facts does not change anymore. The well-founded model $\mathcal{M}_{\mathcal{D}}$ is then represented by $\mathtt{dt}^{-1}(\mathbb{DT}) \uplus \neg \cdot \overline{\mathtt{ndf}^{-1}(\mathbb{NDF})}$.

Due to the doubled program rewriting of the given rule set, the sets $\mathbb{DT}^i$ and $\mathbb{NDF}^i$ share facts for base relations only, as these represent definitely true and not definitely false facts at the same time. Hence, it is possible to summarize all facts of $\mathbb{DT}^i$ and $\mathbb{NDF}^i$ within the same fact base. Each individual fixpoint is then computed for a semi-positive database such that traditional fixpoint computation as considered in Section 3.1 can be applied. However, adding the facts for evaluating negative literals to the fact base implies that they are included in each resulting fixpoint as well. These facts have to be eliminated from the resulting fixpoint, if they do not belong to the respective half of the database. Hence, within the algorithm the fixpoints are restricted to either definitely true or not definitely false facts (indicated by $|_{\mathtt{dt}}$ and $|_{\mathtt{ndf}}$). In case of determining $\mathbb{DT}^i$ this eliminates the not definitely false facts and in case of $\mathbb{NDF}^i$ the definitely true facts for all unstratified predicates.

In one of the optimizations proposed for the scheme of alternating fixpoint computation in Algorithm 2 the set DT^{i-1} has been employed for evaluating negative literals while computing the set NDF^i. A similar improvement has been already incorporated in the scheme of AFP materialization. However, since the facts included in $\mathbb{DT}^{i-1}$ refer to definitely true relations (whereas negative body literals in $\mathcal{R}^{\mathtt{ndf}}$ refer to not definitely false relations), the set $\mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^{i-1}))$ is employed including the not definitely false form of each fact in $\mathbb{DT}^{i-1}$ as a basis for computing $\mathbb{NDF}^i$.

In Section 3.3.2 it has been argued that the efficiency of Algorithm 2 can be significantly enhanced by considering a multi-layered rule set which led to the iterated alternating fixpoint computation presented in Algorithm 3. The efficiency of AFP materialization can be improved in a similar way by considering a multi-layered rule set which may consist of stratified as well as unstratified layers. For showing how a layering of the original rule set can be also used for the doubled program transformed rules, we introduce the notion doubled program layering.

Definition 6.6 (Doubled Program Layering) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}}$ the doubled program rewriting of $\mathcal{R}$. The doubled program layering $\lambda^{\mathtt{dp}}$ of λ is a layering of $\mathcal{R}^{\mathtt{dp}}$ such that for all $p \in \mathtt{pred}(\mathcal{D})$ holds that $\lambda^{\mathtt{dp}}(\mathtt{dt}(p)) = \lambda^{\mathtt{dp}}(\mathtt{ndf}(p)) = \lambda(p)$.*

The following lemma shows that the doubled program rewriting of a stratified layer coincides with the original layer.

Lemma 6.1 *Let $\mathcal{R}$ be a deductive rule set, λ a layering on $\mathcal{R}$, and $\mathcal{R}^{\mathtt{dp}}$ the doubled program rewriting of $\mathcal{R}$. Then the rules of $\mathcal{R}^{\mathtt{dp}}$ generated for a stratified layer $\mathcal{R}_l$ of $\mathcal{R}$ are identical with $\mathcal{R}_l$. If $\mathcal{R}_l^{\mathtt{dp}}$ denotes this rule set, the following equations hold in particular:*

1. $\mathcal{R}_l^{\mathtt{ndf}} = \emptyset$
2. $\mathcal{R}_l^{\mathtt{dt}} = \mathcal{R}_l$

Proof: The proposition follows immediately from Definitions 6.1-6.4 and Definition 6.6, as transformations are only performed for unstratified layers. □

We can now define iterated AFP materialization by means of Algorithm 5 for computing the alternating fixpoint on the basis of the doubled program rewriting and layering. This algorithm basically coincides with iterated alternating fixpoint computation as presented in Algorithm 3 and allows a differentiated treatment of stratified and unstratified predicates.

From Lemma 6.1 follows that the doubled program rewriting $\mathcal{R}^{\mathtt{dp}}$ of a stratified rule set $\mathcal{R}$ is identical with $\mathcal{R}$. However, if the rule set $\mathcal{R}$ is unstratified, then there must exist an unstratified layer $\mathcal{R}_l$ of $\mathcal{R}$ which may be divided itself into the rule classes $\mathcal{R}_l^{\circ}$, $\mathcal{R}_l^{\times}$, and $\mathcal{R}_l^{*}$ specifying the hierarchical, the stratified, and the unstratified rules in $\mathcal{R}_l$, respectively. In this case, the computation of definitely true facts can be refined in a similar way as proposed for the scheme of iterated alternating fixpoint computation. During iterated AFP materialization, not definitely false facts are explicitly computed for unstratified layers only. For stratified predicates these facts are implicitly represented by the set of definitely true facts. In particular, the computations performed with respect to stratified layers coincide with traditional fixpoint computation as considered in Section 3.1. The reason is that for stratified layers the rule sets $\mathcal{R}_l^{\mathtt{ndf}}$ and $\mathcal{R}_l^{\mathtt{dt},*}$ are empty, so that only the computation of $\mathbb{DT}_l^0$ may yield facts not already present in the fact base. Note that the 'application' of an empty $\mathcal{R}_l^{\mathtt{ndf}}$ rule set is still necessary in order to correctly compute the not definitely false facts of the stratified layer l, i.e., $\mathbb{NDF}_l := \mathbb{NDF}_l^1$ with $\mathbb{NDF}_l^1 = \mathbb{NDF}_{l-1} \cup \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}_l^0))$. In particular, this equation holds as the used immediate consequence operator is cumulative.

The following Theorem is adopted from [Gri97] and shows the correctness of (iterated) AFP materialization. Note that the proof is omitted and can be found in [Gri97] where instead of doubled program rewriting the notion well-founded rewriting has been used and the `dt` prefix (for differentiating definitely true relations with respect to unstratified relations from stratified ones) is omitted.

Theorem 6.1 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database and λ a layering on $\mathcal{D}$. Then (iterated) AFP materialization always terminates and the sets $\mathbb{DT}$ and $\mathbb{NDF}$ correctly represent the well-founded model of $\mathcal{D}$. It holds that*

$$\mathcal{M}_{\mathcal{D}} = \mathtt{dt}^{-1}(\mathbb{DT}) \cup \neg \cdot \overline{\mathtt{ndf}^{-1}(\mathbb{NDF})}.$$

Proof: cf. [Gri97, p. 118-121]. □

Algorithm 5: Iterated AFP materialization

$\mathbb{DT}_0 := \mathcal{F}$;
$\mathbb{NDF}_0 := \mathcal{F}$;
for each layer $l = 1, \ldots, m$ of $\mathcal{R}^{\mathtt{dp}}$ defined by $\lambda^{\mathtt{dp}}$ **do**
  $i := 0$;
  $\mathbb{DT}_l^0 := \mathtt{lfp}(T^{\star}_{\mathcal{R}_l^{\mathtt{dt},\circ} \cup \mathcal{R}_l^{\mathtt{dt},\times}}, \mathbb{DT}_{l-1} \cup \mathbb{NDF}_{l-1})|_{\mathtt{dt}}$;
  repeat
    $i := i + 1$;
    $\mathbb{NDF}_l^i := \mathtt{lfp}(T^{\star}_{\mathcal{R}_l^{\mathtt{ndf}}}, \mathbb{NDF}_{l-1} \cup \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}_l^{i-1})) \cup \mathbb{DT}_l^{i-1})|_{\mathtt{ndf}}$;
    $\mathbb{DT}_l^i := \mathtt{lfp}(T^{\star}_{\mathcal{R}_l^{\mathtt{dt},\times} \cup \mathcal{R}_l^{\mathtt{dt},*}}, \mathbb{DT}_l^{i-1} \cup \mathbb{NDF}_l^i)|_{\mathtt{dt}}$;
  until $\mathbb{DT}_l^i = \mathbb{DT}_l^{i-1}$;
  $\mathbb{NDF}_l := \mathbb{NDF}_l^i$;
  $\mathbb{DT}_l := \mathbb{DT}_l^i$;
end for
$\mathbb{DT} := \mathbb{DT}_m$;
$\mathbb{NDF} := \mathbb{NDF}_m$;

6.2 Evaluating Doubled Programs

We will now turn our attention to the problem of repeated computations in the algorithms presented for AFP materialization. In order to simplify the following discussion, the non-iterated version is mainly considered. At the end of this section, the results are then transferred to improve the iterated AFP materialization as well.

In Subsection 6.2.1 it is shown how update propagation in general can be used for improving the AFP materialization scheme introduced above. Afterwards, in Subsection 6.2.2 we show how a simplified version of soft update propagation can be employed for improving AFP materialization leading to the so-called soft alternating fixpoint approach. In the following, we will abbreviate the notion doubled program by DP.

6.2.1 DP Materialization Using Update Propagation

A graphical representation of the general course of AFP materialization is presented in Figure 6.2 showing the minor differences in comparison to alternating fixpoint computation as presented in Figure 3.1 in Section 3.3.1. Note that the DT- and NDF-sets presented and the $\mathbb{DT}$- and $\mathbb{NDF}$-sets really computed by the materialization algorithms can again be obtained from each other by applying the `dt` and `ndf` mapping, respectively.

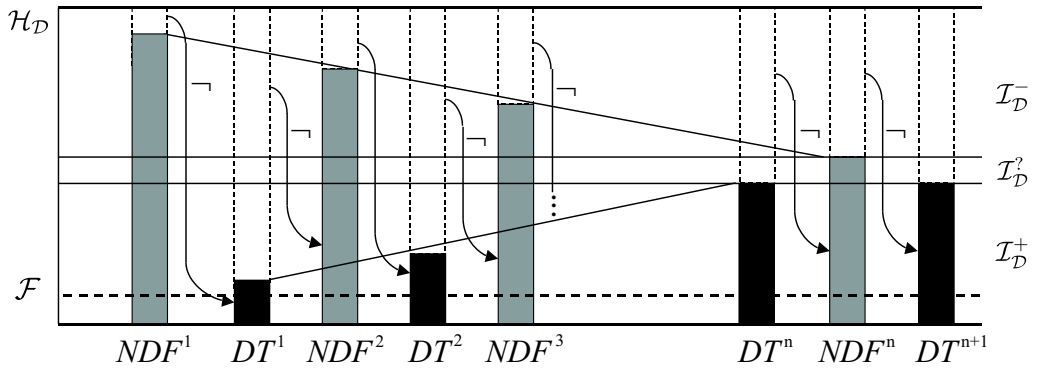


Figure 6.2: AFP materialization.

Several optimizations have been proposed for this scheme of alternating fixpoint materialization (e.g. in [KSS91, KSS95]). However, the problem of repeated computations of facts remained unsolved. Consider again our running example and the corresponding results when applying the scheme in Algorithm 4. Starting from $\mathbb{DT}^0 = \mathcal{F}$, we obtain:

$$\begin{aligned}
\mathbb{NDF}^1 &:= \mathcal{F} \cup \{\texttt{ndf_e(0)}, \texttt{ndf_e(1)}, \texttt{ndf_e(2)}, \texttt{ndf_e(3)}, \texttt{ndf_e(4)}\} \\
\mathbb{DT}^1 &:= \mathcal{F} \cup \{\texttt{dt_e(4)}\} \\
\mathbb{NDF}^2 &:= \mathcal{F} \cup \{\texttt{ndf_e(0)}, \texttt{ndf_e(1)}, \texttt{ndf_e(2)}, \texttt{ndf_e(4)}\} \\
\mathbb{DT}^2 &:= \mathcal{F} \cup \{\texttt{dt_e(2)}, \texttt{dt_e(4)}\} \\
\mathbb{NDF}^3 &:= \mathcal{F} \cup \{\texttt{ndf_e(0)}, \texttt{ndf_e(2)}, \texttt{ndf_e(4)}\} \\
\mathbb{DT}^3 &:= \mathcal{F} \cup \{\texttt{dt_e(0)}, \texttt{dt_e(2)}, \texttt{dt_e(4)}\} \\
\mathbb{NDF}^4 &:= \mathcal{F} \cup \{\texttt{ndf_e(0)}, \texttt{ndf_e(2)}, \texttt{ndf_e(4)}\} \\
\mathbb{DT}^4 &:= \mathcal{F} \cup \{\texttt{dt_e(0)}, \texttt{dt_e(2)}, \texttt{dt_e(4)}\}
\end{aligned}$$

In each phase many facts from previous iteration rounds are repeatedly computed, e.g. all definitely true facts. The changes to the sets of definitely true and not definitely false facts, however, are caused only by the changes of the other sets computed before, respectively. Since $\mathbb{DT}$-facts as well as $\mathbb{NDF}$-facts represent base facts for the other half, it seems to be useful to compute the changes of the $\mathbb{DT}$-facts and $\mathbb{NDF}$-facts only. This can be achieved by means of update propagation rules for true updates that explicitly refer to the given changes of the base facts. Since the set of $\mathbb{DT}$-facts monotonically increases, it is sufficient to consider propagation rules for induced insertions only, whereas for the monotonically decreasing set of $\mathbb{NDF}$-facts, propagation rules for induced deletions have to be considered only.

Suppose the delta sets Δ_i^+ represent induced insertions with respect to $\mathbb{DT}$-relations and Δ_i^- represent induced deletions with respect to $\mathbb{NDF}$-relations. A graphical representation of AFP materialization based on these delta sets is presented in Figure 6.3. Note that the calculation of each delta set (except for the set

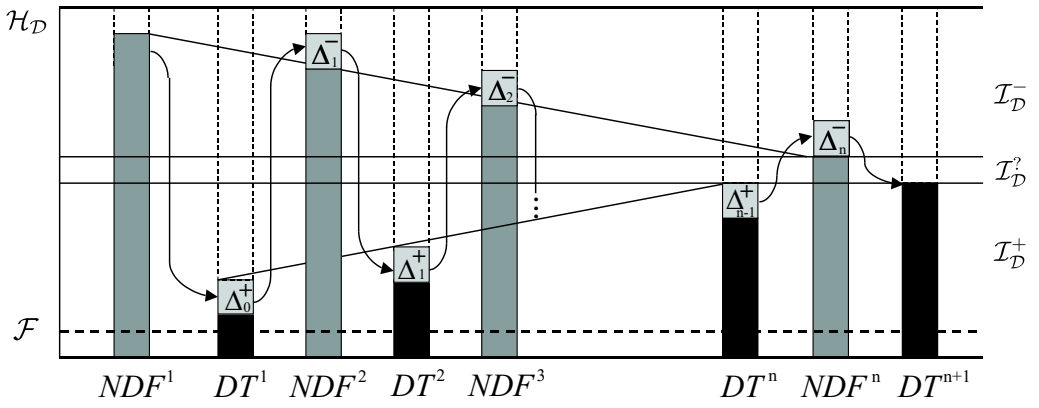


Figure 6.3: AFP materialization using 'delta' sets.

Δ_0^+ as starting point) positively depends on the complement delta set computed in the previous iteration round.

In principle, propagation and transition rules as proposed in Chapter 5 could be used for defining these delta sets. However, update propagation in this context represents a special case as certain combinations of base and induced updates occur, only. Therefore, specific $\mathbb{DT}$- and $\mathbb{NDF}$-propagation rules as well as special $\mathbb{DT}$- and $\mathbb{NDF}$-transition rules are defined for computing induced insertions according to $\mathbb{DT}$-relations and induced deletions according to $\mathbb{NDF}$-relations, respectively.

As stated before, references to both the old as well as the new state are necessary for computing true updates. As proposed in Chapter 5, we will now define propagation and transition rules assuming that the old state is present and the new one is simulated. For the algorithms to come, however, it turned out to be quite useful to consider certain combinations of states in order to get a smaller set of propagation rules. Therefore, in the following we assume the old $\mathbb{DT}$-, the new $\mathbb{NDF}$- and $\Delta^-\mathbb{NDF}$-facts to be present when propagation rules for computing insertions for $\mathbb{DT}$-facts are considered, whereas the old $\mathbb{DT}$-, the old $\mathbb{NDF}$- and $\Delta^+\mathbb{DT}$-facts are present when propagation rules for deletions from the $\mathbb{NDF}$ set are evaluated. According to these conditions we introduce the DT new form and the NDF new form for specifying the `new` mapping in propagation rules for $\Delta^+\mathbb{DT}$-facts and $\Delta^-\mathbb{NDF}$-facts, respectively.

Definition 6.7 (DT New Form) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ the doubled program rewriting of $\mathcal{R}$. The mapping $\mathtt{new}_{\mathtt{dt}}$ assigns to each literal L with $\mathtt{pred}(L) \in \mathtt{pred}(\mathcal{F} \cup \mathcal{R}^{\mathtt{dp}})$ its new DT form such that*

1. *If L is a positive literal, then*

$$\texttt{new}_{\texttt{dt}}(L) := \begin{cases} p(\vec{x}) & \text{if } L \equiv p(\vec{x}) \text{ and } p \in \texttt{pred}(\mathcal{D}) \\ & \text{is stratified} \\ dt\_p^{new}(\vec{x}) & \text{if } L \equiv dt\_p(\vec{x}) \text{ and } p \in \texttt{pred}(\mathcal{D}) \\ & \text{is unstratified.} \end{cases}$$

2. *If L is a negative literal, then*

$$\texttt{new}_{\texttt{dt}}(L) := \begin{cases} \neg p(\vec{x}) & \text{if } L \equiv \neg p(\vec{x}) \text{ and } p \in \texttt{pred}(\mathcal{D}) \\ & \text{is stratified} \\ \neg ndf\_p(\vec{x}) & \text{if } L \equiv \neg ndf\_p(\vec{x}) \text{ and } p \in \texttt{pred}(\mathcal{D}) \\ & \text{is unstratified.} \end{cases}$$

3. *The mapping $\texttt{new}_{\texttt{dt}}$ may be applied to conjunctions of literals as follows:*

$$\texttt{new}_{\texttt{dt}}(L_1 \wedge \ldots \wedge L_n) := \bigwedge_{1 \leq i \leq n} \texttt{new}_{\texttt{dt}}(L_i).$$

Definition 6.8 (NDF New Form) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\texttt{dp}} = \mathcal{R}^{\texttt{dt}} \cup \mathcal{R}^{\texttt{ndf}}$ the doubled program rewriting of $\mathcal{R}$. The mapping $\texttt{new}_{\texttt{ndf}}$ assigns to each literal L with $\texttt{pred}(L) \in \texttt{pred}(\mathcal{F} \cup \mathcal{R}^{\texttt{dp}})$ its new NDF form such that*

1. *If L is a positive literal, then*

$$\texttt{new}_{\texttt{ndf}}(L) := \begin{cases} p(\vec{x}) & \text{if } L \equiv p(\vec{x}) \text{ and } p \in \texttt{pred}(\mathcal{D}) \\ & \text{is stratified} \\ ndf\_p^{new}(\vec{x}) & \text{if } L \equiv ndf\_p(\vec{x}) \text{ and } p \in \texttt{pred}(\mathcal{D}) \\ & \text{is unstratified.} \end{cases}$$

2. *If L is a negative literal, then*

$$\texttt{new}_{\texttt{ndf}}(L) := \begin{cases} \neg p(\vec{x}) & \text{if } L \equiv \neg p(\vec{x}) \text{ and } p \in \texttt{pred}(\mathcal{D}) \\ & \text{is stratified} \\ \neg dt\_p^{new}(\vec{x}) & \text{if } L \equiv \neg dt\_p(\vec{x}) \text{ and } p \in \texttt{pred}(\mathcal{D}) \\ & \text{is unstratified.} \end{cases}$$

3. *The mapping $\texttt{new}_{\texttt{ndf}}$ may be applied to conjunctions of literals as follows:*

$$\texttt{new}_{\texttt{ndf}}(L_1 \wedge \ldots \wedge L_n) := \bigwedge_{1 \leq i \leq n} \texttt{new}_{\texttt{ndf}}(L_i).$$

Note that we assume the new state of NDF relations to be present when evaluating $\mathcal{R}^{\mathtt{dt}}$ such that the application of $\mathtt{new}_{\mathtt{dt}}(L)$ with $L \equiv \neg ndf\_p(\vec{x})$ does not add a *new* suffix to the relation name ndf_p. In contrast to this, unstratified DT relations which are negatively referenced in $\mathcal{R}^{\mathtt{ndf}}$ are modified by the mapping $\mathtt{new}_{\mathtt{ndf}}$ such that the application of $\mathtt{new}_{\mathtt{ndf}}(L)$ with $L \equiv \neg dt\_p(\vec{x})$ yields $\neg dt\_p^{new}(\vec{x})$. This is because we assume the old state of definitely true relations to be present when evaluating $\mathcal{R}^{\mathtt{ndf}}$ whereas the new one has to be simulated.

Based on the DT new form we can now define DT-propagation rules and DT-transition rules for defining induced insertions into DT relations.

Definition 6.9 (DT-Propagation Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ the doubled program rewriting of $\mathcal{R}$ where $\mathcal{R}^{\mathtt{dt}}$ denotes the rules for defining definitely true relations and $\mathcal{R}^{\mathtt{ndf}}$ denotes the rules for defining not definitely false relations. The set of DT-propagation rules for true insertions with respect to $\mathcal{R}^{\mathtt{dt}}$ is denoted $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ and is defined as follows:*

1. *For each rule $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^{\mathtt{dt}}$ with $A \equiv dt\_p(\vec{x})$ and each negative body literal $L_i \equiv \neg ndf\_q(\vec{y})$ where $p, q \in \mathtt{pred}(\mathcal{R})$ are unstratified predicates, a propagation rule of the form*

$$A^+ \leftarrow \Delta^- ndf\_q(\vec{y}) \wedge \mathtt{new}_{\mathtt{dt}}(L_1 \wedge \ldots \wedge L_{i-1} \wedge L_{i+1} \wedge \ldots \wedge L_n) \wedge \neg A$$

   is in $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$.

2. *For each rule $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^{\mathtt{dt}}$ with $A \equiv dt\_p(\vec{x})$ and each positive derived body literal $L_j \equiv dt\_r(\vec{z})$ where $p, r \in \mathtt{pred}(\mathcal{R})$ are unstratified predicates, a propagation rule of the form*

$$A^+ \leftarrow L_j^+ \wedge \mathtt{new}_{\mathtt{dt}}(L_1 \wedge \ldots \wedge L_{j-1} \wedge L_{j+1} \wedge \ldots \wedge L_n) \wedge \neg A$$

   is in $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$.

3. *No other rules are in $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$.*

In Chapter 5 we introduced the superscripts "$+$" and "$-$" for transforming a literal $A \equiv p(\vec{x})$ to its dynamic form, i.e., $A^+ \equiv \Delta^+ p(\vec{x})$ and $A^- \equiv \Delta^- p(\vec{x})$. We will now use the notions $+ \cdot A$ or $- \cdot A$ to refer to A^+ or A^-, respectively. This concatenation may also be applied to a set of ground atoms and is simply used to transform sets of $\mathbb{DT}$-and $\mathbb{NDF}$-facts into their dynamic form. Additionally, $+^{-1} \cdot S$ and $-^{-1} \cdot S$ is used to denote the transformation of dynamic literals in the set S back to their original form.

Proposition 6.10 (Correctness of DT-Propagation Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ the doubled program rewriting of $\mathcal{R}$ where $\mathcal{R}^{\mathtt{dt}}$ denotes the rules for defining definitely true relations. Let $\mathbb{DT}^i$ and $\mathbb{NDF}^i$ with $i \geq 1$ be the sets of definitely true facts, respectively the sets of not definitely false facts computed in the i-th iteration round of AFP materialization. Then the delta relations in $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ correctly represent the state transition $\mathbb{DT}^{i+1} \setminus \mathbb{DT}^i$. In particular, for each unstratified relation $p \in \mathtt{pred}(\mathcal{D})$ the following holds:*

$$\Delta^{+}dt\_p(\vec{t}\,) \in \Delta^{+}\mathbb{DT}^i \iff \Delta^{+}dt\_p(\vec{t}\,) \in \mathtt{lfp}(T^{\star}_{\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})}, \mathbb{NDF}^{i+1} \cup \mathbb{DT}^i \cup \Delta^{-}\mathbb{NDF}^i)$$

where $\Delta^{+}\mathbb{DT}^i := + \cdot [\mathbb{DT}^{i+1} \setminus \mathbb{DT}^i]$ and $\Delta^{-}\mathbb{NDF}^i := - \cdot [\mathbb{NDF}^i \setminus \mathbb{NDF}^{i+1}]$.

Proof (Sketch): In principle, the proof of this proposition coincides with the one performed for Proposition 5.3 in which the correctness of propagation rules for true updates has been shown. This time, the propagation rules are used to describe the transition from the deductive database

$$\mathcal{D}^i = \langle \mathbb{DT}^{i-1} \cup \mathbb{NDF}^i, \mathcal{R}^{\mathtt{dt}} \rangle$$

with $\mathcal{M}_{\mathcal{D}^i}|_{\mathtt{dt}} = \mathbb{DT}^i$ to the deductive database

$$\mathcal{D}^{i+1} = \langle \mathbb{DT}^i \cup \mathbb{NDF}^{i+1}, \mathcal{R}^{\mathtt{dt}} \rangle$$

with $\mathcal{M}_{\mathcal{D}^{i+1}}|_{\mathtt{dt}} = \mathbb{DT}^{i+1}$ due to the application of the update $u_{\mathcal{D}^i} = \langle \emptyset, u^{-}_{\mathcal{D}^i} \rangle$ with $u^{-}_{\mathcal{D}^i} = \mathbb{NDF}^i \setminus \mathbb{NDF}^{i+1}$. The augmented deductive database $\mathcal{D}^p$ with respect to $\mathcal{D}^i$ and $u_{\mathcal{D}^i}$ is given by

$$\mathcal{D}^p = \langle \mathbb{DT}^{i-1} \cup \mathbb{NDF}^i \cup \mathtt{prop\_seeds}(u_{\mathcal{D}^i}), \mathcal{R}^{\mathtt{dt}} \cup \varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}}) \rangle$$

with $\mathtt{prop\_seeds}(u_{\mathcal{D}^i}) = \Delta^{-}\mathbb{NDF}^i$. According to our proposed state simulation, the augmented database $\mathcal{D}^p$ is slightly modified by adding the old state of DT-relations and the new state of NDF-relations with the sets $\mathbb{DT}^i$ respectively $\mathbb{NDF}^{i+1}$ to its fact base. The resulting modified augmented database $\mathcal{D}^{p'}$ is given by

$$\mathcal{D}^{p'} = \langle \mathbb{DT}^i \cup \mathbb{NDF}^{i+1} \cup \Delta^{-}\mathbb{NDF}^i, \varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}}) \rangle$$

where the propagation seeds are applied and the rule set $\mathcal{R}^{\mathtt{dt}}$ for computing the old state is omitted. The only references to the old state of $\mathcal{D}^i$ occur in the effectiveness tests of rules in $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ which has not been made explicit in the definition of DT-propagation rules by using the meta predicate `old`. This is because the modified augmented database $\mathcal{D}^{p'}$ contains with the set $\mathbb{DT}^i$ now the old state of $\mathbb{DT}$-relations in $\mathcal{D}^i$ completely such that the effectiveness tests negatively refer to base relations, only, and thus, are evaluated correctly. Using Proposition 5.3 and assuming the correct evaluation of the meta predicate $\mathtt{new}_{\mathtt{dt}}$, it can be followed that the condition

$$dt\_p(\vec{t}\,) \in u^{+}_{\mathcal{D}^i \to \mathcal{D}^{i+1}} \iff \Delta^{+} dt\_p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^{p'}}$$

must hold. Using the equivalence

$$dt\_p(\vec{t}\,) \in u^{+}_{\mathcal{D}^i \to \mathcal{D}^{i+1}} \iff \Delta^{+} dt\_p(\vec{t}\,) \in \Delta^{+}\mathbb{DT}^i$$

which follows from the prerequisites of our proposition and using the equation

$$\mathcal{M}_{\langle \varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}}), \mathbb{DT}^i \cup \mathbb{NDF}^{i+1} \cup \Delta^{-}\mathbb{NDF}^i \rangle} = \mathtt{lfp}(T^{\star}_{\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})}, \mathbb{NDF}^{i+1} \cup \mathbb{DT}^i \cup \Delta^{-}\mathbb{NDF}^i)$$

which follows from the fact that $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ is semi-positive, the correctness of the proposition has been shown. □

Simulating the new state as in our approach requires the definition of transition rules for $\mathbb{DT}$-relations that are positively referenced in a rule's body in $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$. As the set of $\mathbb{DT}$-facts is monotonically increasing, we do not have to consider deletions within the *DT*-transition rules. Therefore, the new state of $\mathbb{DT}$-relations is simply the union of computed insertions and (old) facts already stored in the database.

Definition 6.11 (DT-Transition Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ the doubled program rewriting of $\mathcal{R}$ with $\mathcal{R}^{\mathtt{dt}}$ denoting the rules for defining definitely true relations. Then the set of DT-transition rules for new state simulation with respect to $\mathcal{R}^{\mathtt{dt}}$ is denoted $\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ and is defined as follows:*

1. *For each n-ary derived predicate symbol $dt\_p \in \mathtt{pred}(\mathcal{R}^{\mathtt{dt}})$ with $p \in \mathtt{pred}(\mathcal{R})$ and p is an unstratified relation, the transition rules*

$$dt\_p^{new}(x_1, \ldots, x_n) \leftarrow dt\_p(x_1, \ldots, x_n)$$
$$dt\_p^{new}(x_1, \ldots, x_n) \leftarrow \Delta^{+} dt\_p(x_1, \ldots, x_n),$$

   are in $\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ where the x_i $(i = 1, \ldots, n)$ are distinct variables.

2. *No other rules are in $\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$.*

Proposition 6.12 (Correctness of DT-Transition Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ the doubled program rewriting of $\mathcal{R}$ where $\mathcal{R}^{\mathtt{dt}}$ denotes the rules for defining definitely true relations. Let $\mathbb{DT}^i$ and $\mathbb{NDF}^i$ with $i \geq 1$ be the sets of definitely true facts, respectively the sets of not definitely false facts computed in the i-th iteration round of AFP materialization. Then the transition rules $\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ correctly represent the new state of DT-relations in $\mathbb{DT}^i$ with respect to $\mathbb{DT}^{i+1}$. In particular, for each unstratified relation $p \in \mathtt{pred}(\mathcal{D})$ the following holds:*

$$
\begin{aligned}
& dt\_p(\vec{t}\,) \in \mathbb{DT}^{i+1} \\
\Longleftrightarrow \\
& dt\_p^{new}(\vec{t}\,) \in \mathtt{lfp}(T^{\star}_{\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})\cup\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})}, \mathbb{NDF}^{i+1} \cup \mathbb{DT}^{i} \cup \Delta^{-}\mathbb{NDF}^{i})
\end{aligned}
$$

where $\Delta^{-}\mathbb{NDF}^{i} := - \cdot [\mathbb{NDF}^{i} \setminus \mathbb{NDF}^{i+1}]$.

Proof: Correctness follows immediately from Definitions 6.7, 6.9 and 6.11 and from Proposition 6.10. Note that for each unstratified predicate $p \in \mathtt{pred}(\mathcal{D})$ the new state of the corresponding DT-relation dt_p^{new} is to be simulated whereas the new state of the corresponding NDF-relation ndf_p^{new} is already given by the set $\mathbb{NDF}^{i+1}$. The new state of stratified relations, however, coincides with their old state. Therefore, these relations remain unchanged when applying the $\mathtt{new}_{\mathtt{dt}}$ mapping and are correctly represented with the included set $\mathbb{DT}^{i}$, as well. □

Similar to the definitions above, we now define NDF-propagation rules and NDF-transition rules for defining induced deletions from $\mathbb{NDF}$-relations.

Definition 6.13 (NDF-Propagation Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ the doubled program rewriting of $\mathcal{R}$ where $\mathcal{R}^{\mathtt{dt}}$ denotes the rules for defining definitely true relations and $\mathcal{R}^{\mathtt{ndf}}$ denotes the rules for defining not definitely false relations. The set of NDF-propagation rules for true deletions with respect to $\mathcal{R}^{\mathtt{ndf}}$ is denoted $\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ and is defined as follows:*

1. *For each rule $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^{\mathtt{ndf}}$ with $A \equiv ndf\_p(\vec{x})$ and each negative body literal $L_i \equiv \neg dt\_q(\vec{y})$ with $p, q \in \mathtt{pred}(\mathcal{R})$ and p,q are unstratified predicates, a propagation rule of the form*

   $$A^{-} \leftarrow \Delta^{+} dt\_q(\vec{y}) \wedge L_1 \wedge \ldots \wedge L_{i-1} \wedge L_{i+1} \wedge \ldots \wedge L_n \wedge \neg \mathtt{new}_{\mathtt{ndf}}(A)$$

   is in $\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$.

2. *For each rule $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^{\mathtt{ndf}}$ with $A \equiv ndf\_p(\vec{x})$ and each positive derived body literal $L_j \equiv ndf\_r(\vec{z})$ with $p, r \in \mathtt{pred}(\mathcal{R})$ and p,r are unstratified predicates, a propagation rule of the form*

   $$A^{-} \leftarrow L_j^{-} \wedge L_1 \wedge \ldots \wedge L_{j-1} \wedge L_{j+1} \wedge \ldots \wedge L_n \wedge \neg \mathtt{new}_{\mathtt{ndf}}(A)$$

   is in $\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$.

3. *No other rules are in $\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$.*

Proposition 6.14 (Correctness of NDF-Propagation Rules) *Let* $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ *be a deductive database,* λ *a layering on* $\mathcal{R}$ *and* $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ *the doubled program rewriting of* $\mathcal{R}$ *where* $\mathcal{R}^{\mathtt{ndf}}$ *denotes the rules for defining not definitely false relations. Let* $\mathbb{DT}^i$ *and* $\mathbb{NDF}^i$ *with* $i \geq 1$ *be the sets of definitely true facts, respectively the sets of not definitely false facts computed in the i-th iteration round of AFP materialization. Then the delta relations in* $\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ *correctly represent the state transition* $\mathbb{NDF}^i \setminus \mathbb{NDF}^{i+1}$. *In particular, for each unstratified relation* $p \in \mathtt{pred}(\mathcal{D})$ *the following holds*

$$\begin{aligned} & \Delta^{-} ndf\_p(\vec{t}\,) \in \Delta^{-}\mathbb{NDF}^i \\ \Longleftrightarrow \\ & \Delta^{-} ndf\_p(\vec{t}\,) \in \mathtt{lfp}(T^{\star}_{\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})}, \mathbb{NDF}^i \cup \mathbb{DT}^{i-1} \cup \Delta^{+}\mathbb{DT}^{i-1}) \end{aligned}$$

where $\Delta^{-}\mathbb{NDF}^i := - \cdot [\mathbb{NDF}^i \setminus \mathbb{NDF}^{i+1}]$ *and* $\Delta^{+}\mathbb{DT}^{i-1} := + \cdot [\mathbb{DT}^i \setminus \mathbb{DT}^{i-1}]$.

Proof (Sketch): The correctness of this proposition is shown by reducing it to the case of propagating true deletions. NDF-propagation rules are used to describe the transition from the deductive database

$$\mathcal{D}^i = \langle \mathbb{DT}^{\mathtt{i-1}} \cup \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^{i-1})), \mathcal{R}^{\mathtt{ndf}} \rangle$$

with $\mathcal{M}_{\mathcal{D}^i}|_{\mathtt{ndf}} = \mathbb{NDF}^i$ to the deductive database

$$\mathcal{D}^{i+1} = \langle \mathbb{DT}^{i} \cup \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^{i})), \mathcal{R}^{\mathtt{ndf}} \rangle$$

with $\mathcal{M}_{\mathcal{D}^{i+1}}|_{\mathtt{ndf}} = \mathbb{NDF}^{i+1}$ due to the application of the update $u_{\mathcal{D}^i} = \langle u^{+}_{\mathcal{D}^i}, \emptyset \rangle$ with $u^{+}_{\mathcal{D}^i} = \mathbb{DT}^i \setminus \mathbb{DT}^{i-1}$. The augmented deductive database $\mathcal{D}^p$ with respect to $\mathcal{D}^i$ and $u_{\mathcal{D}^i}$ is given by

$$\mathcal{D}^p = \langle \mathbb{DT}^{i-1} \cup \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^{i-1})) \cup \mathtt{prop\_seeds}(u_{\mathcal{D}^i}), \mathcal{R}^{\mathtt{ndf}} \cup \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \rangle$$

with $\mathtt{prop\_seeds}(u_{\mathcal{D}^i}) = \Delta^{+}\mathbb{DT}^{i-1}$. According to our proposed state simulation, we will again modify the augmented database $\mathcal{D}^p$ by adding the old state of NDF-relations in $\mathcal{D}^i$ with the set $\mathbb{NDF}^i$ to its fact base. The resulting modified augmented database $\mathcal{D}^{p'}$ is given by

$$\mathcal{D}^{p'} = \langle \mathbb{NDF}^i \cup \mathbb{DT}^{i-1} \cup \Delta^{+}\mathbb{DT}^{i-1}, \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \rangle$$

where the propagation seeds are applied and the rule set $\mathcal{R}^{\mathtt{ndf}}$ for computing the old state of NDF-relations is omitted. In addition, the set $\mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^{i-1}))$ is left out since it is included in set of old NDF-relations, i.e., $\mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^{i-1})) \subseteq \mathbb{NDF}^i$. The modified augmented database $\mathcal{D}^{p'}$ contains with the set $\mathbb{DT}^{i-1}$ and $\mathbb{NDF}^i$ now the old state of DT-relations as well as NDF-relations such that the derivability tests (except from dynamic literals for induced deletions) in NDF-propagation rules solely refer to base relations and thus, are evaluated correctly. Using Proposition 5.3 and assuming the correct evaluation of the meta predicate $\mathtt{new}_{\mathtt{ndf}}$, it can be followed that the condition

$$ndf\_p(\vec{t}\,) \in u^{-}_{\mathcal{D}^i \to \mathcal{D}^{i+1}} \iff \Delta^{-} ndf\_p(\vec{t}\,) \in \mathcal{M}_{\mathcal{D}^{p\prime}}$$

must hold. Using the equivalence

$$ndf\_p(\vec{t}\,) \in u^{-}_{\mathcal{D}^i \to \mathcal{D}^{i+1}} \iff \Delta^{-} ndf\_p(\vec{t}\,) \in \Delta^{-}\mathbb{NDF}^i$$

which follows from the prerequisites of our proposition and using the equation

$$\mathcal{M}_{\langle \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}), \mathbb{NDF}^i \cup \mathbb{DT}^{i-1} \cup \Delta^{+}\mathbb{DT}^{i-1} \rangle} = \mathtt{lfp}(T^{\star}_{\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})}, \mathbb{NDF}^i \cup \mathbb{DT}^{i-1} \cup \Delta^{+}\mathbb{DT}^{i-1})$$

which follows from the fact that $\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ is semi-positive, the correctness of the proposition has been shown. □

As the effectiveness test of NDF-propagation rules refers to the new state of not definitely false relations, we have to consider again transition rules for new state simulation.

Definition 6.15 (NDF-Transition Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ the doubled program rewriting of $\mathcal{R}$ where $\mathcal{R}^{\mathtt{dt}}$ denotes the rules for defining definitely true relations and $\mathcal{R}^{\mathtt{ndf}}$ denotes the rules for defining not definitely false relations. Then the set of NDF-transition rules for new state simulation with respect to $\mathcal{R}^{\mathtt{ndf}}$ is denoted $\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ and is defined as follows:*

1. *For each rule $A \leftarrow L_1 \wedge \ldots \wedge L_n \in \mathcal{R}^{\mathtt{ndf}}$ with $A \equiv ndf\_p(\vec{x})$ and $p \in \mathtt{pred}(\mathcal{R})$ is an unstratified predicate, a transition rule of the form*

   $$\mathtt{new}_{\mathtt{ndf}}(A) \leftarrow \mathtt{new}_{\mathtt{ndf}}(L_1, \ldots L_n)$$

   is in $\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$.

2. *For each negative literal $L \equiv \neg dt\_p(x_1, \ldots, x_n)$ occurring in $\mathcal{R}^{\mathtt{ndf}}$ with $p \in \mathtt{pred}(\mathcal{R})$ and p is an unstratified relation, two DT-transition rules*

   $$dt\_p^{new}(x_1, \ldots, x_n) \leftarrow dt\_p(x_1, \ldots, x_n)$$
   $$dt\_p^{new}(x_1, \ldots, x_n) \leftarrow \Delta^{+} dt\_p(x_1, \ldots, x_n),$$

   are in $\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ where the x_i $(i = 1, \ldots, n)$ are distinct variables.

3. *No other rules are in $\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$.*

Note that NDF-propagation rules have references to the new state as well as to the induced insertions of definitely true relations. Additionally, NDF-transition rules refer to the new state of definitely true relations. Therefore, DT-transition rules have to be additionally included in the set of NDF-transition rules in order to make the rule sets $\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \cup \tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ complete.

Proposition 6.16 (Correctness of NDF-Transition Rules) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, λ a layering on $\mathcal{R}$ and $\mathcal{R}^{\mathtt{dp}} = \mathcal{R}^{\mathtt{dt}} \cup \mathcal{R}^{\mathtt{ndf}}$ the doubled program rewriting of $\mathcal{R}$ where $\mathcal{R}^{\mathtt{ndf}}$ denotes the rules for defining not definitely false relations. Let $\mathbb{DT}^i$ and $\mathbb{NDF}^i$ with $i \geq 1$ be the sets of definitely true facts, respectively the sets of not definitely false facts computed in the i-th iteration round of AFP materialization. Then the transition rules $\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ correctly represent the new state of NDF-relations in $\mathbb{NDF}^i$ with respect to $\mathbb{NDF}^{i+1}$. In particular, for each unstratified relation $p \in \mathtt{pred}(\mathcal{D})$ the following holds*

$$\begin{array}{l} \quad ndf\_p(\vec{t}\,) \in \mathbb{NDF}^{i+1} \\ \Longleftrightarrow \\ \quad ndf\_p^{new}(\vec{t}\,) \in \mathcal{IF}_{\langle \mathbb{NDF}^i \cup \mathbb{DT}^{i-1} \cup \Delta^+\mathbb{DT}^{i-1}, \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \cup \tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \rangle} \end{array}$$

where $\Delta^+\mathbb{DT}^{i-1} := + \cdot [\mathbb{DT}^i \setminus \mathbb{DT}^{i-1}]$.

Proof: Correctness follows immediately from Definitions 6.8, 6.13 and 6.15 and from Propositions 5.5 and 6.14. Since no new state facts are included for any unstratified predicate $p \in \mathtt{pred}(\mathcal{D})$ the new state of the corresponding DT-relation dt_p^{new} and the new state of the corresponding NDF-relation ndf_p^{new} have to be simulated. As the new state of stratified relations is identical with their old state, these relations remain unchanged and are correctly represented with the included set $\mathbb{DT}^{i-1}$. The union of NDF-propagation rules and NDF-transition rules yields a stratifiable rule set such that the iterated fixpoint computation is needed for their correct evaluation. □

Before integrating propagation rules into the AFP materialization scheme, we still have to introduce one more notion. We will use a slightly modified version of the dt- and ndf-restriction from Section 6.1 in oder to access corresponding dynamic literals within sets resulting from a fixpoint computation.

Definition 6.17 ($\mathtt{dt}^+$- and $\mathtt{ndf}^-$-Restriction) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, $\mathcal{D}^{\mathtt{dp}} = \langle \mathcal{F}, \mathcal{R}^{\mathtt{dp}} \rangle$ the deductive database derived from $\mathcal{D}$ by applying the doubled program rewriting to $\mathcal{R}$, and $\mathcal{H}_{\mathcal{D}}$, $\mathcal{H}_{\mathcal{D}^{\mathtt{dp}}}$ the Herbrand base of $\mathcal{D}$, $\mathcal{D}^{\mathtt{dp}}$, respectively. For a set of ground atoms $\mathcal{I} \subseteq \mathcal{H}_{\langle F, \varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}}) \cup \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \rangle}$ we define:*

$$\begin{array}{lcl} \mathcal{I}|_{\mathtt{dt}^+} & := & \{ + \cdot \mathtt{dt}(A) \in \mathcal{I} \mid A \in \mathcal{H}_{\mathcal{D}} \text{ and } \mathtt{dt}(A) \in \mathcal{H}_{\mathcal{D}^{\mathtt{dp}}} \} \\ \mathcal{I}|_{\mathtt{ndf}^-} & := & \{ - \cdot \mathtt{ndf}(A) \in \mathcal{I} \mid A \in \mathcal{H}_{\mathcal{D}} \text{ and } \mathtt{ndf}(A) \in \mathcal{H}_{\mathcal{D}^{\mathtt{dp}}} \}. \end{array}$$

The modified algorithm for computing the alternating fixpoint based on calculating updates is presented in Algorithm 6. The essential difference to the AFP materialization procedure is that the algorithm starts with sets of $\mathbb{DT}$- and $\mathbb{NDF}$-facts which will be updated only by new $\mathbb{DT}$-facts to be added and $\mathbb{NDF}$-facts to be removed within each iteration round until no more new $\mathbb{DT}$-facts can be derived, i.e., $\Delta^+\mathbb{DT}^i = \emptyset$. The expensive evaluation of rules with respect to the

Algorithm 6: AFP materialization using update propagation

$i := 0;$
$\mathbb{DT}^0 := \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{dt},\circ} \cup \mathcal{R}^{\mathtt{dt},\times}}, \mathcal{F})|_{\mathtt{dt}};$
$\mathbb{NDF}^1 := \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{ndf}}}, \mathbb{DT}^0 \cup \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^0)))|_{\mathtt{ndf}};$
$\Delta^+\mathbb{DT}^0 := + \cdot [\mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{dt},\times} \cup \mathcal{R}^{\mathtt{dt},*}}, \mathbb{DT}^0 \cup \mathbb{NDF}^1)|_{\mathtt{dt}} \setminus \mathbb{DT}^0];$
while $\Delta^+\mathbb{DT}^i \neq \emptyset$ **do**
  $i := i + 1;$
  $\Delta^-\mathbb{NDF}^i := \mathcal{IF}_{\langle \mathbb{NDF}^i \cup \mathbb{DT}^{i-1} \cup \Delta^+\mathbb{DT}^{i-1}, \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \cup \tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \rangle}|_{\mathtt{ndf}^-};$
  $\mathbb{DT}^i := \mathbb{DT}^{i-1} \cup +^{-1} \cdot (\Delta^+\mathbb{DT}^{i-1});$
  $\mathbb{NDF}^{i+1} := \mathbb{NDF}^i \setminus -^{-1} \cdot (\Delta^-\mathbb{NDF}^i);$
  $\Delta^+\mathbb{DT}^i := \mathtt{lfp}(T^\star_{\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}}) \cup \tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})}, \mathbb{NDF}^{i+1} \cup \mathbb{DT}^i \cup \Delta^-\mathbb{NDF}^i)|_{\mathtt{dt}^+};$
end while
$\mathbb{NDF} := \mathbb{NDF}^{i+1};$
$\mathbb{DT} := \mathbb{DT}^i;$

underlying database is restricted to the calculation of the smaller set of induced updates. The following theorem shows the correctness of AFP materialization using update propagation as the computed sets of $\mathbb{DT}^i$ and $\mathbb{NDF}^i$ (except from the omitted set $\mathbb{NDF}^0$) coincide with the intermediate results obtained by using AFP materialization.

Theorem 6.2 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database and λ a layering on $\mathcal{D}$. Then AFP materialization using update propagation always terminates and the sets $\mathbb{DT}$ and $\mathbb{NDF}$ correctly represent the well-founded model of $\mathcal{D}$. It holds that*

$$\mathcal{M}_\mathcal{D} = \mathtt{dt}^{-1}(\mathbb{DT}) \cup \neg \cdot \overline{\mathtt{ndf}^{-1}(\mathbb{NDF})}.$$

Proof: The proposition of this theorem is shown by induction on the number of iteration rounds i. In the following, for any stage i we use $\mathbb{DT}^i_{\mathtt{APUP}}$, respectively $\mathbb{NDF}^i_{\mathtt{APUP}}$, for referring to the intermediate results obtained by AFP materialization using update propagation while the sets $\mathbb{DT}^i_{\mathtt{AP}}$, respectively $\mathbb{NDF}^i_{\mathtt{AP}}$, are employed for describing the intermediate results of AFP materialization. From Theorem 6.1 follows that AFP materialization as presented with Algorithm 4 correctly computes the well-founded model of $\mathcal{D}$. The idea is to show that the intermediate results obtained in both algorithms are identical such that AFP materialization using update propagation must be correct as well.

Suppose that $i = 1$: For the sets $\mathbb{DT}^1_{\mathtt{AP}}$ and $\mathbb{NDF}^1_{\mathtt{AP}}$ computed with AFP materialization the following equations hold

$$\mathbb{NDF}^1_{\mathtt{AP}} := \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{ndf}}}, \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^0_{\mathtt{AP}})) \cup \mathbb{DT}^0_{\mathtt{AP}})|_{\mathtt{ndf}}$$
$$\mathbb{DT}^1_{\mathtt{AP}} := \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{dt},\times} \cup \mathcal{R}^{\mathtt{dt},*}}, \mathbb{DT}^0_{\mathtt{AP}} \cup \mathbb{NDF}^1_{\mathtt{AP}})|_{\mathtt{dt}}$$

where $\mathbb{DT}^0_{\mathtt{AP}} := \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{dt},\circ}\cup\mathcal{R}^{\mathtt{dt},\times}}, \mathcal{F})|_{\mathtt{dt}}$. From the scheme in Algorithm 6 it can be immediately concluded that the equations

$$\begin{aligned}\mathbb{NDF}^1_{\mathtt{APUP}} &= \mathbb{NDF}^1_{\mathtt{AP}}\\ \mathbb{DT}^0_{\mathtt{APUP}} &= \mathbb{DT}^0_{\mathtt{AP}}\end{aligned}$$

hold. Using this result, we can show that $\mathbb{DT}^1_{\mathtt{APUP}} = \mathbb{DT}^1_{\mathtt{AP}}$ holds as well:

$$\begin{aligned}\mathbb{DT}^1_{\mathtt{APUP}} &:= \mathbb{DT}^0_{\mathtt{APUP}} \cup +^{-1}\cdot\Delta^+\mathbb{DT}^0_{\mathtt{APUP}}\\ &= \mathbb{DT}^0_{\mathtt{APUP}} \cup [\mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{dt},\times}\cup\mathcal{R}^{\mathtt{dt},*}}, \mathbb{DT}^0_{\mathtt{APUP}} \cup \mathbb{NDF}^1_{\mathtt{APUP}})|_{\mathtt{dt}} \setminus \mathbb{DT}^0_{\mathtt{APUP}}]\\ &= \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{dt},\times}\cup\mathcal{R}^{\mathtt{dt},*}}, \mathbb{DT}^0_{\mathtt{APUP}} \cup \mathbb{NDF}^1_{\mathtt{APUP}})|_{\mathtt{dt}}\\ &= \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{dt},\times}\cup\mathcal{R}^{\mathtt{dt},*}}, \mathbb{DT}^0_{\mathtt{AP}} \cup \mathbb{NDF}^1_{\mathtt{AP}})|_{\mathtt{dt}}\\ &\quad \text{since } \mathbb{DT}^0_{\mathtt{APUP}} = \mathbb{DT}^0_{\mathtt{AP}} \text{ and } \mathbb{NDF}^1_{\mathtt{APUP}} = \mathbb{NDF}^1_{\mathtt{AP}}\\ &= \mathbb{DT}^1_{\mathtt{AP}}.\end{aligned}$$

Suppose that $i > 1$: We assume that $\mathbb{DT}^i_{\mathtt{APUP}} = \mathbb{DT}^i_{\mathtt{AP}}$ and $\mathbb{NDF}^i_{\mathtt{APUP}} = \mathbb{NDF}^i_{\mathtt{AP}}$ hold for all $i > 1$. With respect to AFP materialization the sets $\mathbb{DT}^{i+1}_{\mathtt{AP}}$ and $\mathbb{NDF}^{i+1}_{\mathtt{AP}}$ are computed as follows:

$$\begin{aligned}\mathbb{NDF}^{i+1}_{\mathtt{AP}} &:= \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{ndf}}}, \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^i_{\mathtt{AP}})) \cup \mathbb{DT}^i_{\mathtt{AP}})|_{\mathtt{ndf}}\\ \mathbb{DT}^{i+1}_{\mathtt{AP}} &:= \mathtt{lfp}(T^\star_{\mathcal{R}^{\mathtt{dt},\times}\cup\mathcal{R}^{\mathtt{dt},*}}, \mathbb{DT}^i_{\mathtt{AP}} \cup \mathbb{NDF}^{i+1}_{\mathtt{AP}})|_{\mathtt{dt}}.\end{aligned}$$

We show that $\mathbb{NDF}^{i+1}_{\mathtt{APUP}} = \mathbb{NDF}^{i+1}_{\mathtt{AP}}$:

$$\begin{aligned}\mathbb{NDF}^{i+1}_{\mathtt{APUP}} &:= \mathbb{NDF}^i_{\mathtt{APUP}} \setminus -^{-1}\cdot(\Delta^-\mathbb{NDF}^i_{\mathtt{APUP}})\\ &= \mathbb{NDF}^i_{\mathtt{APUP}} \setminus -^{-1}\cdot\mathcal{IF}_{\langle \mathbb{NDF}^i_{\mathtt{APUP}}\cup\mathbb{DT}^{i-1}_{\mathtt{APUP}}\cup\Delta^+\mathbb{DT}^{i-1}_{\mathtt{APUP}}, \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})\cup\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})\rangle}|_{\mathtt{ndf}^-}\\ &= \mathbb{NDF}^i_{\mathtt{AP}} \setminus -^{-1}\cdot\mathcal{IF}_{\langle \mathbb{NDF}^i_{\mathtt{AP}}\cup\mathbb{DT}^{i-1}_{\mathtt{AP}}\cup\Delta^+\mathbb{DT}^{i-1}_{\mathtt{AP}}, \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})\cup\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})\rangle}|_{\mathtt{ndf}^-}\\ &\quad \text{using the induction hypothesis}\\ &= \mathbb{NDF}^i_{\mathtt{AP}} \setminus (\mathbb{NDF}^i_{\mathtt{AP}} \setminus \mathbb{NDF}^{i+1}_{\mathtt{AP}})\\ &\quad \text{using the results from Propositions 6.14 and 6.16}\\ &= \mathbb{NDF}^{i+1}_{\mathtt{AP}}\\ &\quad \text{since } \mathbb{NDF}^{i+1}_{\mathtt{AP}} \subseteq \mathbb{NDF}^i_{\mathtt{AP}}.\end{aligned}$$

Based on this result, we show that $\mathbb{DT}^{i+1}_{\mathtt{APUP}} = \mathbb{DT}^{i+1}_{\mathtt{AP}}$:

$$\begin{aligned}\mathbb{DT}^{i+1}_{\mathtt{APUP}} &:= \mathbb{DT}^i_{\mathtt{APUP}} \cup +^{-1}\cdot(\Delta^+\mathbb{DT}^i_{\mathtt{APUP}})\\ &= \mathbb{DT}^i_{\mathtt{APUP}} \cup +^{-1}\cdot\mathtt{lfp}(T^\star_{\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})\cup\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})}, \mathbb{NDF}^{i+1}_{\mathtt{APUP}} \cup \mathbb{DT}^i_{\mathtt{APUP}} \cup \Delta^-\mathbb{NDF}^i_{\mathtt{APUP}})|_{\mathtt{dt}^+}\\ &= \mathbb{DT}^i_{\mathtt{AP}} \cup +^{-1}\cdot\mathtt{lfp}(T^\star_{\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})\cup\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})}, \mathbb{NDF}^{i+1}_{\mathtt{AP}} \cup \mathbb{DT}^i_{\mathtt{AP}} \cup \Delta^-\mathbb{NDF}^i_{\mathtt{AP}})|_{\mathtt{dt}^+}\\ &\quad \text{using the induction hypothesis}\\ &= \mathbb{DT}^i_{\mathtt{AP}} \cup (\mathbb{DT}^{i+1}_{\mathtt{AP}} \setminus \mathbb{DT}^i_{\mathtt{AP}})\\ &\quad \text{using the results from Propositions 6.10 and 6.12}\\ &= \mathbb{DT}^{i+1}_{\mathtt{AP}}\\ &\quad \text{since } \mathbb{DT}^i_{\mathtt{AP}} \subseteq \mathbb{DT}^{i+1}_{\mathtt{AP}}.\end{aligned}$$

Thus, we conclude that all intermediate results computed by AFP materialization coincide with the ones obtained by the application of the scheme in Algorithm 6. Correctness of AFP materialization using update propagation then directly follows from the correctness of AFP materialization. □

Consider once again our running example when using the scheme in Algorithm 6. First, we determine the update propagation rules for the sets $\mathcal{R}^{\mathtt{dt}}$ and $\mathcal{R}^{\mathtt{ndf}}$:

$\underline{\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}}):}$

$$\Delta^{+}\mathtt{dt\_e}(\mathtt{X}) \leftarrow \Delta^{-}\mathtt{ndf\_e}(\mathtt{Y}) \wedge \mathtt{succ}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{dt\_e}(\mathtt{X})$$

$\underline{\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}):}$

$$\Delta^{-}\mathtt{ndf\_e}^{-}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{dt\_e}(\mathtt{Y}) \wedge \mathtt{succ}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{ndf\_e}^{\mathtt{new}}(\mathtt{X})$$

The transition rules for $\mathcal{R}^{\mathtt{dt}}$ and $\mathcal{R}^{\mathtt{ndf}}$ are

$\underline{\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}}):}$

$$\mathtt{dt\_e}^{\mathtt{new}}(\mathtt{X}) \leftarrow \mathtt{dt\_e}(\mathtt{X})$$
$$\mathtt{dt\_e}^{\mathtt{new}}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{dt\_e}(\mathtt{X})$$

$\underline{\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}):}$

$$\mathtt{ndf\_e}^{\mathtt{new}}(\mathtt{X}) \leftarrow \mathtt{succ}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{dt\_e}^{\mathtt{new}}(\mathtt{Y})$$
$$\mathtt{dt\_e}^{\mathtt{new}}(\mathtt{X}) \leftarrow \mathtt{dt\_e}(\mathtt{X})$$
$$\mathtt{dt\_e}^{\mathtt{new}}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{dt\_e}(\mathtt{X})$$

At the beginning, the set of $\mathbb{DT}^0$-facts is initialized with the set of base facts $\mathcal{F}$ as the rule sets $\mathcal{R}^{\mathtt{dt},\circ}$ and $\mathcal{R}^{\mathtt{dt},\star}$ are empty. Applying the set $\mathbb{DT}^0$ and the rules $\mathcal{R}^{\mathtt{ndf}}$ for computing $\mathbb{NDF}^1$ yields the first and largest set of not definitely false facts with

$$\mathbb{NDF}^1 := \mathcal{F} \cup \{\mathtt{ndf\_e}(0), \mathtt{ndf\_e}(1), \mathtt{ndf\_e}(2), \mathtt{ndf\_e}(3), \mathtt{ndf\_e}(4)\}.$$

From this set, the first new $\mathbb{DT}$-facts can be calculated yielding $\Delta^{+}\mathbb{DT}^0 = \{dt\_e^{+}(4)\}$. In the following loop, $\Delta^{-}\mathbb{NDF}^i$ and $\Delta^{+}\mathbb{DT}^i$ are computed and the corresponding $\mathbb{NDF}^i$- and $\mathbb{DT}^i$-sets are updated:

$$\begin{aligned}
\Delta^{-}\mathbb{NDF}^1 &:= \{\Delta^{-}\mathtt{ndf_e}(3)\}\\
\mathbb{DT}^1 &:= \mathcal{F} \cup \{\mathtt{e}(4)\}\\
\mathbb{NDF}^2 &:= \mathcal{F} \cup \{\mathtt{ndf_e}(0), \mathtt{ndf_e}(1), \mathtt{ndf_e}(2), \mathtt{ndf_e}(4)\}\\
\Delta^{+}\mathbb{DT}^1 &:= \{\Delta^{+}\mathtt{dt_e}(2)\}
\end{aligned}$$

$$
\begin{array}{ll}
\Delta^{-}\mathbb{NDF}^{2} & := \{\Delta^{-}\texttt{ndf\_e}(1)\} \\
\mathbb{DT}^{2} & := \mathcal{F} \cup \{\texttt{e}(2), \texttt{e}(4)\} \\
\mathbb{NDF}^{3} & := \mathcal{F} \cup \{\texttt{ndf\_e}(0), \texttt{ndf\_e}(2), \texttt{ndf\_e}(4)\} \\
\Delta^{+}\mathbb{DT}^{2} & := \{\Delta^{+}\texttt{dt\_e}(0)\} \\
\Delta^{-}\mathbb{NDF}^{3} & := \emptyset \\
\mathbb{DT}^{3} & := \mathcal{F} \cup \{\texttt{e}(0), \texttt{e}(2), \texttt{e}(4)\} \\
\mathbb{NDF}^{4} & := \mathbb{NDF}^{3} \\
\Delta^{+}\mathbb{DT}^{3} & := \emptyset
\end{array}
$$

The evaluation indeed shows the desired behavior as the computation already provides a focus on the changes of $\mathbb{DT}$- and $\mathbb{NDF}$-sets. However, using transition rules when computing induced insertions or deletions leads to the complete generation of `new` state facts with respect to DT-relations and NDF-relations, respectively. In our example, the following state facts are implicitly derived when computing the corresponding delta sets:

$$
\begin{array}{lll}
\Delta^{-}\mathbb{NDF}^{1} \texttt{<--} \{\Delta^{-}\texttt{ndf\_e}(3)\} & \cup \{\texttt{dt\_e}^{\texttt{new}}(4)\} \\
 & \cup \{\texttt{ndf\_e}^{\texttt{new}}(0), \texttt{ndf\_e}^{\texttt{new}}(1), \texttt{ndf\_e}^{\texttt{new}}(2), \texttt{ndf\_e}^{\texttt{new}}(4)\} \\
\Delta^{+}\mathbb{DT}^{1} \texttt{<--} \{\Delta^{+}\texttt{dt\_e}(2)\} & \cup \{\texttt{dt\_e}^{\texttt{new}}(2), \texttt{dt\_e}^{\texttt{new}}(4)\} \\
\Delta^{-}\mathbb{NDF}^{2} \texttt{<--} \{\Delta^{-}\texttt{ndf\_e}(1)\} & \cup \{\texttt{dt\_e}^{\texttt{new}}(2), \texttt{dt\_e}^{\texttt{new}}(4)\} \\
 & \cup \{\texttt{ndf\_e}^{\texttt{new}}(0), \texttt{ndf\_e}^{\texttt{new}}(2), \texttt{ndf\_e}^{\texttt{new}}(4)\} \\
\Delta^{+}\mathbb{DT}^{2} \texttt{<--} \{\Delta^{+}\texttt{dt\_e}(0)\} & \cup \{\texttt{dt\_e}^{\texttt{new}}(0), \texttt{dt\_e}^{\texttt{new}}(2), \texttt{dt\_e}^{\texttt{new}}(4)\} \\
\Delta^{-}\mathbb{NDF}^{3} \texttt{<--} \emptyset & \cup \{\texttt{dt\_e}^{\texttt{new}}(0), \texttt{dt\_e}^{\texttt{new}}(2), \texttt{dt\_e}^{\texttt{new}}(4)\} \\
 & \cup \{\texttt{ndf\_e}^{\texttt{new}}(0), \texttt{ndf\_e}^{\texttt{new}}(2), \texttt{ndf\_e}^{\texttt{new}}(4)\} \\
\Delta^{+}\mathbb{DT}^{3} \texttt{<--} \emptyset & \cup \{\texttt{dt\_e}^{\texttt{new}}(0), \texttt{dt\_e}^{\texttt{new}}(2), \texttt{dt\_e}^{\texttt{new}}(4)\}.
\end{array}
$$

The reason for this redundancy is that materialization of side literals within the derivability- and effectiveness test is not restricted to the facts that are relevant for the particular propagated update. The same problem has been already discussed in Section 5.2 where the soft update propagation approach has been introduced as a possible solution to it. We will adopt this idea and show how the application of the Magic Sets method can provide a focus on the relevant part of the derivability and effectiveness tests in this context, as well.

6.2.2 DP Materialization Using Soft Update Propagation

Transforming the propagation and transition rules for definitely true relations using Magic Updates is unproblematic since the original rule set is semi-positive. The application of Magic Sets with respect to the stratifiable set of propagation and transition rules for not definitely false relations, however, may introduce unstratifiable cycles among these rules. Thus, for their evaluation iterated fixpoint computation is not sufficient anymore. As an example consider the following (unstratifiable) rule set $\mathcal{R}$ for defining a relation p:

$\underline{\mathcal{R}^{\mathtt{ndf}}:}$

$$\mathtt{p}(\mathtt{X}) \leftarrow \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{p}(\mathtt{Y}) \wedge \mathtt{p}(\mathtt{Z})$$
$$\mathtt{p}(\mathtt{X}) \leftarrow \mathtt{d}(\mathtt{X}).$$

The corresponding rule set for defining not definitely false relations $\mathcal{R}^{\mathtt{ndf}}$ is

$\underline{\mathcal{R}^{\mathtt{ndf}}:}$

$$\mathtt{ndf\_p}(\mathtt{X}) \leftarrow \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{dt\_p}(\mathtt{Y}) \wedge \mathtt{ndf\_p}(\mathtt{Z})$$
$$\mathtt{ndf\_p}(\mathtt{X}) \leftarrow \mathtt{d}(\mathtt{X}).$$

Rewriting the two rules yields the following stratifiable set of propagation and transition rules

$\underline{\varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}):}$

$$\Delta^{-}\mathtt{ndf\_p}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{dt\_p}(\mathtt{Y}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \mathtt{ndf\_p}(\mathtt{Z}) \wedge \neg\mathtt{ndf\_p}^{\mathtt{new}}(\mathtt{X})$$
$$\Delta^{-}\mathtt{ndf\_p}(\mathtt{X}) \leftarrow \Delta^{-}\mathtt{ndf\_p}(\mathtt{Z}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{dt\_p}(\mathtt{Y}) \wedge \neg\mathtt{ndf\_p}^{\mathtt{new}}(\mathtt{X})$$

$\underline{\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}):}$

$$\mathtt{ndf\_p}^{\mathtt{new}}(\mathtt{X}) \leftarrow \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{dt\_p}^{\mathtt{new}}(\mathtt{Y}) \wedge \mathtt{ndf\_p}^{\mathtt{new}}(\mathtt{Z})$$
$$\mathtt{ndf\_p}^{\mathtt{new}}(\mathtt{X}) \leftarrow \mathtt{d}(\mathtt{X})$$

$$\mathtt{dt\_p}^{\mathtt{new}}(\mathtt{X}) \leftarrow \mathtt{dt\_p}(\mathtt{X})$$
$$\mathtt{dt\_p}^{\mathtt{new}}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{dt\_p}(\mathtt{X}).$$

Applying the Magic Updates rewriting to $\mathcal{R}^{\mathtt{ndfp}} = \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \cup \tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ as proposed in Chapter 5 leads to the following negative cycle in the corresponding dependency graph of $\mathtt{mu}(\mathcal{R}^{\mathtt{ndfp}}_{Q^u})$:

$$\Delta^{-}\mathtt{ndf\_p} \xrightarrow{pos} \mathtt{m\_ndf\_p}^{\mathtt{new}}_{\mathtt{b}} \xrightarrow{pos} \mathtt{ndf\_p}^{\mathtt{new}}_{\mathtt{b}} \xrightarrow{neg} \Delta^{-}\mathtt{ndf\_p}$$

For the correct evaluation of the rules $\mathtt{mu}(\mathcal{R}^{\mathtt{ndfp}}_{Q^u})$, the soft stratification approach could be used. Because of the specific structure of these rules, however, we propose a different evaluation strategy which makes no use of the concept stratification at all. Since we know that the new state of DT-relations is simply the union of computed insertions and (old) facts already stored in the database, we will fold this subset of transition rules in $\tau_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ into the remaining rules for defining the new state of NDF-relations. The resulting set is denoted $\tau^{\mathtt{f}}_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ and for the above example it is as follows:

$\underline{\tau^{\mathtt{f}}_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}):}$

$$\begin{aligned}
\mathtt{ndf_p}^{\mathtt{new}}(\mathtt{X}) &\leftarrow \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{dt_p}(\mathtt{Y}) \wedge \mathtt{ndf_p}^{\mathtt{new}}(\mathtt{Z})\\
\mathtt{ndf_p}^{\mathtt{new}}(\mathtt{X}) &\leftarrow \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\Delta^{+}\mathtt{dt_p}(\mathtt{Y}) \wedge \mathtt{ndf_p}^{\mathtt{new}}(\mathtt{Z})\\
\mathtt{ndf_p}^{\mathtt{new}}(\mathtt{X}) &\leftarrow \mathtt{d}(\mathtt{X}).
\end{aligned}$$

In this way, the only negative references to derived relations left are the ones occurring in the effectiveness tests. Applying now the Magic Updates rewriting to the set $\mathcal{R}^{\mathtt{ndfpf}} = \varphi_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}}) \cup \tau^{\mathtt{f}}_{\mathtt{ndf}}(\mathcal{R}^{\mathtt{ndf}})$ still leads to an unstratifiable rule set. Therefore, we will consider the transformed propagation rules $\mathcal{R}^{\Delta\mathtt{ndf}} \subset \mathtt{mu}(\mathcal{R}^{\mathtt{ndfpf}}_{Q^u})$ and the transformed transition rules $\mathcal{R}^{\mathtt{nndf}} \subset \mathtt{mu}(\mathcal{R}^{\mathtt{ndfpf}}_{Q^u})$ with

$$\mathtt{mu}(\mathcal{R}^{\mathtt{ndfpf}}_{Q^u}) = \mathcal{R}^{\Delta\mathtt{ndf}} \uplus \mathcal{R}^{\mathtt{nndf}},$$

separately, getting two stratifiable rule sets. Consider again our example from above after applying the Magic Updates rewriting with respect to the abstract propagation queries represented by $\Delta^{+}\mathtt{dt\_p}(\mathtt{Y})$ and $\Delta^{-}\mathtt{ndf\_p}(\mathtt{Z})$. The resulting sets $\mathcal{R}^{\Delta\mathtt{ndf}}$ and $\mathcal{R}^{\mathtt{nndf}}$ are then given by

$\underline{\mathcal{R}^{\Delta\mathtt{ndf}}:}$

$$\begin{aligned}
\Delta^{-}\mathtt{ndf_p}(\mathtt{X}) &\leftarrow \Delta^{+}\mathtt{dt_p}(\mathtt{Y}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \mathtt{ndf_p}(\mathtt{Z}) \wedge \neg\mathtt{ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X})\\
\Delta^{-}\mathtt{ndf_p}(\mathtt{X}) &\leftarrow \Delta^{-}\mathtt{ndf_p}(\mathtt{Z}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{dt_p}(\mathtt{Y}) \wedge \neg\mathtt{ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X})\\
\mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) &\leftarrow \Delta^{+}\mathtt{dt_p}(\mathtt{Y}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \mathtt{ndf_p}(\mathtt{Z})\\
\mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) &\leftarrow \Delta^{-}\mathtt{ndf_p}(\mathtt{Z}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{dt_p}(\mathtt{Y})
\end{aligned}$$

$\underline{\mathcal{R}^{\mathtt{nndf}}:}$

$$\begin{aligned}
\mathtt{ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) &\leftarrow \mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{dt_p}(\mathtt{Y}) \wedge \mathtt{ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{Z})\\
\mathtt{ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) &\leftarrow \mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\Delta^{+}\mathtt{dt_p}(\mathtt{Y}) \wedge \mathtt{ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{Z})\\
\mathtt{ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) &\leftarrow \mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) \wedge \mathtt{d}(\mathtt{X})\\
\mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) &\leftarrow \mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\mathtt{dt_p}(\mathtt{Y})\\
\mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) &\leftarrow \mathtt{m_ndf_p}^{\mathtt{new}}_{\mathtt{b}}(\mathtt{X}) \wedge \mathtt{b}(\mathtt{X},\mathtt{Y},\mathtt{Z}) \wedge \neg\Delta^{+}\mathtt{dt_p}(\mathtt{Y}).
\end{aligned}$$

As mentioned above, combining these two sets would still lead to an unstratifiable rule set. However, this unstratifiability is solely caused by the effectiveness tests which negatively refers to the new state of the transformed NDF-relations in $\mathcal{R}^{\mathtt{nndf}}$. Therefore, we propose to evaluate the rule sets $\mathcal{R}^{\Delta\mathtt{ndf}}$ and $\mathcal{R}^{\mathtt{nndf}}$, separately, by using the so-called sequential consequence operator [Beh01].

Definition 6.18 (Sequential Consequence Operator) *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database, $P_1 \uplus P_2$ a partition of $\mathcal{R}$ and $\mathcal{I} \subseteq \mathcal{H}_{\mathcal{D}}$ a set of ground atoms. The sequential consequence operator $\widetilde{T}_{\langle P_1,P_2\rangle}$ is a mapping on sets of ground atoms and is defined as*

$$\widetilde{T}_{\langle P_1,P_2\rangle}(\mathcal{I}) := T^{\star}_{P_2}(\mathtt{lfp}(T^{\star}_{P_1},\mathcal{I})).$$

The basic property of $\widetilde{T}_{\langle P_1,P_2\rangle}$ is that before P_2 is applied once, the rule set P_1 is evaluated until no more derivations can be made. In principle, the evaluation coincides with the special case of applying the soft consequence operator to a binary partition.

Lemma 6.2 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R}\rangle$ be a deductive database and $\mathcal{P} = P_1 \uplus P_2$ a binary partition of $\mathcal{R}$. Then the least fixpoint of $\widetilde{T}_{\langle P_1,P_2\rangle}$ always exists and coincides with the least fixpoint of $T^s_{\mathcal{P}}$, i.e.,*

$$\mathtt{lfp}(\widetilde{T}_{\langle P_1,P_2\rangle},\mathcal{F}) = \mathtt{lfp}(T^s_{\mathcal{P}},\mathcal{F}).$$

Proof: This proposition follows immediately from Definitions 3.4 and 6.18. □

Although both operators obtain the same result, the sequential consequence operator explicitly computes the fixpoint of the lower component P_1 before applying P_2 whereas the soft consequence operator always has to test whether the application of P_1 still leads to new derivations. This allows for a more efficient implementation of $\widetilde{T}_{\langle P_1,P_2\rangle}$ in comparison to the soft consequence operator $T^s_{\mathcal{P}}$.

It is easy to see that the partition $\mathcal{R}^{\Delta\mathtt{ndf}} \uplus \mathcal{R}^{\mathtt{nndf}}$ satisfies the condition of a soft stratification. Using $\mathcal{R}^{\Delta\mathtt{ndf}}$ and $\mathcal{R}^{\mathtt{nndf}}$ as first respectively second rule set, the sequential operator makes sure that all necessary new state facts are derived before a propagation rule using these facts within its derivability and effectiveness test is evaluated. Thus, from Lemma 6.2 and Proposition 5.1 it can be followed that the least fixpoint of the sequential consequence operator coincides with the total well-founded model $\mathcal{M}_{\mathcal{D}}$ of the softly stratifiable database $D = \langle \mathcal{F}, \mathcal{R}^{\Delta\mathtt{ndf}} \uplus \mathcal{R}^{\mathtt{nndf}}\rangle$:

$$\mathcal{M}_{\langle \mathcal{F}, \mathcal{R}^{\Delta\mathtt{ndf}} \uplus \mathcal{R}^{\mathtt{nndf}}\rangle} = \mathtt{lfp}(\widetilde{T}_{\langle \mathcal{R}^{\Delta\mathtt{ndf}},\mathcal{R}^{\mathtt{nndf}}\rangle},\mathcal{F}).$$

The least fixpoint of $\widetilde{T}$ with respect to the rule sets $\mathcal{R}^{\Delta\mathtt{ndf}}$ and $\mathcal{R}^{\mathtt{nndf}}$ corresponds to the fixpoint of the following sequence:

$$\begin{aligned}
\mathcal{F}_1 &:= \mathtt{lfp}(T_{\mathcal{R}^{\mathtt{nndf}}},\mathcal{F})\\
\mathcal{F}_2 &:= T_{\mathcal{R}^{\Delta\mathtt{ndf}}}(\mathcal{F}_1)\\
\mathcal{F}_3 &:= \mathtt{lfp}(T_{\mathcal{R}^{\mathtt{nndf}}},\mathcal{F}_2)\\
\mathcal{F}_4 &:= T_{\mathcal{R}^{\Delta\mathtt{ndf}}}(\mathcal{F}_3)\\
&\vdots\\
\mathcal{F}_i &:= T_{\mathcal{R}^{\Delta\mathtt{ndf}}}(\mathcal{F}_{i-1})
\end{aligned}$$

Algorithm 7 AFP materialization using Magic Updates

i $:= 0$;
$\mathbb{DT}^0$ $:= \mathtt{lfp}(T^{\star}_{\mathcal{R}^{\mathtt{dt},\circ}\cup\mathcal{R}^{\mathtt{dt},\times}}, \mathcal{F})|_{\mathtt{dt}}$;
$\mathbb{NDF}^1$ $:= \mathtt{lfp}(T^{\star}_{\mathcal{R}^{\mathtt{ndf}}}, \mathbb{DT}^0 \cup \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}^0)))|_{\mathtt{ndf}}$;
$\Delta^{+}\mathbb{DT}^0 := + \cdot [\mathtt{lfp}(T^{\star}_{\mathcal{R}^{\mathtt{dt},\times}\cup\mathcal{R}^{\mathtt{dt},*}}, \mathbb{DT}^0 \cup \mathbb{NDF}^1)|_{\mathtt{dt}} \setminus \mathbb{DT}^0]$;
while $\Delta^{+}\mathbb{DT}^i \neq \emptyset$ **do**
  i $:= i + 1$;
  $\Delta^{-}\mathbb{NDF}^i := \mathtt{lfp}(\widetilde{T}_{\langle \mathcal{R}^{\Delta\mathtt{ndf}}, \mathcal{R}^{\mathtt{nndf}} \rangle}, \mathbb{NDF}^i \cup \mathbb{DT}^{i-1} \cup \Delta^{+}\mathbb{DT}^{i-1})|_{\mathtt{ndf}^-}$;
  $\mathbb{DT}^i$ $:= \mathbb{DT}^{i-1} \cup +^{-1} \cdot (\Delta^{+}\mathbb{DT}^{i-1})$;
  $\mathbb{NDF}^{i+1}$ $:= \mathbb{NDF}^i \setminus -^{-1} \cdot (\Delta^{-}\mathbb{NDF}^i)$;
  $\Delta^{+}\mathbb{DT}^i$ $:= \mathtt{lfp}(T^{\star}_{\mathcal{R}^{\Delta\mathtt{ndt}}}, \mathbb{NDF}^{i+1} \cup \mathbb{DT}^i \cup \Delta^{-}\mathbb{NDF}^i)|_{\mathtt{dt}^+}$;
end while
$\mathbb{NDF}$ $:= \mathbb{NDF}^{i+1}$;
$\mathbb{DT}$ $:= \mathbb{DT}^i$;

The application of $\widetilde{T}_{\langle \mathcal{R}^{\Delta\mathtt{ndf}}, \mathcal{R}^{\mathtt{nndf}} \rangle}$ alternates between the determination of induced deletions from $\mathbb{NDF}$ after proving their effectiveness within the inner fixpoint calculation. Starting from the set of base facts, the effectiveness of all induced updates to be derived is tested one iteration round before in the inner fixpoint computation such that the operator never evaluates negative literals too early.

Similar to the transformation of NDF-relations, the rule set $\mathtt{mu}(\mathcal{R}^{\mathtt{dtp}}_{Q^u})$ with $\mathcal{R}^{\mathtt{dtp}} = \varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}}) \cup \tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ is used to denote the rules resulting from the application of the Magic Updates rewriting to the propagation and transition rules for DT-relations in $\mathcal{R}^{\mathtt{dtp}}$. As the rules $\mathcal{R}^{\mathtt{dtp}}$, however, are semi-positive the transformed rules $\mathtt{mu}(\mathcal{R}^{\mathtt{dtp}}_{Q^u})$ must be semi-positive as well. Hence, for their evaluation the simple immediate consequence operator can be used again. Additionally, it is not necessary to partition the Magic Updates transformed rules $\mathtt{mu}(\mathcal{R}^{\mathtt{dtp}}_{Q^u})$ and we will use the single set $\mathcal{R}^{\Delta\mathtt{ndt}}$ to denote the Magic Updates rewritten DT-relations, i.e., $\mathcal{R}^{\Delta\mathtt{ndt}} := \mathtt{mu}(\mathcal{R}^{\mathtt{dtp}}_{Q^u})$.

Based on these results, we can now define the scheme of AFP materialization using Magic Updates with Algorithm 7. The basic difference to the previously introduced Algorithm 6 is that only relevant new state facts are computed. Consider once again our running example for defining the unstratifiable relation e. The set $\mathcal{R}^{\Delta\mathtt{ndt}}$ of Magic Updates transformed rules with respect to DT-relations is given by:

$\mathcal{R}^{\Delta\mathtt{ndt}}$:

$$\Delta^{+}\mathtt{dt\_e(X)} \leftarrow \Delta^{-}\mathtt{ndf\_e(Y)} \wedge \mathtt{succ(X,Y)} \wedge \neg\mathtt{dt\_e(X)}$$

Note that the original transition rules $\tau_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ are not included in $\mathcal{R}^{\Delta\mathtt{ndt}}$ anymore as the (single) propagation rule in $\varphi_{\mathtt{dt}}(\mathcal{R}^{\mathtt{dt}})$ contains no references to the new state relation $\mathtt{dt\_e}^{\mathtt{new}}$. The sets $\mathcal{R}^{\Delta\mathtt{ndf}}$ and $\mathcal{R}^{\mathtt{nndf}}$ of Magic Updates rewritten NDF-relations are:

$\underline{\mathcal{R}^{\Delta\mathtt{ndf}}:}$

$$\Delta^{-}\mathtt{ndf\_e}^{-}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{dt\_e}(\mathtt{Y}) \wedge \mathtt{succ}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{ndf\_e}_{\mathtt{b}}^{\mathtt{new}}(\mathtt{X})$$
$$\mathtt{m\_ndf\_e}_{\mathtt{b}}^{\mathtt{new}}(\mathtt{X}) \leftarrow \Delta^{+}\mathtt{dt\_e}(\mathtt{Y}) \wedge \mathtt{succ}(\mathtt{X},\mathtt{Y})$$

$\underline{\mathcal{R}^{\mathtt{nndf}}:}$

$$\mathtt{ndf\_e}_{\mathtt{b}}^{\mathtt{new}}(\mathtt{X}) \leftarrow \mathtt{m\_ndf\_e}_{\mathtt{b}}^{\mathtt{new}}(\mathtt{X}) \wedge \mathtt{succ}(\mathtt{X},\mathtt{Y}) \wedge \neg\mathtt{dt\_e}(\mathtt{Y})$$
$$\mathtt{ndf\_e}_{\mathtt{b}}^{\mathtt{new}}(\mathtt{X}) \leftarrow \mathtt{m\_ndf\_e}_{\mathtt{b}}^{\mathtt{new}}(\mathtt{X}) \wedge \mathtt{succ}(\mathtt{X},\mathtt{Y}) \wedge \neg\Delta^{+}\mathtt{dt\_e}(\mathtt{Y})$$

During the application of Algorithm 7 using these rule sets the same $\Delta^{+}\mathbb{DT}^{i}$ and $\Delta^{-}\mathbb{NDF}^{i}$ are computed as in the previous case of applying Algorithm 6. However, only relevant new state facts are derived when computing the corresponding delta sets:

$$\begin{aligned}
\Delta^{-}\mathbb{NDF}^{1} &\leftarrow\!\!- \{\Delta^{-}\mathtt{ndf_e}(3)\} \cup \{\mathtt{m_ndf_e}_{\mathtt{b}}^{\mathtt{new}}(3)\}\\
\Delta^{+}\mathbb{DT}^{1} &\leftarrow\!\!- \{\Delta^{+}\mathtt{dt_e}(2)\}\\
\Delta^{-}\mathbb{NDF}^{2} &\leftarrow\!\!- \{\Delta^{-}\mathtt{ndf_e}(1)\} \cup \{\mathtt{m_ndf_e}_{\mathtt{b}}^{\mathtt{new}}(1)\}\\
\Delta^{+}\mathbb{DT}^{2} &\leftarrow\!\!- \{\Delta^{+}\mathtt{dt_e}(0)\}\\
\Delta^{-}\mathbb{NDF}^{3} &\leftarrow\!\!- \emptyset\\
\Delta^{+}\mathbb{DT}^{3} &\leftarrow\!\!- \emptyset.
\end{aligned}$$

In each phase, only those facts are computed that lead to changes in the corresponding $\mathbb{DT}$- and $\mathbb{NDF}$-set avoiding full materialization of respective new state relations. In this example, only two sub-queries with respect to the new state relation $\mathtt{ndf\_e}^{\mathtt{new}}$ have to be generated, asking for alternative derivations of the facts $\mathtt{ndf\_e}(3)$ and $\mathtt{ndf\_e}(1)$ in order to show the effectiveness of their deletion from the respective $\mathbb{NDF}$-sets.

With Algorithm 8 we now define the final algorithm, i.e., iterated AFP materialization using Magic Updates, for the efficient computation of the well-founded model of general deductive databases. This algorithm extends the scheme of Algorithm 7 by considering a multi-layered rule set which allows a differentiated treatment of stratified and unstratified layers. The evaluation of stratified layers coincides with the iterated fixpoint computation as proposed in Section 3.2.1. For the evaluation of unstratified predicates, however, Magic Updates transformed rules are applied. Note that quite similar to the iterated AFP materialization from Algorithm 5, for computing the sets $\mathbb{DT}_{l}^{0}$ and $\mathbb{NDF}_{l}^{1}$ the not definitely false

Algorithm 8 Iterated AFP materialization using Magic Updates

$\mathbb{DT}_0 := \mathcal{F}$;
$\mathbb{NDF}_0 := \mathcal{F}$;
for each layer $l = 1, \ldots, m$ of $\mathcal{R}^{\mathtt{dp}}$ defined by $\lambda^{\mathtt{dp}}$ **do**
  $i := 0$;
  $\mathbb{DT}_l^0 := \mathtt{lfp}(T^{\star}_{\mathcal{R}_l^{\mathtt{dt},\circ} \cup \mathcal{R}_l^{\mathtt{dt},\times}}, \mathbb{DT}_{l-1} \cup \mathbb{NDF}_{l-1})|_{\mathtt{dt}}$;
  $\mathbb{NDF}_l^1 := \mathtt{lfp}(T^{\star}_{\mathcal{R}_l^{\mathtt{ndf}}}, \mathbb{NDF}_{l-1} \cup \mathtt{ndf}(\mathtt{dt}^{-1}(\mathbb{DT}_l^0)) \cup \mathbb{DT}_l^0)|_{\mathtt{ndf}}$;
  $\Delta^{+}\mathbb{DT}_l^0 := + \cdot [\mathtt{lfp}(T^{\star}_{\mathcal{R}_l^{\mathtt{dt},\times} \cup \mathcal{R}_l^{\mathtt{dt},*}}, \mathbb{DT}_l^0 \cup \mathbb{NDF}_l^1)|_{\mathtt{dt}} \setminus \mathbb{DT}_l^0]$;
  while $\Delta^{+}\mathbb{DT}_l^i \neq \emptyset$ **do**
    $i := i + 1$;
    $\Delta^{-}\mathbb{NDF}_l^i := \mathtt{lfp}(\widetilde{T}_{\langle \mathcal{R}_l^{\Delta\mathtt{ndf}}, \mathcal{R}_l^{\mathtt{nndf}} \rangle}, \mathbb{NDF}_l^i \cup \mathbb{DT}_l^{i-1} \cup \Delta^{+}\mathbb{DT}_l^{i-1})|_{\mathtt{ndf}^{-}}$;
    $\mathbb{DT}_l^i := \mathbb{DT}_l^{i-1} \cup +^{-1} \cdot (\Delta^{+}\mathbb{DT}_l^{i-1})$;
    $\mathbb{NDF}_l^{i+1} := \mathbb{NDF}_l^i \setminus -^{-1} \cdot (\Delta^{-}\mathbb{NDF}_l^i)$;
    $\Delta^{+}\mathbb{DT}_l^i := \mathtt{lfp}(T^{\star}_{\mathcal{R}_l^{\Delta\mathtt{ndt}}}, \mathbb{NDF}_l^{i+1} \cup \mathbb{DT}_l^i \cup \Delta^{-}\mathbb{NDF}_l^i)|_{\mathtt{dt}^{+}}$;
  end while
  $\mathbb{NDF}_l := \mathbb{NDF}_l^{i+1}$;
  $\mathbb{DT}_l := \mathbb{DT}_l^i$;
end for
$\mathbb{DT} := \mathbb{DT}_m$;
$\mathbb{NDF} := \mathbb{NDF}_m$;

facts of deeper layers, i.e., $\mathbb{NDF}_{l-1}$, are additionally employed. This in turn requires to initialize not only the first set of definitely true facts but also the first set of not definitely false relations $\mathbb{NDF}_0$ with the fact base $\mathcal{F}$.

Theorem 6.3 *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{R} \rangle$ be a deductive database and λ a layering on $\mathcal{D}$. Then iterated AFP materialization using Magic Updates as in Algorithm 8 always terminates, and the sets $\mathbb{DT}$ and $\mathbb{NDF}$ correctly represent the well-founded model of $\mathcal{D}$. It holds that*

$$\mathcal{M}_{\mathcal{D}} = \mathtt{dt}^{-1}(\mathbb{DT}) \cup \neg \cdot \overline{\mathtt{ndf}^{-1}(\mathbb{NDF})}.$$

Proof: The proposition of this theorem follows from the results of Theorems 5.1, 6.2, and 6.1 as well as from Lemma 6.2. □

6.3 Discussion

In this chapter, we have presented a new efficient bottom-up evaluation procedure for computing well-founded models of arbitrary, i.e., potentially unstratifiable deductive databases. This procedure represents a generalization of differential

fixpoint computation [BR87] proposed for the efficient evaluation of stratifiable databases (cf. Section 3.1). It provides a practical method for handling normal logic programs that involve unstratified negation in a manner that may be mixed with other approaches such as sip strategies and further rule optimization techniques (e.g. [RBK88, NRSU89, Sag90, SSS90, CG94, NRSU95, Aze97]). Based on the doubled program approach [KSS95] we used the Magic Updates transformation from Section 5.2 in order to restrict computation to changes of definitely true and not definitely false facts. Because of the specific context, we are able to solve stratification problems which arise if the Magic Sets transformation is used in combination with propagation rules by introducing the sequential consequence operator. Its application in combination with Magic Updates transformed doubled programs allows for an even more efficient evaluation than the more general soft stratification approach.

Apparently, our approach represents a significant improvement of the approach for computing the KSS Alternating Fixpoint Model in Algorithm 3 because any repeated computations are avoided. A similar result has been obtained by methods proposed for well-founded model computation based on residual program evaluation [Bry90a]. A residual program consists of conditional facts [Bry89, DK89, Bry90a] which are ground instances of rules without positive body literals. The advantage of this notion is that negative dependencies are made explicit. Consequently, this approach can provide additional information about the reason why certain atoms are considered undefined within the resulting well-founded model by showing which negative dependencies could not be dissolved. In addition, this approach can be faster than the original alternating fixpoint approach. As an example consider again the rule

$$\texttt{e(X)} \leftarrow \texttt{succ(X,Y)} \wedge \neg \texttt{e(Y)}$$

together with the following finite successor relation $succ := \{(i, i+1) |\ 0 \leq i \leq n\}$. Computing the corresponding well-founded model using the residual program approach (or our proposed soft alternating method from Algorithm 8) would need time $O(n)$. However, the alternating fixpoint approach (cf. Algorithm 2) needs n iterations, each costing $O(n)$. Thus, the total cost is $O(n^2)$. On the other hand, the alternating fixpoint approach or our soft alternating fixpoint always need polynomial time whereas the residual program can grow to exponential size. In addition, redundant derivations may also occur during the evaluation of residual programs.

Solutions to these problems have been suggested in [BZF96, BZF97, BDFZ01] where the authors propose a delayed generation and reduction of certain conditional facts. In particular, this prevents exponential growth of residual programs and can be employed for avoiding redundant derivations as well. This in turn requires a complex rule analysis which implicitly takes place in our approach by using specialized (transformed) propagation rules. It can be concluded that our

approach of optimizing alternating fixpoint computation and the optimized evaluation of residual programs lead to similar improvements and are closely related to each other. However, our approach represents a much simpler way of achieving these results and fits well into the database context. An implementation of the residual program approach requires new index structures and new rule optimizing techniques to be added to the database in order to handle conditional facts and algorithms working with them. This is not necessary in our framework as the proposed rule transformation is independent of other rule optimizing techniques.

Chapter 7

Conclusion

In this thesis, we have developed new efficient inference mechanisms for transformation-based approaches to handling stratifiable as well as unstratifiable recursion in deductive databases. To this end, deductive services are uniformly accomplished by encoding the respective tasks into deductive rules and evaluating these rules by means of the soft stratification approach. The suitability of this approach has been investigated on the basis of query evaluation and update propagation. Additionally, it has been shown that the concept of soft stratification can be also used for an efficient implementation of the alternating fixpoint operator in order to compute the well-founded model of arbitrary unstratifiable databases.

In *Chapter 3* constructive bottom-up methods for computing the semantics of deductive database are recalled which are based on fixpoint computations. These methods iteratively materialize derived facts by applying a deductive rule set over a given input fact base until no more new derivations can be made. For the derivation of facts different consequence operators are employed which represent variants of the immediate consequence operator proposed by van Emden and Kowalski. Among them, the new soft consequence operator is introduced which is closely related to Kerisit's weak consequence operator [KP88] and serves as the basic evaluation mechanism for the transformation-based techniques suggested in subsequent chapters.

Chapter 4 shows how Magic Sets-based query evaluation in stratifiable databases can be efficiently realized using the soft stratification approach. To this end, a stratification problem arising when applying Magic Sets to an originally stratifiable rule set is cured by means of soft stratification. This new stratification concept employs additional information from the Magic Sets transformation in order to find an appropriate rule ordering for the evaluation of Magic Sets rewritten rules using the soft consequence operator. Soft stratification together with the soft consequence operator then represent the soft stratification approach. On its basis, a new transformation-based solution to the problem of optimizing existential (derived) queries is presented by extending the Magic Sets approach.

Chapter 5 illustrates how the new inference mechanism developed in Chapter 4 can be employed for efficient update propagation. It recalls the structured update propagation approach [Gri97] that propagates updates in a bottom-up manner and at each stage initiates top-down query evaluation processes (according to the Magic Sets approach) in order to determine further induced updates. In this approach, deductive rules are compiled by means of Magic Updates rewriting [Gri97, Man94] which encodes the task of update propagation as well as Magic Sets optimizations into deductive rules. It is shown, that a slightly modified Magic Updates rewriting may not only provide a more compact representation of propagation rules relevant for computing the induced changes but additionally always specifies a softly stratifiable rule set. Thus, the soft stratification approach can be employed for their efficient evaluation avoiding the expensive application of too general inference mechanisms like Van Gelder's alternating fixpoint operator [vG89].

Chapter 6 is concerned with improving the implementation of the alternating fixpoint computation by using the results presented in previous chapters. To this end, the doubled program approach to implementing the alternating fixpoint operator is extended by update propagation techniques. This avoids redundant computations of facts, as only those definitely true and not definitely false facts are derived at the end of each iteration round which have to be inserted into or deleted from the database during the fixpoint evaluation process, respectively. It is shown that the used propagation rules are always softly stratifiable such that the soft stratification approach (based on a simplified soft consequence operator) can be employed for their efficient evaluation.

Summary

On the whole, this thesis shows that deductive services are well realizable by means of the soft stratification approach. The proposed soft consequence operator represents an efficient inference component for softly stratifiable rules and is well-suited for extending the DBMS of existing relational database systems. It allows for implementing the well-known differential fixpoint computation of recursive views and is independent of other established optimization techniques such as algebraic manipulation. On the basis of the soft stratification approach, we have presented transformation-based techniques (known approaches as well as new ones) which allow for an efficient implementation of query evaluation and update propagation with respect to stratifiable recursion. These techniques may provide a realistic framework for extending the expressive power of relational database systems in order to implement the class of recursive views as proposed by the new SQL:1999 standard.

Apart from the positive results, a possible drawback of the suggested transformation-based approaches is the high number of deductive rules that have to

be generated for each database service. For instance, the application of Magic Sets to one deductive rule may lead to the generation of an exponential number of rules with respect to the arity of the rule's head and their evaluation does not necessarily improve the efficiency of query evaluation. However, several approaches to optimizing the Magic Sets transformation have been already developed which may reduce the number of rewritten rules for specific schemata or allow for a more compact representation of query restricted rules in certain cases. These approaches are independent of our soft stratification method and thus, can be applied to improving our suggested transformation techniques, too. Another drawback of our proposed framework is the usage of purely transformation-based approaches to solving various database services. The advantage of their independence of the underlying inference mechanism may be considered their weakness as well. For example, new algorithmic ideas for improving the soft update propagation approach which cannot be solely incorporated into its transformation process but require modifications of the used inference mechanism may be not feasible because these changes may negatively influence other database services like query evaluation based on the same inference procedure.

In fact, there is a price to pay for the independence of database services and database engine quite similar to the one for storing data with physical and logical independence. The latter concept of data abstraction, however, is well-established in the database context because it simplifies and systematizes application maintenance leaving changes in any of the abstraction levels to be largely contained locally. The benefits of the resulting ANSI/SPARC layered model of database architecture can be rediscovered in our proposed architecture of a transformation-based deductive database system from Figure 1.1 in Chapter 1.

Future Work

As far as future work is concerned, our approach can be extended and optimized in several ways: A major aim is to investigate how the proposed methods can be transferred into the SQL context such that additional language concepts of SQL like Null values, multisets and aggregates are taken into account as well. [Pie01] proposed to consider a complete syntactical subset of SQL, called *Basis-SQL*, allowing the definition of SQL expressions which can be most directly translated into equivalent Datalog rules. However, the problem of how to treat the different transaction concept and the additional language features of SQL mentioned above remained unsolved. Another possible way is to transfer our results into relational algebra. To this end, our transformation-based approaches are to be interpreted as special algebraic manipulation rules which can be applied like other algebraic laws such as selection pushing or splitting. In this context it would be interesting to investigate to which extent these known algebraic laws can be freely combined with the new ones resulting from our rewriting techniques. A third approach

to transferring our methods into the SQL world is the usage of triggers which has been already suggested in [CW90, SJGP90, CW91, CFPT94, GL96, Gri97, Pie01]. To this end, active rules are automatically (or semi-automatically) derived from high level specifications such as view definitions and integrity constraints. In [CW90] and subsequent publications the authors have shown that active rules are well-suited for implementing deductive inference. The idea of using active rules for materializing derived relations has been taken up by Griefahn in [Gri97] where a uniform approach to the implementation of query evaluation and update propagation has been developed. In this context, it ought to be investigated how our proposed inference component and our transformation-based techniques can be efficiently realized by means of active rules, too.

Another possible enhancement of our proposed framework is the development of cost-based approaches to query evaluation and update propagation which solely rewrite a subset of the considered deductive rules such that intermediate results are only partially materialized. These transformation-based methods ought to take estimated relation sizes and additional cost measurements for performing join and union operations into account. Work in this area is closely related to methods of dynamic query processing, e.g. [CG94, SHP$^+$96, GPFS02]. Moreover, the realization of further deductive services such as view updating is to be addressed. Finally, practical work based on the foundations provided in this thesis is just in its initial phase. First implementations of the soft stratification approach have been completed using the programming languages JAVA and PROLOG. The results may form a basis from which prototypical implementations can be developed in order to extend the expressive power of commercial systems such as Oracle or Microsoft Access.

Bibliography

[ABW88] Krzysztof R. Apt, Howard A. Blair, and Adrian Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of deductive databases and logic programs*, pages 89–148. Morgan Kaufmann, Los Altos, USA, 1988.

[Apt90] Krzysztof R. Apt. Logic programming. In Jan van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 10, pages 493–574. The MIT Press, New York, USA, 1990.

[Aze97] Paulo J. Azevedo. Magic sets with full sharing. *Journal of Logic Programming*, 30(3):223–237, March 1997.

[Ban86] François Bancilhon. Naive evaluation of recursively defined relations. In Michael L. Brodie and John Mylopoulos, editors, *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, pages 165–178, New York, 1986. Springer.

[BB82] Philip A. Bernstein and Barbara T. Blaustein. Fast methods for testing quantified relational calculus assertions. In *Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data*, pages 39–50, New York, USA, June 1982. ACM Press.

[BBC80] Philip A. Bernstein, Barbara T. Blaustein, and Edmund M. Clarke. Fast maintenance of semantic integrity assertions using redundant aggregate data. In *Proceedings of the International Conference on Very Large Databases (VLDB '80)*, pages 126–136, Long Beach, USA, October 1980. IEEE Computer Society Press.

[BD95] S. Brass and J. Dix. Characterizations of the stable semantics by partial evaluation. In *Proceedings of the 3rd International Conference on Logic Programming and Nonmonotonic Reasoning*, volume 928 of *LNAI*, pages 85–98, Berlin, June 1995. Springer.

[BDD+98] Randall G. Bello, Karl Dias, Alan Downing, James Feenan, Jr., William D. Norcott, Harry Sun, Andrew Witkowski, and Mohamed

Ziauddin. Materialized views in Oracle. In *Proceedings of the International Conference on Very Large Databases (VLDB '98)*, pages 659–664, Los Altos, USA, August 1998. Morgan Kaufmann.

[BDFZ01] Stefan Brass, Jürgen Dix, Burkhard Freitag, and Ulrich Zukowski. Transformation-based bottom-up computation of the well-founded model. *Theory and Practice of Logic Programming (TPLP)*, 1(5):497–538, September 2001.

[BDM88] F. Bry, H. Decker, and R. Manthey. A uniform approach to constraint satisfaction and constraint satisfiability in deductive databases. In *Proceedings of the International Conference on Extending Database Technology (EDBT '88)*, volume 303 of *LNCS*, pages 488–505, Venice, Italy, March 1988. Springer.

[Beh00] Andreas Behrend. A dynamic approach to deductive query evaluation. In *15th Workshop on Logic Programming and Constraint Systems, Collocated with ECAI 2000*, pages 99–112, Berlin, August 2000. GMD Report 110.

[Beh01] Andreas Behrend. Efficient computation of the well-founded model using update propagation. In *Proceedings of the Eigth International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2250 of *LNCS*, pages 422–437, Berlin, December 2001. Springer.

[Beh03] Andreas Behrend. Soft stratification for magic set based query evaluation in deductive databases. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '03)*, pages 102–110, New York, June 2003. ACM Press.

[BKR+99] Yuri Breitbart, Raghavan Komondoor, Rajeev Rastogi, S. Seshadri, and Avi Silberschatz. Update propagation protocols for replicated databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 97–108, New York, USA, June 1999. ACM Press.

[BMM91] Francois Bry, Rainer Manthey, and Bern Martens. Integrity verification in knowledge bases. In *Proceedings of the 2nd Russian Conference on Logic Programming*, volume 592 of *LNCS*, pages 114–139, St. Petersburg, Russia, September 1991. Springer 1992.

[BMR88] I. Balbin, K. Meenakshi, and K. Ramamohanarao. A query independent method for magic set computation on stratified databases. In *Proceedings of the International Conference on Fifth Generation*

Computer Systems, volume 2, pages 711–718, Berlin, December 1988. Springer.

[BMSU86] François Bancilhon, David Maier, Yehoshua Sagiv, and Jeffrey D. Ullman. Magic sets and other strange ways to implement logic programs (extended abstract). In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '86)*, pages 1–15, New York, USA, March 1986. ACM Press.

[BNR+87] C. Beeri, S. Naqvi, R. Ramakrishnan, O. Shmueli, and S. Tsur. Sets and negation in a logic data base language (LDL1). In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '87)*, pages 21–37, New York, USA, March 1987. ACM Press.

[BPRM91] I. Balbin, G. S. Port, K. Ramamohanarao, and K. Meenakshi. Efficient bottom-up computation of queries. *Journal of Logic Programming*, 11(3&4):295–344, October 1991.

[BR86] François Bancilhon and Raghu Ramakrishnan. An amateur's introduction to recursive query processing strategies. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, pages 16–52, Washington, D.C., May 1986. ACM Press.

[BR87] Isaac Balbin and Kotagiri Ramamohanarao. A generalization of the differential approach to recursive query evaluation. *Journal of Logic Programming*, 4(3):259–262, September 1987.

[BR91] Catriel Beeri and Raghu Ramakrishnan. On the power of magic. *Journal of Logic Programming*, 10(3,4):255–299, April 1991.

[Bra96] Stefan Brass. SLDMagic — an improved magic set technique. In *Proceedings of the Third International Workshop on Advances in Databases and Information Systems - ADBIS'96*, pages 75–83, Moscow, Russia, September 1996. MEPhI.

[Bry89] François Bry. Logic programming as constructivism: a formalization and its application to databases. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '89)*, pages 34–50, New York, USA, March 1989. ACM Press.

[Bry90a] François Bry. Negation in logic programming: A formalization in constructive logic. In *Proceedings of the First Workshop on Information Systems and Artificial Intelligence: Integration Aspects*, volume 474 of *LNCS*, pages 30–46, New York, USA, March 1990. Springer.

[Bry90b] François Bry. Query evaluation in recursive databases: Bottom-up and top-down reconciled. *Data & Knowledge Engineering*, 5(4):289–312, 1990.

[BW93] Elena Baralis and Jennifer Widom. A rewriting technique for using delta relations to improve condition evaluation in active databases. Technical Report CS-93-1495, Department of Computer Science, Stanford University, November, 1993.

[BZF96] S. Brass, S. Zukowski, and H. Freitag. Transformation-based bottom-up computation of the well-founded model. In *Non-Monotonic Extensions of Logic Programming (NMELP '96), Selected Papers*, volume 1216 of *LNCS*, pages 171–201, Berlin, September 1996. Springer.

[BZF97] S. Brass, S. Zukowski, and H. Freitag. Differential bottom-up computation of the well-founded semantics. In *Proceedings of the 14th International Conference on Logic Programming*, pages 421–421, Cambridge, July 1997. MIT Press.

[CFPT94] Stefano Ceri, Piero Fraternali, Stefano Paraboschi, and Letizia Tanca. Automatic generation of production rules for integrity maintenance. *ACM Transactions on Database Systems (TODS)*, 19(3):367–422, September 1994.

[CG94] Richard L. Cole and Goetz Graefe. Optimization of dynamic query evaluation plans. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 150–160, New York, June 1994. ACM Press.

[CGH94] A. B. Cremers, U. Griefahn, and R. Hinze. *Deduktive Datenbanken–Eine Einführung aus der Sicht der Logischen Programmierung*. Vieweg, Braunschweig, 1994.

[CGL+96] Latha S. Colby, Timothy Griffin, Leonid Libkin, Inderpal Singh Mumick, and Howard Trickey. Algorithms for deferred view maintenance. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 469–480, New York, June 1996. ACM Press.

[CGT90] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer Verlag, Berlin, 1990.

[Cha98] Surajit Chaudhuri. An overview of query optimization in relational systems. In *Proceedings of the ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS '98)*, pages 34–43, New York, June 1998. ACM Press.

[Che93] Yangjun Chen. A bottom-up query evaluation method for stratified databases. In *Proceedings of the International Conference on Data Engineering*, pages 568–576, Los Alamitos, USA, April 1993. IEEE Computer Society Press.

[CKL+97] Latha S. Colby, Akira Kawaguchi, Daniel F. Lieuwen, Inderpal Singh Mumick, and Kenneth A. Ross. Supporting multiple view maintenance policies. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 405–416, New York, May 1997. ACM Press.

[Cla78] Keith L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322, New York, 1978. Plenum Press.

[CW90] Stefano Ceri and Jennifer Widom. Deriving production rules for constraint maintenance. In *Proceedings of the International Conference on Very Large Data Bases (VLDB '90)*, pages 566–577, Los Altos, USA, August 1990. Morgan Kaufmann.

[CW91] Stefano Ceri and Jennifer Widom. Deriving production rules for incremental view maintenance. In *Proceedings of the International Conference on Very Large Data Bases (VLDB '91)*, pages 577–589, Los Altos, USA, September 1991. Morgan Kaufmann.

[CW94] Stefano Ceri and Jennifer Widom. Deriving incremental production rules for deductive data. *Information Systems*, 19(6):467–490, 1994.

[Dec86] Hendrik Decker. Integrity enforcement on deductive databases. In *Proceedings of the 1th International Conference on Expert Database Systems (EDS '86)*, pages 381–395, Redwood City, USA, April 1986. Benjamin Cummings.

[DK89] P. M. Dung and K. Kanchanasut. A fixpoint approach to declarative semantics of logic programs. In *Proceedings of the North American Conference on Logic Programming (NACLP '89)*, pages 604–625, Cleveland, Ohio, October 1989. MIT Press.

[DS00] Guozhu Dong and Jianwen Su. Database principles – incremental maintenance of recursive views using relational calculus / SQL. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(1):44–51, 2000.

[DW86] S. W. Dietrich and D. S. Warren. Extension tables: Memo relations in logic programming. Technical Report 86/18, Department of Computer Science, SUNY at Stony Brook, July 1986.

[DW89] Subrata K. Das and M. Howard Williams. A path finding method for constraint checking in deductive databases. *Data and Knowledge Engineering*, 4:223–244, July 1989.

[GL90] Ulrike Griefahn and Stefan Lüttringhaus. Top-down integrity constraint checking for deductive databases. In *Proceedings of the 7th International Conference on Logic Programming (ICLP '90)*, pages 130–146, Jerusalem, Israel, June 1990. The MIT Press.

[GL95] Timothy Griffin and Leonid Libkin. Incremental maintenance of views with duplicates. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 328–339, New York, USA, May 1995. ACM Press.

[GL96] Michael Gertz and Udo W. Lipeck. Deriving optimized integrity monitoring triggers from dynamic integrity constraints. *Data & Knowledge Engineering*, 20(2):163–193, 1996.

[GM92] Ashish Gupta and Inderpal Singh Mumick. Magic sets transformation in nonrecursive systems. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '92)*, pages 354–367, New York, USA, June 1992. ACM Press.

[GM95] Ashish Gupta and Inderpal Singh Mumick. Maintenance of materialized views: Problems, techniques and applications. *IEEE Quarterly Bulletin on Data Engineering; Special Issue on Materialized Views and Data Warehousing*, 18(2):3–18, 1995.

[GMR95] Ashish Gupta, Inderpal Singh Mumick, and Kenneth A. Ross. Adapting materialized views after redefinitions. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 24(2):211–222, June 1995.

[GMS93] Ashish Gupta, Inderpal Singh Mumick, and V. S. Subrahmanian. Maintaining views incrementally. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 157–166, New York, USA, May 1993. ACM Press.

[GPFS02] Anastasios Gounaris, Norman W. Paton, Alvaro A. A. Fernandes, and Rizos Sakellariou. Adaptive query processing: A survey. In *Advances in Databases, Proceedings of the 19th British National Conference on Databases (BNCOD 19)*, volume 2405 of *LNCS*, pages 11–25. Springer, July 2002.

[Gri97] Ulrike Griefahn. *Reactive Model Computation–A Uniform Approach to the Implementation of Deductive Databases*. Dissertation, University of Bonn, 1997.

[GSSS91] Gösta Grahne, Seppo Sippu, and Eljas Soisalon-Soininen. Efficient evaluation for a subset of recursive queries. *Journal of Logic Programming*, 10(3,4):301–332, April 1991.

[GSUW94] Ashish Gupta, Yehoshua Sagiv, Jeffrey D. Ullman, and Jennifer Widom. Constraint checking with partial information. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '94)*, pages 45–55, New York, USA, May 1994. ACM Press.

[Han88] Jiawei Han. Selection of processing strategies for different recursive queries. In *Proceedings of the Third International Conference on Data and Knowledge Bases: Improving Usability and Responsiveness*, pages 59–68, Jerusalem, Israel, June 1988. Morgan Kaufmann.

[IN88] Tomasz Imielinski and Shamim A. Naqvi. Explicit control of logic programs through rule algebra. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '88)*, pages 103–116, Austin, Texas, March 1988. ACM Press.

[KK88] T. Kawamura and T. Kanamori. Preservation of stronger equivalence in unfold/fold logic program transformation. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, volume 2, pages 413–421, Berlin, November 1988. Springer.

[Kol91] P. G. Kolaitis. The expressive power of stratified logic programs. *Information and Computation*, 90(1):50–66, January 1991.

[KP88] Jean-Marc Kerisit and Jean-Marc Pugin. Efficient query answering on stratified databases. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 719–726, Berlin, November 1988. Springer Verlag.

[KRS90] D. Kemp, K. Ramamohanarao, and Z. Somogyi. Right-, left-, and multi-linear rule transformations that maintain context information. In *Proceedings of the International Conference On Very Large Data Bases (VLDB '90)*, pages 380–391, Palo Alto, USA, August 1990. Morgan Kaufmann.

[KSS87] Robert A. Kowalski, Fariba Sadri, and Paul Soper. Integrity checking in deductive databases. In *Proceedings of the International Confer-*

ence on Very Large Data Bases (VLDB '87), pages 61–69, Los Altos, USA, September 1987. Morgan Kaufmann.

[KSS91] David B. Kemp, Peter J. Stuckey, and Divesh Srivastava. Magic Sets and Bottom-Up Evaluation of Well-Founded Models. In *Proceedings of the 1991 International Symposium on Logic Programming*, pages 337–351, San Diego, USA, June 1991. The MIT Press.

[KSS95] David B. Kemp, Divesh Srivastava, and Peter J. Stuckey. Bottom-up evaluation and query optimization of well-founded models. *Theoretical Computer Science*, 146(1–2):145–184, July 1995.

[KT88] David B. Kemp and Rodney W. Topor. Completeness of a top-down query evaluation procedure for stratified databases. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 178–194, Seatle, USA, August 1988. The MIT Press.

[Küc91] Volker Küchenhoff. On the efficient computation of the difference between consecutive database states. In *Proceedings of Deductive and Object–Oriented Databases (DOOD '91)*, volume 566 of *LNCS*, pages 478–502, Munich, December 1991. Springer.

[LL96] S. Y. Lee and T. W. Ling. Further improvement on integrity constraint checking for stratifiable deductive databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB '96)*, pages 495–505, San Francisco, USA, September 1996. Morgan Kaufmann.

[Llo87] John W. Lloyd. *Foundations of Logic Programming (2nd Edition)*. Springer, Berlin, 1987.

[LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and D. Srivastava. Answering queries using views (extended abstract). In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '95)*, pages 95–104, New York, USA, May 1995. ACM Press.

[LR92] Nicola Leone and Pasquale Rullo. Safe computation of the well-founded semantics of Datalog queries. *Information Systems*, 17(1):17–31, 1992.

[LR01] Alexandros Labrinidis and Nick Roussopoulos. Update propagation strategies for improving the quality of data on the Web. In *Proceedings of the International Conference on Very Large Data Bases (VLDB '01)*, pages 391–400, Los Altos, USA, September 2001. Morgan Kaufmann.

[LS92] Alon Levy and Yehoshua Sagiv. Constraints and redundancy in datalog. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '92)*, pages 67–80, New York, USA, June 1992. ACM Press.

[LS93] Alon Y. Levy and Yehoshua Sagiv. Queries independent of updates. In *Proceedings of the International Conference on Very Large Data Bases (VLDB '93)*, pages 171–181, Los Altos, USA, August 1993. Morgan Kaufmann.

[LS95] Alon Y. Levy and Yehoshua Sagiv. Semantic query optimization in Datalog programs (extended abstract). In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '95)*, pages 163–173, New York, USA, May 1995. ACM Press.

[LST87] John W. Lloyd, E. A. Sonenberg, and Rodney W. Topor. Integrity constraint checking in stratified databases. *Journal of Logic Programming*, 4(4):331–343, December 1987.

[LTD95] Hongjun Lu, Kian-Lee Tan, and Son Dao. The fittest survives: An adaptive approach to query optimization. In *Proceedings of International Conference on Very Large Data Bases (VLDB'95)*, pages 251–262, Zürich, Switzerland, September 1995. Morgan Kaufmann.

[Mah88] Michael J. Maher. Equivalence of logic programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 627–658. Morgan Kaufmann, Los Altos, 1988.

[Man94] Rainer Manthey. Reflections on some fundamental issues of rule-based incremental update propagation. In *Proceedings of the International Workshop on the Deductive Approach to Information Systems and Databases*, pages 255–276, Universitat Politècnica de Catalunya (UPC), September 1994. Report de recerca LSI/94-28-R.

[Man03] Rainer Manthey. *Deduktive Datenbanken*. Folien zur Vorlesung im SS 2003, Institut für Informatik III an der Universität Bonn, 2003. http://www.cs.uni-bonn.de/~manthey/skripten.html.

[MB88] Bern Martens and Maurice Bruynooghe. Integrity constraint checking in deductive databases using a rule/goal graph. In *Proceedings of the International Conference on Expert Database Systems (EDS'88)*, pages 567–601, Vienna, Virginia, USA, April 1988. Benjamin Cummings 1989.

[MFPR90] Inderpal Singh Mumick, Sheldon J. Finkelstein, Hamid Pirahesh, and Raghu Ramakrishnan. Magic is relevant. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 247–258, Atlantic City, USA, May 1990. ACM Press.

[MFPR96] Inderpal Singh Mumick, Sheldon J. Finkelstein, Hamid Pirahesh, and Raghu Ramakrishnan. Magic conditions. *ACM Transactions on Database Systems*, 21(1):107–155, March 1996.

[MK88] Guido Moerkotte and Stefan Karl. Efficient consistency control in deductive databases. In *Proceedings of the International Conference on Database Theory (ICDT'88)*, volume 326 of *LNCS*, pages 118–128, Bruges, Belgium, August 1988. Springer.

[Mor93] Shinichi Morishita. An alternating fixpoint tailored to magic programs. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '93)*, pages 123–134, New York, USA, May 1993. ACM Press.

[MP94] Inderpal Singh Mumick and Hamid Pirahesh. Implementation of magic-sets in a relational database system. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 103–114, New York, USA, May 1994. ACM Press.

[MT99] Enric Mayol and Ernest Teniente. Addressing efficiency issues during the process of integrity maintenance. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA '99)*, volume 1677 of *LNCS*, pages 270–281, Florence, Italy, August 1999. Springer.

[MT00] Enric Mayol and Ernest Teniente. Dealing with modification requests during view updating and integrity constraint maintenance. In *Proceedings of the International Symposium on Foundations of Information and Knowledge Systems (FOIKS '00)*, volume 1762 of *LNCS*, pages 192–212, Burg, Germany, February 2000. Springer.

[Naq86] Shamim A. Naqvi. A logic for negation in database systems. In *Proceedings of the XP / 7.52 Workshop on Database Theory*, Austin, USA, August 1986. University of Texas.

[Nic82] Jean-Marie Nicolas. Logic for improving integrity checking in relational databases. *Acta Informatica*, 18(3):227–253, 1982.

[NR91] Jeffrey F. Naughton and Raghu Ramakrishnan. Bottom-up evaluation of logic programs. In Jean-Louis Lassez and Gordon Plotkin,

editors, *Computational Logic – Essays in Honor of Alan Robinson*, pages 640–700. The MIT Press, Cambridge, USA, 1991.

[NRSU89] J. F. Naughton, R. Ramakrishnan, Y. Sagiv, and J. D. Ullman. Efficient evaluation of right, left, and multi-linear rules. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, page 235, Portland, USA, May 1989. ACM Press.

[NRSU95] J. F. Naughton, R. Ramakrishnan, Y. Sagiv, and J. D. Ullman. Argument reduction by factoring. *Theoretical Computer Science*, 146(1–2):269–310, 24 July 1995.

[Oli91] Antoni Olivé. Integrity constraints checking in deductive databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB '91)*, pages 513–523, Los Altos, USA, September 1991. Morgan Kaufmann.

[Pie01] Birgit Pieper. *Inkrementelle Integritätsprüfung und Sichtenaktualisierung in SQL*. Dissertation, University of Bonn, 2001.

[Prz88] Teodor C. Przymusinski. On the declarative semantics of deductive databases and logic programming. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, Washington, D.C., 1988.

[QS87] Xiaolei Qian and Douglas R. Smith. Integrity constraint reformulation for efficient validation. In Peter M. Stocker, William Kent, and Peter Hammersley, editors, *Proceedings of the International Conference on Very Large Data Bases (VLDB '87)*, pages 417–425, Los Altos, USA, September 1987. Morgan Kaufmann.

[QW91] Xiaolei Qian and Gio Wiederhold. Incremental recomputation of active relational expressions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 3(3):337–341, September 1991.

[Ram91] Raghu Ramakrishnan. Magic templates: A spellbinding approach to logic programs. *Journal of Logic Programming*, 11(3&4):189–216, October 1991.

[RBK88] Raghu Ramakrishnan, Catriel Beeri, and Ravi Krishnamurthy. Optimizing existential datalog queries. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '88)*, pages 89–102, New York, USA, March 1988. ACM Press.

[Rei78] Raymond Reiter. On closed world databases. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 55–76, New York, 1978. Plenum Press.

[RLK86] J. Rohmer, R. Lescoeur, and J.-M. Kerisit. The Alexander method– a technique for the processing of recursive axioms in deductive databases. *New Generation Computing*, 4(3):273–285, 1986.

[Ros90] Kenneth A. Ross. Modular stratification and magic sets for DATALOG programs with negation. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '90)*, pages 161–171, Nashville, USA, April 1990. ACM Press.

[Ros91] Kenneth A. Ross. Modular acyclicity and tail recursion in logic programs. In *Proceedings of the ACM SIGACT-SIGMOD-SOGART Symposium on Principles of Database Systems (PODS '91)*, pages 92–101, New York, USA, May 1991. ACM Press.

[RS91] R. Ramakrishnan and S. Sudarshan. Top-down vs. bottom-up revisited. In *Proceedings of the 1991 International Symposium on Logic Programming (ISLP'91)*, pages 321–336, San Diego, USA, October 1991. The MIT Press.

[RSS92] Raghu Ramahrishnan, Divesh Srivastava, and S. Sudarshan. Controlling the search in bottom-up evaluation. In *Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP-92)*, pages 273–287, Washington, DC, November 1992. The MIT Press.

[RSS94] Raghu Ramakrishnan, Divesh Srivastava, and S. Sudarshan. Rule ordering in bottom-up fixpoint evaluation of logic programs. *IEEE Transactions on Knowledge and Data Engineering*, 6(4):501–517, August 1994.

[RSS96] Kenneth A. Ross, Divesh Srivastava, and S. Sudarshan. Materialized view maintenance and integrity constraint checking: Trading space for time. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(2):447–458, June 1996.

[Sag88] Yehoshua Sagiv. Optimizing DATALOG programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 659–698. Morgan Kaufmann, Los Altos, 1988.

[Sag90] Yehoshua Sagiv. Is there anything better than magic? In *Proceedings of the 1990 North American Conference on Logic Programming*, pages 235–254, Austin, USA, October 1990. The MIT Press.

[SBLC00] Kenneth Salem, Kevin Beyer, Bruce Lindsay, and Roberta Cochrane. How to roll a join: Asynchronous incremental view maintenance. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(2):129–140, 2000.

[Sek89] Hirohisa Seki. On the power of Alexander templates. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '89)*, pages 150–159, New York, USA, March 1989. ACM Press.

[SHP+96] Praveen Seshadri, Joseph M. Hellerstein, Hamid Pirahesh, T. Y. Cliff Leung, Raghu Ramakrishnan, Divesh Srivastava, Peter J. Stuckey, and S. Sudarshan. Cost-based optimization for magic: Algebra and implementation. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(2):435–446, June 1996.

[SI88] Hirohisa Seki and Hidenori Itoh. A query evaluation method for stratified programs under the extended CWA. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 195–211, Seatle, USA, August 1988. The MIT Press.

[SJGP90] M. Stonebraker, A. Jhingran, J. Goh, and S. Potamianos. On Rules, Procedures, Caching and Views in Database Systems. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 281–290, New York, May 1990. ACM Press.

[SK88] F. Sadri and R. A. Kowalski. A theorem proving approach to database integrity. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 313–362. Morgan Kaufmann, Los Altos, USA, 1988.

[SMK97] Michael Steinbrunn, Guido Moerkotte, and Alfons Kemper. Heuristic and randomized optimization for the join ordering problem. *The VLDB Journal*, 6(3):191–208, 1997.

[SNV95] V. S. Subrahmanian, Dana Nau, and Carlo Vago. WFS + Branch and Bound = Stable Models. *IEEE Transactions on Knowledge and Data Engineering*, 7(3):362–377, June 1995.

[SSS90] Seppo Sippu and Eljas Soisalon-Soininen. Multiple SIP strategies and bottom-up adorning in logic query optimization. In *Proceedings of the International Conference on Database Theory (ICDT'90)*, volume 470 of *LNCS*, pages 485–498, Paris, France, December 1990. Springer.

[Sud92] S. Sudarshan. *Optimizing Bottom-Up Query Evaluation for Deductive Databases*. Dissertation, University of Wisconsin-Madison, 1992.

[SZ87a] Domenico Saccà and Carlo Zaniolo. Implementation of recursive queries for a data language based on pure Horn logic. In *Proceedings of the Fourth International Conference on Logic Programming*, pages 104–135, Melbourne, Australia, May 1987. The MIT Press.

[SZ87b] Domenico Saccà and Carlo Zaniolo. Magic counting methods. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pages 49–59, San Francisco, USA, May 1987. ACM Press.

[TO95] Ernest Teniente and Antoni Olivé. Updating knowledge bases while maintaining their consistency. *VLDB Journal*, 4(2):193–241, April 1995.

[TS86] Hisao Tamaki and Taisuke Sato. Old resolution with tabulation. In *Proceedings of the Third International Conference on Logic Programming*, volume 225 of *LNCS*, pages 84–98, London, July 1986. Springer.

[Ull85] Jeffrey D. Ullman. Implementation of logical query languages for databases (abstract). In *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data*, Austin, USA, May 1985. ACM Press.

[Ull89] Jeffrey. D. Ullman. *Principles of Database and Knowledge-base Systems*, volume II. Computer Science Press, Rockville, Maryland, 1989.

[UO92] Toni Urpí and Antoni Olivé. A method for change computation in deductive databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB '92)*, pages 225–237, Los Altos, USA, August 1992. Morgan Kaufmann.

[UO94] Toni Urpí and Antoni Olivé. Semantic Change Computation Optimization in Active Databases. In *Proceedings of the 4th International Workshop on Research Issues in Data Engineering - Active Database Systems*, pages 19–27, Houston, USA, Februar 1994. IEEE Computer Society Press.

[VBK91] Laurent Vieille, Petra Bayer, and Volker Küchenhoff. Integrity checking and materialized views handling by update propagation in the EKS-V1 system. Technical Report TR-KB-35, ECRC, München, June 1991.

[vEK76] M. H. van Emden and R. Kowalski. The Semantics of Predicate Logic as Programming Language. *Journal of ACM*, 23(4):733–743, 1976.

[vG88] Allen van Gelder. Negation as failure using tight derivation for general logic programs. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, Los Altos, USA, 1988.

[vG89] Allen van Gelder. The alternating fixpoint of logic programs with negation. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '89)*, pages 1–10, New York, USA, March 1989. ACM Press.

[vG93] Allen van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185–221, August 1993.

[vGRS88] Allen van Gelder, Kenneth Ross, and John S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '88)*, pages 221–230, New York, USA, March 1988. ACM Press.

[vGRS91] Allen van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, July 1991.

[Vie88] Laurent Vieille. From qsq towards qosaq: Global optimization of recursive queries. In *Proceedings of the Second International Conference on Expert Database Systems (EDS '88)*, pages 743–778. Benjamin Cummings 1989, Vienna, USA, April 1988.

[VM96] Bennet Vance and David Maier. Rapid bushy join-order optimization with Cartesian products. In *Proceedings of theACM SIGMOD International Conference on Management of Data*, pages 35–46, New York, USA, June 1996. ACM Press.

[Wüt90] B. Wüthrich. Detecing inconsistencies in deductive databases. Technical Report 1990TR-123, Swiss Federal Institute of Technology (ETH), Zürich, January 1990.

Index

Curriculum Vitae

Andreas Behrend
Dorotheenstraße 161
53119 Bonn

Geboren am 30. Mai 1972 in Rostock
Familienstand: ledig

1978 - 1988	Polytechnische Oberschule in Rostock
1988 - 1990	Erweiterte Oberschule in Rostock
Juni 1990	Abitur
1990 - 1991	Wehrdienst
1991 - 1994	Informatikstudium mit Nebenfach Wirtschaftswissenschaften an der Universität Rostock
1994 - 1995	Informatikstudium an der Universität Aberdeen
1995 - 1999	Informatikstudium mit Nebenfach Wirtschaftswissenschaften an der Universität in Bonn
Februar 1999	Einreichung der Diplomarbeit *"Effiziente Materialisierung regeldefinierter Daten in PROLOG"*
Februar 1999	Abschluß des Studiums
seit März 1999	Wissenschaftlicher Mitarbeiter im Institut für Informatik III an der Universität Bonn