Efficient Computation and Maintenance of Datalog Materialisations

Pan Hu

Worcester College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2019

Acknowledgements

I would like to first express my heartfelt gratitude to my supervisors, Prof. Boris Motik and Prof. Ian Horrocks, without whom this thesis would not have been possible. I thank Boris for his guidance and support over the past four years; passionate about science and attentive to details, he has set a good example for me. I thank Ian for always being able to find time to look at my papers; his encouragement has been invaluable to me, especially when I struggled with my research. I consider myself very fortunate to have studied under their supervision.

My thanks also go to Prof. Bernardo Cuenca Grau and Dr. Egor V. Kostylev, who provided excellent tutoring in my first year and have been my examiners in many occasions. In addition, I wish to thank Prof. Michael Zakharyaschev for taking the time to examine my thesis, and for his detailed and helpful comments. I am further grateful to Alessandro, Alina, Mark, Yavor, and other colleagues in the KRR Group at Oxford. Our conversations over lunch breaks have always been interesting and inspiring.

Finally, I am deeply indebted to my family, who were worried about me pursuing my studies in a place thousands of miles away from home, but have supported me nonetheless. Their unconditional love and support have been a great source of motivation for me. I would also like to thank Yue for always being by my side. She inspires me everyday with her enthusiasm about study, research, and life in general.

Abstract

Datalog is a prominent knowledge representation language whose popularity is mainly due to its ability to express recursive definitions. Datalog applications typically require efficient reasoning over datalog programs and facts. To support this, datalog systems often materialise all consequences of a datalog program so that all queries can later be evaluated directly over the materialisation. This style of reasoning is usually complemented with an incremental maintenance algorithm that updates the materialisation whenever the input facts change. Existing solutions to the incremental maintenance problem either handle only nonrecursive rules or contain steps that evaluate rules "backwards" by matching their heads to a fact and evaluating the partially instantiated rule bodies as queries. We show that this can be a considerable source of overhead even on very small updates, and we present two hybrid approaches that reduce or even eliminate "backward" rule evaluation while still handling arbitrary datalog programs. Moreover, we observe that for both materialisation and incremental maintenance, certain (combinations of) rules can be handled much more efficiently using custom algorithms. To integrate such algorithms into a general reasoning approach that can handle arbitrary rules, we propose a modular framework for computing and maintaining a materialisation. We split a datalog program into modules that can be handled using specialised algorithms, and we handle the remaining rules using the seminaïve evaluation strategy as in most existing solutions. We also present two algorithms for computing the transitive and the symmetric–transitive closure of a relation that can be used within our framework. Finally, we show empirically that the proposed solutions are usually significantly faster than existing approaches, sometimes by orders of magnitude.

Publications

Pan Hu, Boris Motik, and Ian Horrocks. "Optimised Maintenance of Datalog Materialisations". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018, pp. 1871–1879.

Pan Hu, Boris Motik, and Ian Horrocks. "Modular Materialisation of Datalog Programs". In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. 2019, pp. 2859–2866.

Pan Hu, Jacopo Urbani, Boris Motik, and Ian Horrocks. "Datalog Reasoning over Compressed RDF Knowledge Bases". In: *Proceedings of the Twenty-Eighth ACM International Conference on Information and Knowledge Management*. 2019.

Pan Hu, Boris Motik, and Ian Horrocks. "Modular Materialisation of Datalog Programs". Extended version of the AAAI-19 paper to be submitted to *IEEE Transactions on Knowledge and Data Engineering*

Contents

List of Tables **vii**

1 Introduction **1**

1.1 Datalog Materialisation 2

1.2 Incremental Materialisation Maintenance 3

1.3 Research Problems 3

1.4 Open Challenges 4

1.5 Contributions 6

2 Preliminaries **9**

3 Related Work **13**

3.1 Overview 13

3.2 The Seminaïve Algorithm 16

3.3 The Delete/Rederive Algorithm 19

3.4 The Backward/Forward Algorithm 24

3.5 The Counting Algorithm 26

4 Optimised Incremental Maintenance **28**

4.1 Motivation and Intuition 28

4.2 Combining DRed with Counting 31

4.2.1 Intuition 31

4.2.2 Formalisation 33

4.3 Combining B/F with Counting 36

4.4 Evaluation 39

5 Modular Materialisation and Maintenance 44

5.1 Motivation 44

5.2 Framework 46

5.2.1 Module Functions 47

5.2.2 Computing the Materialisation 51

5.2.3 Incremental Updates 52

5.2.4 Correctness Conditions for Module Functions 54

5.3 Further Optimisation 59

5.4 Transitive Closure 60

5.5 Symmetric–Transitive Closure 61

5.6 Evaluation 63

6 Conclusion and Outlook 67

Appendices

A Proof of Theorems in Chapter 4 71

A.1 Proof of Theorem 4 71

A.2 Proof of Theorem 5 84

B Proof of Theorems in Chapter 5 86

B.1 Proof of Theorem 13 86

B.2 Proof of Theorem 14 92

B.3 Proof of Theorem 18 112

B.4 Proof of Theorem 20 117

B.5 Proof of Theorem 21 138

References 160

List of Tables

4.1 Benchmark statistics 40
4.2 Average running times for deleting 1000 facts (seconds) 40
4.3 Running times for handling large deletions (seconds) 41
4.4 Running times for rematerialisation (seconds) 41

5.1 Module function calls 49
5.2 Correctness of module function calls 49
5.3 A partition of the program Π 50
5.4 Conditions on the structure of a call sequence 57
5.5 Benchmark statistics 64
5.6 Running times for materialisation computation (seconds) 64
5.7 Running times for incremental maintenance (seconds) 64

1 Introduction

Datalog [1] is a prominent rule language that has been extensively studied and used across several communities over the past decades. In the database community, datalog has been studied and implemented as a recursive query language [5, 12, 30, 62]. In the artificial intelligence community, datalog and its various extensions have been examined for understanding their complexity and expressive power as knowledge representation formalisms [23, 24, 41]. Datalog programs can be evaluated in time polynomial in the size of the input data, which makes datalog a suitable language for reasoning over large-scale data. Moreover, datalog captures OWL 2 RL [54] ontologies extended with SWRL rules [37], so datalog-based implementations can be used in many Semantic Web applications that require answering queries over such ontologies. An eminent feature of datalog is its ability to declaratively model domain knowledge of an application as 'if-then' rules, over which implicit facts can be inferred. Despite its simple syntax and straightforward semantics, datalog is rich enough to express many useful recursive queries such as reachability and transitive closure.

Thanks to its favourable computational properties and its ability to declaratively express complex definitions, datalog has recently found application in a wide range of practical scenarios. For example, datalog can be used to support succint expression and efficient execution of network protocols [45–47]. In the healthcare domain, datalog has

been used to generate a complex set of quality measures from healthcare records [58]. In the area of distributed computing, Datalog has been extended and implemented as a data analytics language [2, 3]. It has also been used as a specification language for information extraction tasks over both unstructured and semi-structured data [27, 64]. In addition, variants of datalog have been used as languages for specifying program analysis tasks [11, 73], and security policies [19, 40]. The growing interest in datalog has motivated the development of many datalog systems, such as Oracle's database [75], RDFox [55], LogicBlox [4], VLog [68], Vadalog [7], GraphDB [28], and Datomic [17].

1.1 Datalog Materialisation

A key computational task that datalog systems have to perform is to answer queries over consequences derived from a set of datalog rules and a set of explicitly given facts. To perform such a task, one common approach is to precompute and store all derived facts so that queries can be directly evaluated over these facts without consulting the program. This process, as well as its output, are called *materialisation*. Materialisation-based query answering has been implemented in many systems, including but not limited to WebPIE [69], VLog, Oracle's RDF Store, OWLIM [8], and RDFox.

A naïve approach to computing the materialisation is to repeatedly apply the datalog rules to the given facts and the facts derived so far, until no new fact can be derived. However, this can be very inefficient since each round of rule application has to repeat all inferences performed in the previous rounds. Numerous algorithms and optimisation techniques have thus been proposed to avoid such redundant inferences. Most of them are based on the seminaïve evaluation strategy [1]. In each iteration of rule application, the seminaïve algorithm considers only inferences involving facts newly derived in the previous iteration. If implemented carefully, the seminaïve algorithm completely avoids redundant inferences in the sense that no inference is considered more than once [52]. Technical details of the seminaïve algorithm are provided in Section 3.2.

1.2 Incremental Materialisation Maintenance

The main drawback of materialisation-based query answering is that the materialisation must be updated when the input facts change. Doing so 'from scratch' can be inefficient if changes are small. Systems thus typically use an *incremental maintenance algorithm*, which aims to avoid repeating most of the work. Fact insertion can be effectively handled using the seminaïve algorithm [1], but deletion is much more involved since one has to check whether deleted facts have derivations that persist after the update. A number of incremental maintenance algorithms have been proposed, among which the *Delete/Rederive* (DRed) algorithm [35, 65], the *Backward/Forward* (B/F) algorithm [51], and the *Counting* algorithm [35] are representative. We will present these algorithms and briefly survey their variants in Chapter 3.

The DRed algorithm handles deletion by first overdeleting all facts that depend on the removed explicit facts, and then rederiving the facts that still hold after overdeletion. The rederivation stage further involves rederiving all overdeleted facts that have alternative derivations, and then recomputing the consequences of the rederived facts until a fixpoint is reached. In contrast to DRed, the B/F algorithm searches for alternative derivations immediately, rather than after overdeletion, using a combination of backward and forward chaining. This makes deletion exact and avoids the potential inefficiency of overdeletion (at the cost of eager search for alternative derivations). In practice, B/F outperforms DRed in many cases.

1.3 Research Problems

In this thesis we study two research problems, i.e., the problem of computing the materialisation of a program over a set of given facts, and the problem of maintaining the materialisation of a program in response to changes in the given facts. Both problems are of great practical importance. Materialisation computation helps speed up query answering by precomputing all consequence of the program and the given facts, and this is only possible when the materialisation can be correctly maintained. Hence, as the performance of query answering is critical in many datalog applications, solutions to

both problems have already been implemented in many datalog systems. Please refer to Section 2 for a formalisation of these two problems.

1.4 Open Challenges

We next briefly outline several open challenges that remain to be addressed for datalog materialisation computation and maintenance. These challenges will be discussed in more detail with concrete examples later in Sections 4.1 and 5.1.

Challenges for Datalog Materialisation

Although the seminaïve algorithm does not repeat inferences, it has to consider all the applicable ones. However, for certain rules or rule combinations, considering all applicable inferences is not necessary and there exist better algorithms for computing all the consequences. For example, for a datalog program that axiomatises a relation as transitive and symmetric and a set of facts that describe a connected graph with n vertices, computing all consequences can be done straightforwardly by first identifying the connected components of the graph and then enumerating each pair of nodes belonging to the same component. These operations can be performed in $O(n^2)$ time. In contrast, the seminaïve algorithm requires $O(n^3)$ time to consider all applicable inferences since the rule that axiomatises a relation as transitive has three variables, each of which can be mapped to one of the n available constants. As another example, Nuutila's algorithm [57] for transitive closure computation can be used to efficiently materialise all consequences of a rule aximotising transitivity [67]. Despite such custom algorithms, it remains unclear whether and how they can be used in general-purpose datalog systems that are capable of materialising arbitrary datalog programs.

Challenges for Incremental Materialisation Maintenance

Both DRed and B/F search for derivations of deleted facts by evaluating rules 'backwards': for each rule whose head matches the fact being deleted, they evaluate the partially instantiated rule body as a query; each query answer thus corresponds to a derivation. This has two consequences. First, one can examine derivations that hold both before

and after the update, which is redundant. Second, evaluating rules 'backwards' can be inherently more difficult than matching the rules during initial materialisation: our experiments show that this step can, in some cases, prevent effective materialisation maintenance even for very small updates.

In contrast, the Counting algorithm [35] does not evaluate rules 'backwards', but instead maintains the number of distinct derivations of each fact. A fact is deleted when its counter drops to zero (i.e., no derivation for the fact still holds). The algorithm is in a sense 'optimal' on nonrecursive rules since it only considers derivations affected by the update and never examines derivations that hold both before and after the update (as opposed to DRed and B/F which do have to examine such derivations). The main drawback of Counting is that, unlike DRed and B/F, it lacks generality and is applicable only to nonrecursive rules [56].

In addition to the above issues, existing incremental maintenance algorithms also suffer from deficiencies of the seminaïve algorithm since these algorithms all apply certain variants of the seminaïve evaluation strategy. More specifically, they can be suboptimal for certain rules and rule combinations. Indeed, approaches that can maintain the closure of specific datalog programs have been already considered in the literature. For example, maintaining transitive closure of a graph has been studied extensively [18, 39, 43, 59]. In the database context, a transitively closed relation can be updated in response to insertions by evaluating four nonrecursive first-order queries [21]. However, these approaches can only handle datalog programs for which they have been specifically developed—that is, the programs are not allowed to contain any additional rules. The presence of other rules introduces additional complexity since updates computed by specialised algorithms must be propagated to the remaining rules and vice versa. Thus, it is currently not clear whether and how customised algorithms can be used in general-purpose datalog systems to support incremental additions and deletions of facts for arbitrary programs.

1.5 Contributions

To address the inefficiencies of the existing solutions, in this thesis we present optimised solutions for both materialisation computation and incremental maintenance. Our algorithms are not dependent on any specific data structure, which makes it easy for them to be implemented either from scratch or on top of existing systems. In fact, we have implemented our solutions on top of the RDFox system, and the extended systems are available online.[1][2] In our experiments, the proposed algorithms perform consistently better than their unoptimised counterparts in a number of Semantic Web benchmarks. Thus, our algorithms can be considered as an important addition to the toolbox of techniques available to the datalog developers in the Semantic Web community. Furthermore, while our implementation and tests have been carried out with RDFox, which deals mainly with Semantic Web data, the proposed algorithms are general purpose and are readily amenable to implementations in a wide range of application scenarios. Applications of datalog in areas such as distributed computing, information extraction, and program analysis often require efficient evaluation of datalog programs, so we believe that our techniques could potentially be of interest. Below we summarise our technical contributions.

First, in Chapter 4 we present two hybrid approaches to incremental materialisation maintenance, DRed^c and B/F^c, which combine DRed and B/F with Counting, respectively. The DRed^c algorithm tracks the nonrecursive and the recursive derivations separately, which allows the algorithm to eliminate all 'backward' rule evaluation and also limit overdeletion. The B/F^c algorithm tracks nonrecursive derivations only, which eliminates 'backward' rule evaluation for nonrecursive rules; however, recursive rules can still be evaluated 'backwards' to eagerly identify alternative derivations. Both combinations can handle recursive rules, and they exhibit 'pay-as-you-go' behaviour in the sense that they become equivalent to Counting on nonrecursive rules. Apart from the modest cost of maintaining counters, our algorithms never involve more computation steps than their unoptimised counterparts. Thus, our algorithms combine the best aspects of DRed, B/F,

[1] `http://krr-nas.cs.ox.ac.uk/2017/counting/`
[2] `http://krr-nas.cs.ox.ac.uk/2018/modular/`

and Counting: with a modest cost, they eliminate or reduce 'backward' rule evaluation, are optimal for nonrecursive rules, and can also handle recursive rules. Furthermore, we have implemented our hybrid algorithms and have compared them with the original DRed and B/F algorithms on several synthetic and real-life benchmarks. Our experiments confirm that the cost of counter maintenance is negligible, and that our hybrid algorithms typically outperform existing solutions, sometimes by orders of magnitude.

Second, in Chapter 5 we present a modular framework for the computation of datalog materialisations that can integrate specialised reasoning algorithms with the seminaïve evaluation. The framework partitions the rules of a datalog program into disjoint subsets called *modules*. For each module, a function computing the consequences of the module's rules must be made available; there are no restrictions on how this function is implemented, as long as its outputs satisfy certain conditions. Moreover, if no specialised algorithm for a module is available, the function can be implemented using seminaïve evaluation. Thus, our framework can efficiently handle certain combinations of rules, but it can simultaneously handle arbitrary rules as well while avoiding repeated inferences.

Third, building upon the DRed^c algorithm, we extend the modular framework for modularised materialisation computation to handle incremental updates. More specifically, in addition to the insertion function required by the modular materialisation framework, two more functions responsible for computing other types of consequences need to be made available for each module. Again, the implementation details of these functions are not restricted as long as they satisfy certain conditions necessary for the correctness of the framework.

Fourth, we examine two specific kinds of module axiomatising the transitive and the symmetric–transitive closure of a relation. These capture node reachability in directed and undirected graphs, respectively, both of which frequently occur in practice and are thus highly relevant. We present custom algorithms that implement the functions required by our framework and show that these algorithms satisfy the general correctness properties. Moreover, we discuss the kinds of input on which one can expect significant

performance improvements. Finally, our experiments show that our modular approach is often orders of magnitude faster than DRed^c and B/F^c.

2
Preliminaries

We introduce datalog with stratified negation. A *term* is a constant or a variable. An *atom* has the form $P(t_1, \ldots, t_k)$, where P is a k-ary *predicate* with $k \geq 0$, and each t_i, $1 \leq i \leq k$, is a term. A *fact* is a variable-free atom, and a *dataset* is a finite set of facts. A *negative literal* has the form $\text{not}\, B$ for B an atom. A *literal* is an atom or a negative literal. A rule r has the form

$$B_1 \wedge \cdots \wedge B_m \wedge \text{not}\, B_{m+1} \wedge \cdots \wedge \text{not}\, B_n \rightarrow H,$$

where $m \geq 0$, $n \geq 0$, and B_i and H are atoms. For r a rule, $\mathsf{h}(r) = H$ is the *head*, $\mathsf{b}(r) = B_1 \wedge \cdots \wedge \text{not}\, B_n$ is the *body*, $\mathsf{b}^+(r) = \{B_1, \ldots, B_m\}$ is the set of *positive body atoms*, and $\mathsf{b}^-(r) = \{B_{m+1}, \ldots, B_n\}$ is the set of *negative body atoms*. Each rule r must be *safe*: each variable occurring in r must occur in at least one positive body atom. A *program* is a finite set of rules. A program Π is *semipositive* iff no predicate in the head of a rule in Π occurs in $\bigcup_{r \in \Pi} \mathsf{b}^-(r)$.

A *stratification* λ of a program Π maps each predicate occurring in Π to a positive integer such that, for each rule $r \in \Pi$ with predicate P in its head, $\lambda(P) \geq \lambda(R)$ holds for each predicate R occurring in $\mathsf{b}^+(r)$, and $\lambda(P) > \lambda(R)$ holds for each predicate R occurring in $\mathsf{b}^-(r)$. Such r is *recursive* with regards to λ if $\lambda(P) = \lambda(R)$ holds for some predicate R occurring in $\mathsf{b}^+(r)$; otherwise, r is *nonrecursive* with regards to λ. Program

Π is *stratifiable* if a stratification λ of Π exists. For s an integer, the *stratum* s of Π is the program Π^s containing each rule $r \in \Pi$ whose head predicate P satisfies $\lambda(P) = s$. Moreover, let Π^s_{r} and Π^s_{nr} be the recursive and the nonrecursive subsets, respectively, of Π^s. Finally, let $\mathsf{O}^s = \{P(c_1, \ldots, c_n) \mid \lambda(P) = s\}$. In other words, O^s is the (infinite) set of all facts whose predicate P satisfies $\lambda(P) = s$.

A *substitution* σ is a mapping of finitely many variables to constants. For α a term, an atom, a rule, or a set thereof, $\alpha\sigma$ is the result of replacing each occurrence of a variable x in α with $\sigma(x)$, if the latter is defined.

If r is a rule and σ is a substitution mapping all variables of r to constants, then rule $r\sigma$ is an *instance* of r; moreover, we say that fact $\mathsf{h}(r\sigma)$ is *derived* by rule instance $r\sigma$, and that $r\sigma$ is a *derivation* for $\mathsf{h}(r\sigma)$. For I a dataset, we define the set $\mathsf{inst}_r\Big[I\Big]$ of instances of r obtained by applying a rule r to I, and the set $\Pi\Big[I\Big]$ of facts obtained by applying a program Π to I as follows. Each rule instance considered in computing $\Pi\Big[I\Big]$ is said to be *applicable* to I.

$$\mathsf{inst}_r\Big[I\Big] = \{r\sigma \mid \mathsf{b}^+(r\sigma) \subseteq I \text{ and } \mathsf{b}^-(r\sigma) \cap I = \emptyset\} \tag{2.1}$$

$$\Pi\Big[I\Big] = \bigcup_{r \in \Pi} \{\mathsf{h}(r') \mid r' \in \mathsf{inst}_r\Big[I\Big]\}. \tag{2.2}$$

A *query* is a conjunction of literals

$$q = B_1 \wedge \cdots \wedge B_m \wedge \text{not } B_{m+1} \wedge \cdots \wedge \text{not } B_n$$

where $m \geq 0$ and $n \geq 0$. For q a query, $\mathsf{b}^+(q) = \{B_1, \ldots, B_m\}$ is the set of *positive query atoms*, and $\mathsf{b}^-(q) = \{B_{m+1}, \ldots, B_n\}$ is the set of *negative query atoms*. The safety condition is defined in a similar way as for rules: each variable occuring in q must also occur in at least one of its positive query atoms $\mathsf{b}^+(q)$. Given a dataset I and a query q, a substitution σ is an *answer* of evaluating q over I iff σ maps all variables in q to constants, and $\mathsf{b}^+(q\sigma) \subseteq I$ and $\mathsf{b}^-(q\sigma) \cap I = \emptyset$ both hold.

It is easy to see that $\mathsf{inst}_r\Big[I\Big] = \{r\sigma \mid \sigma \text{ is an answer of evaluating } \mathsf{b}(r) \text{ over } I\}$, and this shows us how $\Pi\Big[I\Big]$ can be computed. For each rule $r \in \Pi$, we evaluate the body of the rule as a query over I, and for each query answer σ, we add $\mathsf{h}(r\sigma)$ to the result

set. Throughout the thesis we assume that rules are evaluated this way; in addition, we assume that all facts are indexed in some way, and it takes $O(1)$ time to check if a fact belongs to a set or to add a fact to a set. Now let $n = |I|$ be the number of facts in I, and let $m = |\Pi|$ be the number of rules in Π; moreover, let k be the largest possible number of distinct variables in the body of one rule in Π, let u be the largest possible arity of a predicate that occurs in Π, and let w be the largest possible number of atoms in the body of a rule. We are now ready to analyse the cost of evaluating $\Pi\Big[I\Big]$. There are a total number of $u \cdot n$ constants available in I; for each rule $r \in \Pi$, we evaluate its body as a query over I by iterating through at most $u^k \cdot n^k$ substitutions; for each of these substitutions, it takes $O(w)$ time to check if the instantiated query atoms satisfy the relevant conditions and to add the instantiated head to the result set if the conditions are indeed satisfied. There are in total m rules to be considered, so the overall cost is $O(m \cdot w \cdot u^k \cdot n^k)$.

We are now ready to formalise the two computational problems that this thesis focuses on. Let E be a dataset of *explicit facts* and let λ be a stratification of Π with maximum stratum index S. Then, we define the following sequence of datasets:

- let $I^0_\infty = E$;
- for each $1 \leq s \leq S$,
  - let $I^s_0 = I^{s-1}_\infty$,
  - for each $i \geq 1$, let $I^s_i = I^s_{i-1} \cup \Pi^s\Big[I^s_{i-1}\Big]$, and
  - let $I^s_\infty = \bigcup_{i \geq 0} I^s_i$.

Set I^S_∞ is the *materialisation* of Π with regards to E and λ. It is known that I^S_∞ does not depend on λ, so we write it as $\mathsf{mat}(\Pi, E)$. Computing the materialisation $\mathsf{mat}(\Pi, E)$ given Π and E is the first problem that we are interested in.

In this thesis we also study solutions for the problem of updating a materialisation in response to changes in the explicit facts. More specifically, given a program Π, a set of explicit facts E, the materialisation $I = \mathsf{mat}(\Pi, E)$, and sets of facts E^- and E^+ to remove from and to add to E, respectively, the problem of *materialisation maintenance*

is to compute the new materialisation $I' = \mathsf{mat}(\Pi, (E \setminus E^-) \cup E^+)$. A naïve approach to this problem is to recompute the materialisation 'from scratch' using a standard materialisation algorithm such as the seminaïve algorithm. However, such an approach is very inefficient especially when E^- and E^+ are small compared to E: much of the work performed during the initial materialisation will be repeated. Consequently, in this thesis we focus on various *incremental* maintenance algorithms which aim at efficiently computing I' by taking advantage of what has already been computed in I.

3
Related Work

In this chapter we review existing solutions for datalog materialisation computation, incremental maintenance, and other related problems. Moreover, we discuss in detail several existing algorithms which lay the foundation for our work. This chapter is organised as follows. Section 3.1 provides an overview of existing approaches. Section 3.2 presents details of the seminaïve algorithm, which underpins our modular materialisation algorithm. The rest of the chapter formalises three important algorithms, *Delete/Rederive*, *Backward/Forward*, and *Counting*, which form the basis of our optimised algorithms and modular framework for incremental maintenance.

3.1 Overview

In this section we briefly survey existing solutions for related problems. It should be noted that the problem of datalog materialisation maintenance can be seen as a special case of *view maintenance*. Views are defined by queries over base relations and are often materialised so that they could be efficiently accessed just as normal relations. When base relations change, view maintenance algorithms compute only the incremental changes to the views, rather than reevaluate the queries that define them. In our work, datalog materialisations correspond to materialised views defined by queries written in the datalog language. There is a large body of work on view maintenance, so for clarity

we classify existing techniques according to whether the language that defines the views is recursive. For the recursive group, which is more relevant to our work, we further divide the relevant techniques into two categories based on whether additional information, such as fact counts, has to be maintained.

Datalog Materialisation Computation

Bancilhon [6] proposed the seminaïve evaluation strategy for evaluating recursively defined relations. Abiteboul, Hull, and Vianu [1] presented an improved version of this method for the bottom-up evaluation of datalog programs. Ramakrishnan, Srivastava, and Sudarshan [61] studied the impact of rule orderings on the performance of seminaïve evaluation, and proposed two variants of the seminaïve algorithm that are capable of handling user-defined rule orderings, although it was unclear how promising rule orderings could be automatically obtained. Ramakrishnan et al. [62] have also implemented the seminaïve evaluation strategy in their datalog system CORAL. Ganguly, Silberschatz, and Tsur [25] and Zhang, Wang, and Chau [77] considered parallelising seminaïve evaluation of datalog programs by statically distributing rule instantiations to multiple processors. In contrast, Motik et al. [53] implemented a parallel variant of the seminaïve algorithm that achieved dynamic partition of rule instantiations, which was less susceptible to skewed workload distribution. Urbani, Jacobs, and Krötzsch [68] and Hu et al. [38] presented adaptations of the seminaïve algorithm for the materialisation of datalog programs over column-oriented RDF data.

Maintenance of Nonrecursive Views

Blakeley, Larson, and Tompa [10] presented a derivation counting method for incrementally updating materialised views defined by Select-Project-Join (SPJ) expressions. The idea is to maintain for each tuple a counter which tracks the number of derivations for the tuple. The counter gets incremented by one when a new derivation becomes available, and it gets decremented by one when an existing derivation no longer holds. A tuple could thus be safely deleted from the view when its counter drops to zero. Hanson [36] presented a refined version of the algorithm and compared the performance of counting to the performance of view rematerialisation with a cost analysis. Nicolas and Yazdanian

[56] and Gupta, Mumick, and Subrahmanian [35] presented similar counting approaches for maintaining views defined by nonrecursive datalog programs, which we will discuss in more detail in Section 3.5.

Ceri and Widom [13] presented an approach that incrementally maintains duplicate-free views by using production rules automatically generated from view definitions. Qian and Wiederhold [60] considered relational algebra with set semantics and presented a method that derives the minimal incremental relational expressions to be evaluated based on relational expressions formed by the initial base relations and updates to these relations. Griffin, Libkin, and Trickey [32] showed with a counter-example that this method does not always compute the minimal expressions, and they provided an improved version that preserves minimality. Griffin and Libkin [31] also extended this idea to relational algebra with bag semantics, which then allows aggregate functions to be incrementally computed. Compared with algorithmic solutions, a key advantage of such algebraic approaches is that changes to the views are represented by expressions that can be optimised as queries. As an example, Vista [71, 72] implemented a query optimiser that extended standard query optimisation techniques to support maintenance queries.

Maintenance of Recursive Views with Bookkeeping

Gupta, Katiyar, and Mumick [34] extended the counting approaches by Nicolas and Yazdanian [56] and Gupta, Mumick, and Subrahmanian [35] to handle recursive datalog rules. However, this approach might not work correctly in cases where a fact recursively derives itself. Wolfson et al. [74] proposed a counting-based algorithm that overcomes the above problem and is capable of handling recursion correctly for arbitrary datasets. Dewan et al. [20] reformulated this algorithm and proved its correctness. The main idea is to maintain for each fact multiple counters instead of just one. The ith counter of a fact tracks the number of times the fact is derived in the ith iteration of the initial materialisation process. One major drawback of this algorithm is that it is devised based on the naïve evaluation strategy for datalog, which is inherently inefficient. Motik et al. [52] described an optimised variant of this algorithm that is based on the more efficient seminaïve evaluation strategy.

Maintenance of Recursive Views without Bookkeeping

Gupta, Mumick, and Subrahmanian [35] proposed the Delete/Rederive (DRed) algorithm that update views defined by general datalog programs. To handle fact deletion (which is usually more difficult than fact insertion), the algorithm first deletes all consequences of the deleted facts and then uses what remains to rederive facts that still hold after the update. Staudt and Jarke [65] presented a very similar algorithm, but formalised it declaratively with maintenance rules. Motik et al. [52] identified several inefficiencies of the DRed algorithm by Gupta, Mumick, and Subrahmanian [35] and Staudt and Jarke [65], and presented an optimised version that addressed these issues. In Section 3.3 we discuss this version of the algorithm in more detail.

Motik et al. [51] proposed the Backward/Forward (B/F) algorithm for the incremental maintenance of datalog materialisations. During the deletion phase, instead of computing an overestimation of the deleted facts as in the case of DRed, the B/F algorithm eagerly searches for alternative derivations of facts so that deletion can be made exact. We provide details of this algorithm in Section 3.4. Motik et al. [52] further presented the Forward/Backward/Forward (FBF) algorithm, which could be viewed as a generalisation of both DRed and B/F algorithms.

3.2 The Seminaïve Algorithm

Constructing the materialisation $\mathsf{mat}(\Pi, E)$ based on its definition in Section 2 can be very inefficient. Consider an arbitrary stratification λ of Π with maximum stratum index S, an arbitrary stratum index s with $1 \leq s \leq S$, and an arbitrary $i > 1$, to compute I_i^s, the program application $\Pi^s\left[I_{i-1}^s\right]$ will repeat all applicable rule instances previously considered to compute I_j^s with $1 \leq j < i$ (i.e., rule instances already considered by $\Pi^s\left[I_{i-2}^s\right], \ldots, \Pi^s\left[I_0^s\right]$). In contrast, the seminaïve algorithm [1], formalised in Algorithm 1, eliminates this major source of inefficiency by considering each applicable rule instance only once.

Algorithm 1 takes as input a set of explicit facts E, a program Π, and a stratification λ of Π, and it computes $\mathsf{mat}(\Pi, E)$. To apply each applicable rule instance only once, it

Algorithm 1 SEMINAÏVE(Π, λ, E)

1: $I := \emptyset$
2: **for each** stratum index s with $1 \leq s \leq S$ **do**
3: $\quad \Delta := (E \cap \mathsf{O}^s) \cup \Pi^s_{\mathsf{nr}}\big[I\big]$
4: $\quad$ **while** $\Delta \neq \emptyset$ **do**
5: $\quad\quad I := I \cup \Delta$
6: $\quad\quad \Delta := \Pi^s_{\mathsf{r}}\big[I \,\vdots\, \Delta\big] \setminus I$

identifies in each round of rule application 'newly applicable' rule instances (i.e., instances that depend on a fact derived in the previous round) as shown in Algorithm 1. For each stratum, the algorithm initialises Δ, the set of newly derived facts, by combining the explicit facts in the current stratum ($E \cap \mathsf{O}^s$) with the facts derivable from previous strata via nonrecursive rules ($\Pi^s_{\mathsf{nr}}\big[I\big]$). Then, in lines 4–6 it iteratively computes all consequences of Δ. To this end, in line 6 it uses operator $\Pi\big[I \,\vdots\, \Delta\big]$, which extends $\Pi\big[I\big]$ to allow identifying 'newly applicable' rule instances. Specifically, given datasets I and $\Delta \subseteq I$, we define $\mathsf{inst}_r\big[I \,\vdots\, \Delta\big]$ and $\Pi\big[I \,\vdots\, \Delta\big]$ as follows.

$$\mathsf{inst}_r\big[I \,\vdots\, \Delta\big] = \{r\sigma \mid \mathsf{b}^+(r\sigma) \subseteq I,\ \mathsf{b}^-(r\sigma) \cap I = \emptyset,\ \text{and } \mathsf{b}^+(r\sigma) \cap \Delta \neq \emptyset\} \tag{3.1}$$

$$\Pi\big[I \,\vdots\, \Delta\big] = \bigcup_{r \in \Pi} \{\mathsf{h}(r') \mid r' \in \mathsf{inst}_r\big[I \,\vdots\, \Delta\big]\} \tag{3.2}$$

$\Pi\big[I \,\vdots\, \Delta\big]$ returns a dataset containing the head of each rule instance r' that is applicable to I and is affected by Δ (i.e., a positive body atom of r' occurs in the set of facts Δ derived in the previous round of rule application). $\Pi\big[I \,\vdots\, \Delta\big]$ can be evaluated as follows. For each fact F in Δ, we identify positive body atoms in Π that could be matched to F. For each of these matches, we use the matched body atom as the *pivot*, evaluate the partially instantiated rule body as a query, and apply the query answers to the head of the rule to obtain the derived facts. It is not hard to see that the algorithm computes $I = \mathsf{mat}(\Pi, E)$. Moreover, since Δ is different between rounds of rule application, the algorithm conisders each applicable rule instance only once.

We now analyse the running time of Algorithm 1 on input Π, λ, and E. To facilitate our discussion, let $n = |E|$ be the number of facts in E, and let $m = |\Pi|$ be the number of rules in Π; moreover, let k be the largest possible number of distinct variables in the

body of a rule in Π; finally, let u be the largest possible arity of a predicate that occurs in Π, and let w be the largest possible number of body atoms for a rule in Π. Consider the execution of Algorithm 1. Line 1 clearly requires only $O(1)$ time. We next look into the cost of handling one stratum with index s, $1 \leq s \leq S$. Line 3 involves evaluating each nonrecursive rule of stratum s over dataset I. Each of these rules has a maximum of k variables in the body, and the total number of constants available in I is in the order of $u \cdot n$ since each fact in E gives rise to a maximum number of u constants. Hence, the total number of possible substitutions to be checked for each rule is bounded by $u^k \cdot n^k$. Moreover, for each of these substitutions, it takes $O(w)$ time to check whether positive and negative body atoms of the rule satisfy the respective conditions against I, assuming that all facts are indexed. Consequently, the cost of evaluating $\Pi^s_{\mathsf{nr}}\Big[I\Big]$ is $O(|\Pi^s_{\mathsf{nr}}| \cdot w \cdot u^k \cdot n^k)$. Now the size of the output, i.e., the number of facts derived from the above evaluation, is bounded by $|\Pi^s_{\mathsf{nr}}| \cdot u^u \cdot n^u$, and so the set operations in line 3 require $O(n + |\Pi^s_{\mathsf{nr}}| \cdot u^u \cdot n^u) = O(|\Pi^s_{\mathsf{nr}}| \cdot u^u \cdot n^u)$ time. By definition we have $u \leq k$, and so the total running time for line 3 is simply $O(|\Pi^s_{\mathsf{nr}}| \cdot w \cdot u^k \cdot n^k)$.

Next, the algorithm enters the loop of lines 4–6. As in the case of line 3, the overall cost is dominated by the rule evaluation in line 6, and we can safely ignore the cost of set operations. We examine the cost of handling a rule $r \in \Pi^s_{\mathsf{r}}$ throughout the loop. The rule could have at most w body atoms. Consider the ith body atom B of the rule, and let l be the number of distinct variables in B. Since Δ is distinct among iterations, during the execution of the loop, B could be used as the pivot atom in a maximum number of $u^l \cdot n^l$ queries. For each of these queries, there are at most $k - l$ variables to be matched, leading to a total number of $u^{k-l} \cdot n^{k-l}$ substitutions to be checked. Checking each of these substitutions requires $O(w)$ time, so the cost of evaluating rule r in the loop of lines 4–6 using B as the pivot is $O(w \cdot u^l \cdot n^l \cdot u^{k-l} \cdot n^{k-l}) = O(w \cdot u^k \cdot n^k)$. The number of positive body atoms in r is at most w, and the number of recursive rules is $|\Pi^s_{\mathsf{r}}|$, so the overall cost for the loop of lines 4–6 is $O(|\Pi^s_{\mathsf{r}}| \cdot w^2 \cdot u^k \cdot n^k)$. Hence, the running time for stratum Π^s is $O(|\Pi^s_{\mathsf{nr}}| \cdot w \cdot u^k \cdot n^k + |\Pi^s_{\mathsf{r}}| \cdot w^2 \cdot u^k \cdot n^k)$, which could be simplified as $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$. As a result, the overall running time is $O(\Sigma_{1 \leq s \leq S} |\Pi^s| \cdot w^2 \cdot u^k \cdot n^k) = O(m \cdot w^2 \cdot u^k \cdot n^k)$.

Seminaïve evaluation is guaranteed to apply each applicable inference exactly once. However, for certain rules or rule combinations, all consequences can actually be computed without considering every applicable inference. For example, consider applying a program that axiomatises a relation R as symmetric and transitive to input facts that describe a connected graph of n vertices. In a later section we show that computing all consequences using seminaïve evaluation involves $O(n^3)$ rule applications, whereas a custom algorithm that we discuss in a later section can achieve the same goal using only $O(n^2)$ steps.

3.3 The Delete/Rederive Algorithm

The DRed algorithm is a well-known known solution to the datalog materialisation maintenance problem. The algorithm and its variants have been extensively used in practice [63, 70]. Let Π be a program and let λ be a stratification of Π; moreover, let E be a set of explicit facts; finally, let I denote the materialisation of Π with regards to E. Now, assume that we would like to update the explicit facts E by deleting E^- and adding E^+. Then, the DRed algorithm efficiently modifies the 'old' materialisation $I = \mathsf{mat}(\Pi, E)$ to the 'new' materialisation $I' = \mathsf{mat}(\Pi, (E \setminus E^-) \cup E^+)$ by deleting some facts and adding others. We call such facts *affected* by the update.

Due to the update, some rule instances that are applicable to I will no longer be applicable to I', and some rule instances that are not applicable to I will be applicable to I'; we also call such rule instances *affected* by the update. A key problem in materialisation maintenance is to identify the affected rule instances. Clearly, the body of each affected rule instance must contain an affected fact. The operators $\mathsf{inst}_r\Big[I \,\vdots\, \Delta\Big]$ and $\Pi\Big[I \,\vdots\, \Delta\Big]$ defined in Section 3.2 can only capture rule instances and consequences affected by changes to the matching of positive body atoms. To identify rule instances and consequences affected by changes to the matching of negative body atoms, we further generalise these operators. In particular, let I^{p}, I^{n}, P, and N be datasets such that $P \subseteq I^{\mathsf{p}}$ and $N \cap I^{\mathsf{n}} = \emptyset$; then, let

$$\begin{aligned}\mathsf{inst}_r\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \,\vdots\, P, N\Big] = \\ \{r\sigma \mid \mathsf{b}^+(r)\sigma \subseteq I^{\mathsf{p}} \text{ and } \mathsf{b}^-(r)\sigma \cap I^{\mathsf{n}} = \emptyset, \text{ and} \\ \mathsf{b}^+(r)\sigma \cap P \neq \emptyset \text{ or } \mathsf{b}^-(r)\sigma \cap N \neq \emptyset\}\end{aligned} \tag{3.3}$$

and let

$$\Pi\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\Big] = \bigcup_{r \in \Pi} \{\mathsf{h}(r') \mid r' \in \mathsf{inst}_r\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\Big]\}. \tag{3.4}$$

It should be noted that in both $\mathsf{inst}_r\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\Big]$ and $\Pi\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\Big]$, operator '$\colon$' has lower priority than the comma operator. The application of the comma results in two pairs of datasets, which are then distinguished by the '$\colon$' operator. The first pair consists of I^{p} and I^{n}, the datasets in which the positive and the negative rule atoms are evaluated, and the second pair of datasets, P and N, refer to the affected positive and negative facts. Intuitively, $\mathsf{inst}_r\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\Big]$ are the affected instances of r; and $\Pi\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\Big]$ are the affected consequences of Π. We define $\mathsf{inst}_r\Big[I^{\mathsf{p}}, I^{\mathsf{n}}\Big]$ and $\Pi\Big[I^{\mathsf{p}}, I^{\mathsf{n}}\Big]$ analogously to above, but without the condition '$\mathsf{b}^+(r)\sigma \cap P \neq \emptyset$ or $\mathsf{b}^-(r)\sigma \cap N \neq \emptyset$'. We omit for readability I^{n} whenever $I^{\mathsf{p}} = I^{\mathsf{n}}$, and furthermore we omit N when $N = \emptyset$. Sets $\Pi\Big[I^{\mathsf{p}}, I^{\mathsf{n}}\Big]$ and $\Pi\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\Big]$ can be computed efficiently in practice by evaluating the body of each rule $r \in \Pi$ as a conjunctive query and instantiating the head as needed.

Algorithm 2 formalises DRed. The algorithm processes each stratum s and accumulates the necessary changes to I in the set D of *overdeleted* and the set A of *added* facts. The materialisation is updated in line 12. Therefore, prior to that, I and $(I \setminus D) \cup A$ are the 'old' and the 'new' materialisation, respectively. The computation proceeds in three phases.

In the *overdeletion* phase, D is extended with all facts that potentially need to be deleted due to the update. In line 14 the algorithm identifies the facts that are explicitly deleted ($E^- \cap \mathsf{O}^s$) or are affected by deletions and insertions in the previous strata ($\Pi^s\Big[I \colon D \setminus A, A \setminus D\Big]$), and then in lines 15–19 it computes their consequences. It uses a form of the seminaïve strategy, which ensures that each rule instance is considered only once. Please note that a fact derived in the overdeletion phase may still hold after the update due to the presence of alternative derivations, so D is essentially an overestimation of the facts that no longer hold after the update.

In the *one-step rederivation* phase, R is computed as the set of facts that have been overdeleted, but that hold nonetheless. To this end, in line 10 the algorithm considers each

Algorithm 2 DRed$(\Pi, \lambda, E, I, E^-, E^+)$

7: $D := A := \emptyset, \quad E^- = (E^- \cap E) \setminus E^+, \quad E^+ = E^+ \setminus E$
8: **for** each stratum index s with $1 \leq s \leq S$ **do**
9: OVERDELETE
10: $R := \{F \in D \cap \mathsf{O}^s \mid F \in E \setminus E^-$ or there exist $r \in \Pi^s$ and $r' \in \mathsf{inst}_r\Big[I \setminus (D \setminus A), I \cup A\Big]$ with $F = \mathsf{h}(r')\}$
11: INSERT
12: $E := (E \setminus E^-) \cup E^+, \quad I := (I \setminus D) \cup A$

13: **procedure** OVERDELETE
14: $N_D := (E^- \cap \mathsf{O}^s) \cup \Pi^s\Big[I \,\vdots\, D \setminus A, A \setminus D\Big]$
15: **loop**
16: $\Delta_D := N_D \setminus D$
17: **if** $\Delta_D = \emptyset$ **then break**
18: $N_D := \Pi^s_{\mathsf{r}}\Big[I \setminus (D \setminus A), I \cup A \,\vdots\, \Delta_D\Big]$
19: $D := D \cup \Delta_D$

20: **procedure** INSERT
21: $N_A := R \cup (E^+ \cap \mathsf{O}^s) \cup \Pi^s\Big[(I \setminus D) \cup A \,\vdots\, A \setminus D, D \setminus A\Big]$
22: **loop**
23: $\Delta_A := N_A \setminus ((I \setminus D) \cup A)$
24: **if** $\Delta_A = \emptyset$ **then break**
25: $A := A \cup \Delta_A$
26: $N_A := \Pi^s_{\mathsf{r}}\Big[(I \setminus D) \cup A \,\vdots\, \Delta_A\Big]$

fact F in $D \cap \mathsf{O}^s$, and it adds F to R if F is explicit or it is rederived by a rule instance. The latter involves evaluating rules 'backwards': the algorithm identifies each rule $r \in \Pi^s$ whose head can be matched to F, and it evaluates over the 'new' materialisation the body of r as a query with the head variables bound; fact F holds if the query returns at least one answer. As we discuss shortly, this step can be a major source of inefficiency in practice, and a main feature of our DRed^c algorithm, which we discuss in detail in a later section, is that DRed^c completely eliminates 'backward' rule evaluation and thus significantly improves the performance.

In the *insertion* step, in line 21 the algorithm combines the one-step rederived facts (R) with the explicitly added facts ($E^+ \cup \mathsf{O}^s$) and the facts added due to the changes in previous strata ($\Pi^s\Big[(I \setminus D) \cup A \,\vdots\, A \setminus D, D \setminus A\Big]$), and then in lines 22–26 it computes all of their consequences and adds them to A. Again, the seminaïve strategy ensures that

each rule instance is considered only once during insertion.

We now analyse the running time of Algorithm 2 on input Π, λ, E, I, E^-, and E^+. For simplicity, let $n = |E| + |E^-| + |E^+|$ be the total number of facts in E, E^-, and E^+, and let $m = |\Pi|$ be the number of rules in Π. Note that we measure the performance in the size of the explicit facts instead of I so that we can directly compare the running time of Algorithm 2 with that of Algorithm 1. We define k, u, and w in the same way as in Section 3.2. In particular, let k be the largest possible number of distinct variables in the body of one rule in Π; let u be the largest possible arity of a predicate that occurs in Π; and let w be the largest possible rule body size. As in Section 3.2, we assume that all facts are indexed, and that it takes $O(1)$ time to check if a fact is in a particular set or to add a fact to a set. Consider the execution of Algorithm 2. It is straightforward to see that line 7 requires only $O(n)$ time. We next look into the cost of handling one stratum Π^s with $1 \leq s \leq S$. More specifically, we will examine the running time for each of the three phases, i.e., overdeletion, one-step rederivation, and insertion.

Overdeletion starts with line 14, which identifies facts affected by changes in the previous stratum by evaluating $\Pi^s\big[I \,\vdots\, D \setminus A, A \setminus D\big]$. We analyse the cost of handling one rule $r \in \Pi^s$ in line 14. The rule could have at most w positive and negative body atoms. Consider the ith body atom B of the rule, and let l be the number of distinct variables in B. Then, whether B is a positive or a negative body atom, during the execution of line 14, B could be used as the pivot atom in a maximum number of $u^l \cdot n^l$ queries; for each of these queries, there are at most $k - l$ variables left to be matched; hence, the running time for using B as the pivot atom is $O(w \cdot u^k \cdot n^k)$. There is a maximum number of w body atoms in r, and the number of rules in the current stratum is $|\Pi^s|$, so the overall cost for line 14 is $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$. The set operations in line 14 require only $O(n + |\Pi^s| \cdot u^u \cdot n^u)$ time, which is negligible compared to the cost of rule evaluation. The algorithm then enters the loop of lines 15–19. We focus on the cost of rule evaluation first, i.e., the cost of line 18, and we start with examining the running time of handling one rule $r \in \Pi^s_{\mathsf{r}}$ throughout the loop. The rule could have at most w positive body atoms. Consider the ith body atom B of the rule, and let l be the number of distinct variables in B. Since Δ_D is distinct among iterations, atom B could be used as the pivot atom in at

most $u^l \cdot n^l$ queries. Each of these queries require $O(w \cdot u^{k-l} \cdot n^{k-l})$ time to execute, so the cost of evaluating rule r in the loop of lines 15–19 using B as the pivot is $O(w \cdot u^k \cdot n^k)$. The number of positive body atoms in r is at most w, and the number of recursive rules is $|\Pi^s_\mathsf{r}|$. Therefore, the overall cost for the rule evaluation in line 18 throughout the loop is $O(|\Pi^s_\mathsf{r}| \cdot w^2 \cdot u^k \cdot n^k)$. Finally, in each iteration of lines 15–19, the cost incurred due to set operations is linear in the size of N_D, which is in turn bounded by the maximum number of substitutions checked in line 18. Hence, the cost of overdeletion for stratum Π^s is simply $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$.

We next examine the one-step rederivation step that takes place in line 10. For each rule $r \in \Pi^s$, let l be the number of distinct variables in its head. Then, the head of r is matched to a fact in D for at most $u^l \cdot n^l$ times. For each of these matches, the body of the rule will be partially instantiated and then evaluated as a query, with at most $k - l$ variables to be mapped; hence, this query requires $O(w \cdot u^{k-l} \cdot n^{k-l})$ time to compute. The overall running time for rule r is thus $O(w \cdot u^k \cdot n^k)$. The total number of rules is $|\Pi^s|$, so the overall cost of rule evaluation is $O(|\Pi^s| \cdot w \cdot u^k \cdot n^k)$. The cost of iterating through $D \cap \mathsf{O}^s$ and checking if each fact belongs to $E \setminus E^-$ require $O(|\Pi^s| \cdot u^u \cdot n^u)$ time. Since $u \leq k$, the running time for one-step rederivation is $O(|\Pi^s| \cdot w \cdot u^k \cdot n^k)$.

As for the insertion stage, the behaviour of the algorithm in terms of running time is similar to that of the overdeletion stage. In particular, line 21 is analogous to line 14 and requires $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$ time. The loop of lines 22–26 is analogous to the loop of lines 15–19, and thus requires $O(|\Pi^s_\mathsf{r}| \cdot w^2 \cdot u^k \cdot n^k)$ time. Hence, the overall complexity for insertion is $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$.

The overall cost of handling one stratum Π^s is $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$. As a result, the loop of lines 8–11 requires $O(\Sigma_{1 \leq s \leq S} |\Pi^s| \cdot w^2 \cdot u^k \cdot n^k) = O(m \cdot w^2 \cdot u^k \cdot n^k)$. Finally, line 12 requires $O(m \cdot u^u \cdot n^u)$ time, which can be safely discarded. The algorithm thus runs in $O(m \cdot w^2 \cdot u^k \cdot n^k)$ time. This is the same as the complexity of Algorithm 1, which matches our expectation: if we take deletion as an example, in the worst case one will have to delete everything, which means redoing all the work that has been carried out during the initial materialisation.

An interesting observation is that the worst-case complexity of one-step rederivation is lower than that of overdeletion or insertion by a factor of w. This is so because during overdeletion and insertion a rule could be evaluated with any body atom as the pivot, whereas during one-step rederivation the evaluation of a rule could only start from its head. In reality, however, one-step rederivation could still be the dominating source of inefficiency in many cases. In a later section we present two examples for which the time required for 'backward' rule evaluation in one-step rederivation dominates the total running time.

In both overdeletion and insertion, DRed uses variants of the seminaïve evaluation strategy, which considers each applicable rule instance exactly once. In the context of incremental maintenance, this means that DRed has to consider each affected rule instance. This can be problematic even for small updates. For example, consider again applying a program that axiomatises a relation R as symmetric and transitive to input facts that describe a connected graph of n vertices. Then, deleting one fact from the input facts using DRed involves considering $O(n^3)$ rule instances in overdeletion only, whereas a custom solution that we discuss in a later section can achieve the same goal in $O(n^2)$ steps.

3.4 The Backward/Forward Algorithm

In contrast to DRed which often generates an overestimation of the deleted facts, the B/F algorithm searches for alternative derivations of a fact immediately after it is derived during deletion by using a combination of backward and forward chaining. This makes deletion precise and avoids the potential inefficiency of overdeletion. In practice, B/F often, but not always, outperforms DRed [51]. Algorithm 3 formalises B/F.

Procedure DeleteUnproved plays an analogous role to the overdeletion step of DRed. The main difference is that a fact F is deleted (i.e., added to Δ_D) in line 39 only if no alternative derivation can be found using a combination of backward and forward chaining implemented in functions Check and Saturate. If an alternative derivation is found, F is added to the set P of *proved* facts.

Algorithm 3 B/F$(\Pi, \lambda, E, I, E^-, E^+)$

27: $D := A := \emptyset, \quad E^- = (E^- \cap E) \setminus E^+, \quad E^+ = E^+ \setminus E$
28: **for** each stratum index s with $1 \leq s \leq S$ **do**
29: $\quad C := P := Y := \emptyset$
30: $\quad$ DeleteUnproved
31: $\quad$ Insert
32: $E := (E \setminus E^-) \cup E^+, \quad I := (I \setminus D) \cup A$

33: **procedure** DeleteUnproved
34: $\quad N_D := (E^- \cap \mathsf{O}^s) \cup \Pi^s\big[I : D \setminus A, A \setminus D\big]$
35: $\quad$ **loop**
36: $\quad\quad \Delta_D := \emptyset$
37: $\quad\quad$ **for** $F \in N_D \setminus D$ **do**
38: $\quad\quad\quad$ Check(F)
39: $\quad\quad\quad$ **if** $F \notin P$ **then** $\Delta_D := \Delta_D \cup \{F\}$
40: $\quad\quad$ **if** $\Delta_D = \emptyset$ **then break**
41: $\quad\quad N_D := \Pi^s_{\mathsf{r}}\big[I \setminus (D \setminus A), I \cup A : \Delta_D\big]$
42: $\quad\quad D := D \cup \Delta_D$

43: **function** Check(F)
44: $\quad$ **if** $F \notin C$ **then**
45: $\quad\quad$ **if** Saturate$(F) = \mathsf{f}$ **then**
46: $\quad\quad\quad$ **for** each $r \in \Pi^s_{\mathsf{r}}$ and each $r' \in \mathsf{inst}_r\big[I \setminus ((D \cup \Delta_D) \setminus A), I \cup A\big]$ s.t. $\mathsf{h}(r') = F$ **do**
47: $\quad\quad\quad\quad$ **for** $G \in \mathsf{b}^+(r') \cap \mathsf{O}^s$ **do**
48: $\quad\quad\quad\quad\quad$ Check(G)
49: $\quad\quad\quad\quad\quad$ **if** $F \in P$ **then return**

50: **function** Saturate(F)
51: $\quad C := C \cup \{F\}$
52: $\quad$ **if** $F \in (E \setminus E^-) \cup Y$ or $\exists r \in \Pi^s_{\mathsf{nr}}$ and $r' \in \mathsf{inst}_r\big[I \setminus (D \setminus A), I \cup A\big]$ s.t. $\mathsf{h}(r') = F$ **then**
53: $\quad\quad N_P := \{F\}$
54: $\quad\quad$ **loop**
55: $\quad\quad\quad \Delta_P := (N_P \cap C) \setminus P, \quad Y := Y \cup N_P \setminus C$
56: $\quad\quad\quad$ **if** $\Delta_P = \emptyset$ **then return** t
57: $\quad\quad\quad P := P \cup \Delta_P$
58: $\quad\quad\quad N_P := \Pi^s_{\mathsf{r}}\big[P \cup (\mathsf{O}^{<s} \cap (I \setminus (D \setminus A))), I \cup A : \Delta_P\big]$
59: $\quad$ **else return** f

60: **procedure** Insert
61: $\quad N_A := (E^+ \cap \mathsf{O}^s) \cup \Pi^s\big[(I \setminus D) \cup A : A \setminus D, D \setminus A\big]$
62: $\quad$ **loop**
63: $\quad\quad \Delta_A := N_A \setminus ((I \setminus D) \cup A)$
64: $\quad\quad$ **if** $\Delta_A = \emptyset$ **then break**
65: $\quad\quad A := A \cup \Delta_A$
66: $\quad\quad N_A := N_A \cup \Pi^s_{\mathsf{r}}\big[(I \setminus D) \cup A : \Delta_A\big]$

A call to Check(F) searches for the alternative derivations of F using backward chaining. The function maintains the set C of *checked* facts, which ensures that each F is checked only once (line 44 and 51). The procedure first calls Saturate(F) to determine

whether F follows from the facts considered thus far; we discuss this step in more detail shortly. If F is not proved, the procedure then examines in lines 46–49 each instance r' of a recursive rule that derives F in the 'old' materialisation, and it tries to prove all body atoms of r' from the current stratum. The function terminates once F is successfully proved (line 49).

Set P accumulates facts that are checked and successfully proved, and it is computed in function SATURATE using forward chaining. Given a fact F that is being checked, it first verifies whether F has a nonrecursive derivation. This is done by evaluating the nonrecursive rules 'backwards' in the same way as in line 10 of DRed. If F does have a nonrecursive derivation, then it is added to P via lines 53 and 55. The procedure then propagates the consequences of F (line 53–58). In particular, the procedure ensures that each consequence F' of P and the facts in the 'new' materialisation in the previous strata with regards to the recursive rules is added to P if $F' \in C$, or is added to the set Y of *delayed* facts if $F' \notin C$. Intuitively, set Y contains facts that are proved but that have not been checked yet. If a fact in Y is checked at a later point, it is proved in line 52 without having to apply the rules again.

Since the deletion step of B/F is 'exact' in the sense that it deletes precisely those facts that no longer hold after the update, rederivation is not needed. Thus, DELETEUNPROVED is directly followed by INSERT, which is the same as in DRed.

The B/F algorithm evaluates recursive and nonrecursive rules 'backwards' in lines 46 and 52, respectively. This may constitute a major source of inefficiency due to similar reasons as explained in Section 3.3 for the DRed algorithm. Moreover, the deletion, insertion, and saturation procedures all use variants of the seminaïve evaluation strategy, which has to consider each applicable rule instance. Therefore, the B/F algorithm is likely to suffer from deficiencies outlined in Sections 3.2 and 3.3 as well.

3.5 The Counting Algorithm

The Counting algorithm is another well-known solution to the datalog materialisation maintenance problem. For each fact in the materialisation, the algorithm keeps track

of the number of applicable rule instances that derive this fact: the counter of a fact is incremented when a new applicable rule instance for the fact is found; and it is decremented when a rule instance is no longer applicable. A fact can thus be deleted when its counter drops to zero, without the potentially costly 'backward' rule evaluation.

Counting can be made optimal (for nonrecursive rules) in the sense that it considers precisely the rule instances that no longer hold after the update and the rule instances that only hold after the update. Please note that the DRed and B/F algorithms are not optimal in this sense: the 'backward' rule evaluation in line 10 of the DRed algorithm and line 125 of the B/F algorithm try to identify alternative derivations for the deleted facts and thus may consider rule instances that hold both before and after the update.

A major drawback of the Counting algorithm is that, unlike DRed and B/F which are capable of handling arbitrary datalog programds, the Counting algorithm can be incorrect when the program contains recursion. Intuitively, when there is recursion, the counters may not be reliable during deletion: the counter of a fact being nonzero does not necessarily imply that the fact still holds after the update since it may affect itself via some recursive rules. Recursion is a key feature of datalog, allowing one to express common properties such as transitivity. Thus, despite its favourable properties, the Counting algorithm does not provide us with a general solution to the materialisation maintenance problem.

4

Optimised Incremental Maintenance

In this chapter we present two novel algorithms, DRed^c and B/F^c, for the incremental maintenance of datalog materialisations. We begin in Section 4.1 with two examples that illustrate the drawbacks of existing approaches and thus motivate our work. In Sections 4.2 and 4.3 we describe our two new algorithms in full detail. Finally, in Section 4.4 we empirically compare our solutions to existing approaches.

4.1 Motivation and Intuition

As already mentioned in Section 3.3 and Section 3.4, both DRed and B/F algorithms involve steps that evaluate rules 'backwards'. To our work, we next discuss how evaluating rules 'backwards' can be a significant source of inefficiency during materialisation maintenance. We base our discussion on the DRed algorithm for simplicity, but our conclusions apply to the B/F algorithm as well.

The one-step rederivation in line 10 of Algorithm 2 evaluates rules 'backwards'. In this section we present two examples that demonstrate how this can be a major source of inefficiency. It should be noted that Example 1 is derived from datasets we used in our empirical evaluation that we present in Section 4.4; hence, these problems actually arise in practice.

Our discussion depends on several details. In particular, we assume that all facts are indexed so that all facts matching any given atom (possibly containing constants) can be identified efficiently. Moreover, we assume that conjunctive queries corresponding to rule bodies are evaluated left-to-right: for each match of the first conjunct, we partially instantiate the rest of the body and match it recursively. Finally, we assume that query atoms are reordered prior to evaluation to obtain an efficient evaluation plan.

Example 1. *Let* Π *and* E *be the program and the dataset as specified in* (4.1) *and* (4.2)*, respectively.*

$$R(x, y_1) \wedge R(x, y_2) \rightarrow S(y_1, y_2) \tag{4.1}$$

$$E = \{R(a_i, b), R(a_i, c_i) \mid 1 \leq i \leq n\} \tag{4.2}$$

The materialisation $\mathsf{mat}(\Pi, E)$ *consists of* E *extended with facts* $S(b, b)$*,* $S(b, c_i)$*,* $S(c_i, b)$*, and* $S(c_i, c_i)$ *for* $1 \leq i \leq n$*.*

During materialisation, the body of rule (4.1) *can be evaluated efficiently left-to-right: we match* $R(x, y_1)$ *to either* $R(a_i, b)$ *or* $R(a_i, c_i)$*; this instantiates* $R(x, y_2)$ *as* $R(a_i, y_2)$*, and we use the index to find the matching facts* $R(a_i, b)$ *and* $R(a_i, c_i)$*. Thus,* $R(x, y_1)$ *has* $2n$ *matches, each of which contributes to two matches of* $R(x, y_2)$*, so the overall cost of rule matching is* $O(n)$*. The rule body is symmetric, so reordering the body atoms has no effect.*

Now assume that we delete all $R(a_i, c_i)$ *with* $1 \leq i \leq n$*. DRed then overdeletes all* $S(b, c_i)$*,* $S(c_i, b)$ *and* $S(c_i, c_i)$ *facts in lines 14–19, and this can be done efficiently as in the previous paragraph. Next, in one-step rederivation, the algorithm will match these facts to the head of the rule* (4.1) *and obtain queries* $R(x, b) \wedge R(x, c_i)$*,* $R(x, c_i) \wedge R(x, b)$*, and* $R(x, c_i) \wedge R(x, c_i)$*. All but the last of these queries contain atom* $R(x, b)$ *and, no matter how we reorder the body atoms of* (4.1)*, we have* n *queries where* $R(x, b)$ *is evaluated first. Each of these* n *queries identifies* n *candidate matches* $R(a_i, b)$ *using the index only to find out that the second atom cannot be matched. Thus,* $R(x, b)$ *is matched to* n^2 *facts in total, so the cost of one-step rederivation is* $O(n^2)$*—one degree higher than for materialisation.*

Example 1 shows that evaluating a rule 'backwards' can be inherently more difficult than evaluating it during materialisation, thus giving rise to a dominating source of inefficiency. Below is another example where evaluating rules 'backwards' can be problematic.

Example 2. *Let program* Π *consist of rules* (4.3) *and* (4.4)*. Note that* a *is a constant in rule* (4.3)*.*

$$B(a, y, z) \rightarrow D(y, z) \tag{4.3}$$

$$D(x, z_1) \wedge B(x, y, z_2) \wedge C(z_1, z_2, z) \rightarrow D(y, z) \tag{4.4}$$

Let E *be the dataset as specified below.*

$$E = \{B(a, b_1, e_1), B(a, c_i, e_i), B(b_i, d_j, e_i), C(e_1, e_i, e_1) \mid 1 \leq i, j \leq n\}$$

During materialisation, rule (4.3) *first derives* $D(b_1, e_1)$ *and all* $D(c_i, e_i)$ *with* $1 \leq i \leq n$*, so the cost of this step is* $O(n)$*. Next, atom* $D(x, z_1)$ *in rule* (4.4) *is matched to* n *facts* $D(c_i, e_i)$ *without deriving anything. Atom* $D(x, z_1)$ *is also matched to* $D(b_1, e_1)$ *once, so atom* $B(x, y, z_2)$ *is instantiated to* $B(b_1, y, z_2)$ *and matched to* n *facts* $B(b_1, d_j, e_1)$*. Each of these matches instantiates* $C(z_1, z_2, z)$ *as* $C(e_1, e_1, z)$*, which is then matched to* $C(e_1, e_1, e_1)$*. As a result,* n *facts* $D(d_j, e_1)$*,* $1 \leq j \leq n$*, will be derived, and the cost of rule matching is* $O(n)$*.*

Now assume that $B(a, b_1, e_1)$ *is deleted. Then,* $D(b_1, e_1)$ *and all* $D(d_j, e_1)$ *can be efficiently overdeleted as in the previous paragraph, but trying to prove them is much more difficult. Matching each* $D(d_j, e_1)$ *to the head of* (4.3) *produces a query* $B(a, d_j, e_1)$*, which does not produce a rule instance. Moreover, matching* $D(d_j, e_1)$ *to the head of* (4.4) *produces a query* $D(x, z_1) \wedge B(x, d_j, z_2) \wedge C(z_1, z_2, e_1)$*. If we evaluate* $B(x, d_j, z_2)$ *first, then we try* n *facts* $B(b_i, d_j, e_i)$ *with* $1 \leq i \leq n$*; for each of them, atom* $D(x, z_1)$ *is instantiated as* $D(b_i, z_1)$ *and is not matched in the surviving facts. If we evaluate* $D(x, z_1)$ *first, then we try* n *facts* $D(c_i, e_i)$*; for each of them, atom* $B(x, d_j, z_2)$ *is instantiated as* $B(c_i, d_j, z_2)$ *and is not matched. If we evaluate* $C(z_1, z_2, e_1)$ *first, then we try* n *facts* $C(e_1, e_i, e_1)$ *with* $1 \leq i \leq n$*; for each of them, atom* $B(x, d_j, z_2)$ *is instantiated as*

$B(x, d_j, e_i)$ and is matched to $B(b_i, d_j, e_i)$; but then, $D(x, z_1)$ is instantiated as $D(b_i, e_1)$ and there is no match. Thus, regardless of how we reorder the body of (4.4)*, the first atom considers a total of n^2 facts, so the cost of one-step rederivation is $O(n^2)$.*

4.2 Combining DRed with Counting

We now address the inefficiencies we outlined in Section 4.1. Towards this goal, in Section 4.2.1 we first present the intuitions, and then in Section 4.2.2 we formalise our solution.

4.2.1 Intuition

As we already mentioned in Section 3.5, the Counting algorithm [35] does not evaluate rules 'backwards'; instead, it tracks the number of derivations of each fact. The main drawback of Counting is that it cannot handle recursive rules. We now illustrate the intuition behind our DRed^c algorithm, which combines DRed with Counting in a way that eliminates 'backward' rule evaluation, while still supporting recursive rules.

The DRed^c algorithm associates with each fact two counters that track the derivations via the nonrecursive and the recursive rules separately. The counters are decremented (resp. incremented) when the associated fact is derived in overdeletion (resp. insertion), which allows for two important optimisations. First, as in the Counting algorithm, the nonrecursive counter always reflects the number of derivations from facts in earlier strata; hence, a fact with a nonzero nonrecursive counter should never be overdeleted because it clearly remains true after the update. This optimisation captures the behaviour of Counting on nonrecursive rules and it also helps limit overdeletion. Second, if we never overdelete facts with nonzero nonrecursive counters, the only way for a fact to still hold after overdeletion is if its recursive counter is nonzero; hence, we can replace 'backward' rule evaluation by a simple check of the recursive counter. Note, however, that the recursive counters can be checked only after overdeletion finishes. This optimisation extends the idea of Counting to recursive rules to completely avoid 'backward' rule evaluation. The following example illustrates these ideas and compares them to DRed.

Example 3. *Let* Π *be the program containing rule* (4.5).

$$A(x) \wedge B(x, y) \rightarrow A(y) \tag{4.5}$$

Moreover, let E *be defined as follows:*

$$E = \{A(a), A(b), A(d), B(a, c), B(b, c), B(c, d), B(d, e)\}$$

The materialisation $\mathsf{mat}(\Pi, E)$ *extends* E *with* $A(c)$ *and* $A(e)$. *Figure 4.1 shows the dependencies between derivations using arrows. For clarity, we do not show the* B*-facts.*

Now assume that $A(a)$ *is deleted. The standard DRed algorithm first overdeletes* $A(a)$, $A(c)$, $A(d)$, *and* $A(e)$; *it rederives* $A(d)$ *since the fact is in* $E \setminus E^-$; *it rederives* $A(c)$ *by evaluating rule* (4.5) *'backwards'; and it derives* $A(d)$ *and* $A(e)$ *from the rederived facts.*

Now consider applying the DRed^c *to the same update. For each fact, Figure 4.1 shows a pair consisting of the nonrecursive and the recursive counter before the update (row T1), after overdeletion (row T2), and after the update (row T3). Note that the presence of a fact in* E *is akin to a nonrecursive derivation, so facts* $A(a)$, $A(b)$, *and* $A(d)$ *have nonrecursive derivation counts of one before the update. Now* $A(c)$ *is derived from* $A(a)$ *and* $A(b)$ *using the recursive rule* (4.5), *so the recursive counter for* $A(c)$ *is two. Analogously,* $A(d)$ *and* $A(e)$ *have just one recursive derivation each. During overdeletion,* $A(a)$ *is first removed from* E, *so the nonrecursive counter of* $A(a)$ *is decremented to zero and the fact is deleted. Since* $A(a)$ *derives* $A(c)$ *via rule* (4.5), *the recursive counter of* $A(c)$ *is decremented; since the nonrecursive counter of* $A(c)$ *is zero, the fact is overdeleted. Since* $A(c)$ *derives* $A(d)$ *via rule* (4.5), *the recursive counter of* $A(d)$ *is decremented. Now the nonrecursive counter of* $A(d)$ *is nonzero, so we know that* $A(d)$ *holds after the update; hence, the fact is* not *overdeleted, and the overdeletion phase stops. Thus, while DRed overdeletes four facts, DRed*^{c} *overdeletes only* $A(a)$ *and* $A(c)$, *and does not 'touch'* $A(e)$.

Next, DRed^c *proceeds to one-step rederivation. The recursive counter of* $A(c)$ *is nonzero, which means that the fact has a recursive derivation (from* $A(b)$ *in this case) that is not affected. Thus, DRed*^{c} *rederives* $A(c)$ *without any 'backward' rule evaluation.*

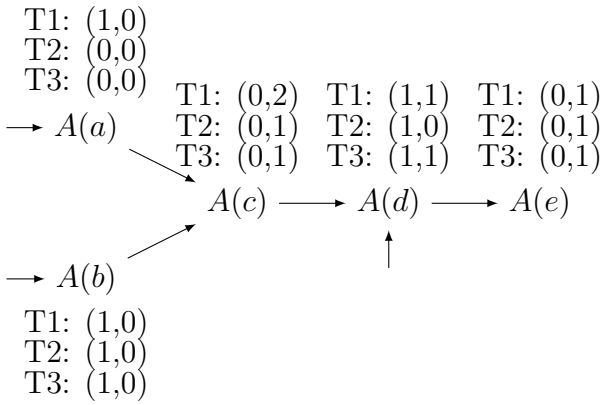

Figure 4.1: Derivations for Example 3

Finally, DRed^c applies insertion. Since $A(c)$ derives $A(d)$ via (4.5), the recursive counter of $A(d)$ is incremented. Fact $A(d)$, however, was not overdeleted, so insertion stops.

By avoiding 'backward' rule evaluation, DRed^c removes the dominating source of inefficiency in Examples 1 and 2. In fact, on the nonrecursive program from Example 1, the recursive counter is never used and DRed^c performs the same inferences as the Counting algorithm.

4.2.2 Formalisation

We now formalise our DRed^c algorithm. Our definitions use the standard notion of *multisets*—a generalisation of sets where each element is associated with a positive integer called the *multiplicity* specifying the number of the element's occurrences in the multiset. Moreover, $\oplus$ is the multiset union operator, which adds the elements' multiplicities. If an operand of $\oplus$ is a set, it is treated as a multiset where all elements have multiplicity one. Finally, we extend the notion of rule matching to correctly reflect the number of times a fact is derived: for I^{p}, I^{n}, P, and N datasets with $P \subseteq I^{\mathsf{p}}$ and $N \cap I^{\mathsf{n}} = \emptyset$, we define $\Pi\llbracket I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N \rrbracket$ as the multiset containing a distinct occurrence of $\mathsf{h}(r')$ for each rule $r \in \Pi$ and its instance $r' \in \mathsf{inst}_r\big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\big]$. This multiset can be computed analogously to $\Pi\big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon P, N\big]$.

Just like DRed, DRed^c takes as input a program Π, a stratification λ, a set of explicit facts E and its materialisation $I = \mathsf{mat}(\Pi, E)$, and the sets of facts E^- and E^+

to remove from and add to E. Additionally, the algorithm also takes as input maps C_{nr} and C_{r} that associate each fact F with its nonrecursive and recursive counters $C_{\mathsf{nr}}[F]$ and $C_{\mathsf{r}}[F]$, respectively. These maps should correctly reflect the relevant numbers of derivations. Formally, C_{nr} and C_{r} must be *compatible* with Π, λ, and E, which is the case if $C_{\mathsf{nr}}[F] = C_{\mathsf{r}}[F] = 0$ for each fact $F \notin I$, and, for each fact $F \in I$ and s the stratum index such that $F \in \mathsf{O}^s$ (i.e., s is the index of the stratum that F belongs to),

- $C_{\mathsf{nr}}[F]$ is the multiplicity of F in $E \oplus \Pi^s_{\mathsf{nr}}\llbracket I \rrbracket$, and
- $C_{\mathsf{r}}[F]$ is the multiplicity of F in $\Pi^s_{\mathsf{r}}\llbracket I \rrbracket$.

For simplicity, we assume that C_{nr} and C_{r} are defined on all facts, with $C_{\mathsf{nr}}[F] = C_{\mathsf{r}}[F] = 0$ for $F \notin I$; thus, we can simply increment the counters for each newly derived fact in procedure Insert. In practice, however, one can maintain counters only for the derived facts and initialise the counters to zero for the freshly derived facts.

DRed^c is formalised in Algorithm 4. Its structure is similar to DRed, with the following main differences: instead of evaluating rules 'backwards', one-step rederivation simply checks the recursive counters (line 70); a fact is overdeleted only if the nonrecursive derivation counter is zero (line 80); and the derivation counters are decremented in overdeletion (lines 75–78 and 82–83) and incremented in insertion (lines 87–90 and 95–96). The algorithm also accumulates changes to the materialisation in sets D and A by iteratively processing the strata of λ in three phases.

In the overdeletion phase, DRed^c first considers explicitly deleted facts or facts affected by the changes in earlier strata (lines 75–78). This is analogous to line 14 of DRed, but DRed^c must distinguish Π^s_{nr} from Π^s_{r} so it can decrement the appropriate counters. Next, DRed^c identifies the set Δ_D of facts that have not yet been deleted and whose nonrecursive counter is zero (line 80): a fact with a nonzero nonrecursive counter will always be part of the 'new' materialisation. Note that recursive derivations can be cyclic, so we cannot use the recursive counter to further constrain overdeletion at this point. Then, in lines 81–84 the algorithm propagates consequences of Δ_D just like Algorithm 2, with additionally decrementing the recursive counters in line 83.

Algorithm 4 DRed$^c(\Pi, \lambda, E, I, E^-, E^+, C_{\mathsf{nr}}, C_{\mathsf{r}})$

67: $D := A := \emptyset, \quad E^- = (E^- \cap E) \setminus E^+, \quad E^+ = E^+ \setminus E$
68: **for** each stratum index s with $1 \leq s \leq S$ **do**
69: OVERDELETE
70: $R := \{F \in D \cap \mathsf{O}^s \mid C_{\mathsf{r}}[F] > 0\}$
71: INSERT
72: $E := (E \setminus E^-) \cup E^+, \quad I := (I \setminus D) \cup A$

73: **procedure** OVERDELETE
74: $N_D := \emptyset$
75: **for** $F \in (E^- \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}\llbracket I \colon D \setminus A, A \setminus D \rrbracket$ **do**
76: $N_D := N_D \cup \{F\}, \quad C_{\mathsf{nr}}[F] := C_{\mathsf{nr}}[F] - 1$
77: **for** $F \in \Pi^s_{\mathsf{r}}\llbracket I \colon D \setminus A, A \setminus D \rrbracket$ **do**
78: $N_D := N_D \cup \{F\}, \quad C_{\mathsf{r}}[F] := C_{\mathsf{r}}[F] - 1$
79: **loop**
80: $\Delta_D := \{F \in N_D \setminus D \mid C_{\mathsf{nr}}[F] = 0\}$
81: **if** $\Delta_D = \emptyset$ **then break**
82: **for** $F \in \Pi^s_{\mathsf{r}}\llbracket I \setminus (D \setminus A), I \cup A \colon \Delta_D \rrbracket$ **do**
83: $N_D := N_D \cup \{F\}, \quad C_{\mathsf{r}}[F] := C_{\mathsf{r}}[F] - 1$
84: $D := D \cup \Delta_D$

85: **procedure** INSERT
86: $N_A := R$
87: **for** $F \in (E^+ \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}\llbracket (I \setminus D) \cup A \colon A \setminus D, D \setminus A \rrbracket$ **do**
88: $N_A := N_A \cup \{F\}, \quad C_{\mathsf{nr}}[F] := C_{\mathsf{nr}}[F] + 1$
89: **for** $F \in \Pi^s_{\mathsf{r}}\llbracket (I \setminus D) \cup A \colon A \setminus D, D \setminus A \rrbracket$ **do**
90: $N_A := N_A \cup \{F\}, \quad C_{\mathsf{r}}[F] := C_{\mathsf{r}}[F] + 1$
91: **loop**
92: $\Delta_A := N_A \setminus ((I \setminus D) \cup A)$
93: **if** $\Delta_A = \emptyset$ **then break**
94: $A := A \cup \Delta_A$
95: **for** $F \in \Pi^s_{\mathsf{r}}\llbracket (I \setminus D) \cup A \colon \Delta_A \rrbracket$ **do**
96: $N_A := N_A \cup \{F\}, \quad C_{\mathsf{r}}[F] := C_{\mathsf{r}}[F] + 1$

In the one-step rederivation phase, instead of evaluating rules 'backwards', DRed^c just checks the recursive counter of each fact $F \in D \cap \mathsf{O}^s$ (line 70): if $C_{\mathsf{r}}[F] \neq 0$, then some derivations of F were not 'touched' by overdeletion so F holds in the 'new' materialisation. Conversely, if $C_{\mathsf{r}}[F] = 0$, then $F \in D$ guarantees that $C_{\mathsf{nr}}[F] = 0$ holds as well, so F is not one-step rederivable by a rule in Π.

The insertion phase of DRed^c just uses the seminaïve evaluation while incrementing

the counters appropriately.

Without recursive rules, DRed^c becomes equivalent to Counting, and it is optimal in the sense that only affected rule instances are considered during the update. Moreover, the computational complexities of both DRed^c and DRed are the same as for the semi-naive materialisation algorithm: ExpTime in combined and PTime in data complexity [16]. Finally, DRed^c never performs more inferences than DRed and is thus more efficient. Theorem 4 states that our algorithm is correct, and its proof is given in Appendix A.1.

Theorem 4. *Algorithm 4 updates dataset I from $\mathsf{mat}(\Pi, E)$ to $\mathsf{mat}(\Pi, (E \setminus E^-) \cup E^+)$, and it updates C_{nr} and C_{r} so they are compatible with Π, λ, and $(E \setminus E^-) \cup E^+$.*

Theorem 5 states that Algorithm 4 has the same complexity as Algorithm 2. Please refer to Appendix A.2 for the details of the complexity analysis. Although we are not able to distinguish the two algorithms in terms of worst-case complexity. It is clear that the cost of one-step rederivation drops by a factor of $w \cdot u^{k-l} \cdot n^{k-l}$, from $O(|\Pi^s| \cdot w \cdot u^k \cdot n^k)$ to $O(|\Pi^s| \cdot u^u \cdot n^u)$. This shows that DRed^c has the potential of performing better than DRed, especially in cases where one-step rederivation dominates the overall cost.

Theorem 5. *Let n, m, k, u, and w be defined as in Section 3.3. Then, Algorithm 4 updates the materialisation I in $O(m \cdot w^2 \cdot u^k \cdot n^k)$ time.*

4.3 Combining B/F with Counting

The B/F algorithm by Motik et al. [51] uses a combination of backward and forward chaining that makes the deletion phase exact. More specifically, when a fact $F \in \Delta_D$ is considered during deletion, the algorithm uses a combination of backward and forward chaining to look for alternative derivations of F, and it deletes F only if no such derivation can be found. Backward chaining allows B/F to be much more efficient than DRed on many datasets, and this is particularly the case if a program contains many recursive rules. Thus, we cannot hope to remove all 'backward' rule evaluation without eliminating the algorithm's main advantage.

Algorithm 5 $\mathrm{B/F}^c(\Pi, \lambda, E, I, E^-, E^+, C_{\mathsf{nr}})$

97: $D := A := \emptyset, \quad E^- = (E^- \cap E) \setminus E^+, \quad E^+ = E^+ \setminus E$
98: **for** each stratum index s with $1 \leq s \leq S$ **do**
99: $\quad C := P := Y := \emptyset$
100: $\quad$ DeleteUnproved
101: $\quad$ Insert
102: $E := (E \setminus E^-) \cup E^+, \quad I := (I \setminus D) \cup A$

103: **procedure** DeleteUnproved
104: $\quad N_D := \emptyset$
105: $\quad$ **for** $F \in (E^- \cap \mathsf{O}^s) \cup \Pi^s_{\mathsf{nr}}[\![I : D \setminus A, A \setminus D]\!]$ **do**
106: $\quad\quad N_D := N_D \cup \{F\}, \quad C_{\mathsf{nr}}[F] := C_{\mathsf{nr}}[F] - 1$
107: $\quad N_D := N_D \cup \Pi^s_{\mathsf{r}}[I : D \setminus A, A \setminus D]$
108: $\quad$ **loop**
109: $\quad\quad \Delta_D := \emptyset$
110: $\quad\quad$ **for** $F \in N_D \setminus D$ **do**
111: $\quad\quad\quad$ Check(F)
112: $\quad\quad\quad$ **if** $F \notin P$ **then** $\Delta_D := \Delta_D \cup \{F\}$
113: $\quad\quad$ **if** $\Delta_D = \emptyset$ **then break**
114: $\quad\quad N_D := \Pi^s_{\mathsf{r}}[I \setminus (D \setminus A), I \cup A : \Delta_D]$
115: $\quad\quad D := D \cup \Delta_D$

116: **function** Check(F)
117: $\quad$ **if** $F \notin C$ **then**
118: $\quad\quad$ **if** Saturate(F) $= \mathsf{f}$ **then**
119: $\quad\quad\quad$ **for** each $r \in \Pi^s_{\mathsf{r}}$ and each
$r' \in \mathsf{inst}_r[I \setminus ((D \cup \Delta_D) \setminus A), I \cup A]$ s.t. $\mathsf{h}(r') = F$ **do**
120: $\quad\quad\quad\quad$ **for** $G \in \mathsf{b}^+(r') \cap \mathsf{O}^s$ **do**
121: $\quad\quad\quad\quad\quad$ Check(G)
122: $\quad\quad\quad\quad\quad$ **if** $F \in P$ **then return**

123: **function** Saturate(F)
124: $\quad C := C \cup \{F\}$
125: $\quad$ **if** $F \in Y$ or $C_{\mathsf{nr}}[F] > 0$ **then**
126: $\quad\quad N_P := \{F\}$
127: $\quad\quad$ **loop**
128: $\quad\quad\quad \Delta_P := (N_P \cap C) \setminus P, \quad Y := Y \cup N_P \setminus C$
129: $\quad\quad\quad$ **if** $\Delta_P = \emptyset$ **then return** t
130: $\quad\quad\quad P := P \cup \Delta_P$
131: $\quad\quad\quad N_P := \Pi^s_{\mathsf{r}}[P \cup (\mathsf{O}^{<s} \cap (I \setminus (D \setminus A))), I \cup A : \Delta_P]$
132: $\quad$ **else return** f

133: **procedure** Insert
134: $\quad N_A := \emptyset$
135: $\quad$ **for** $F \in (E^+ \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}[\![(I \setminus D) \cup A : A \setminus D, D \setminus A]\!]$ **do**
136: $\quad\quad N_A := N_A \cup \{F\}, \quad C_{\mathsf{nr}}[F] := C_{\mathsf{nr}}[F] + 1$
137: $\quad N_A := N_A \cup \Pi^s_{\mathsf{nr}}[(I \setminus D) \cup A : A \setminus D, D \setminus A]$
138: $\quad$ **loop**
139: $\quad\quad \Delta_A := N_A \setminus ((I \setminus D) \cup A)$
140: $\quad\quad$ **if** $\Delta_A = \emptyset$ **then break**
141: $\quad\quad A := A \cup \Delta_A$
142: $\quad\quad N_A := N_A \cup \Pi^s_{\mathsf{r}}[(I \setminus D) \cup A : \Delta_A]$

Still, there is room for improvement: backward chaining involves 'backward' evaluation of both nonrecursive and recursive rules, and we can use nonrecursive counters to eliminate the former. Algorithm 5 formalises B/F^c—our combination of the B/F algorithm by Motik et al. [51] with Counting. The main difference to the original B/F algorithm is that B/F^c associates with each fact a nonrecursive counter that is maintained in lines 105–106 and 135–136, and, instead of evaluating nonrecursive rules 'backwards' to explore alternative derivations of a fact, it just checks in line 125 whether the nonrecursive counter is nonzero. We know that a fact holds if its nonrecursive counter is nonzero; otherwise, we apply backward chaining to *recursive rules only*. We next describe the algorithm's steps in more detail.

Procedure DeleteUnproved plays an analogous role to the overdeletion step of DRed and DRed^c. The procedure maintains the nonrecursive counter for each fact in the same way as DRed^c, and the main difference is that a fact F is deleted (i.e., added to Δ_D) in line 112 only if no alternative derivation can be found using a combination of backward and forward chaining implemented in functions Check and Saturate. If an alternative derivation is found, F is added to the set P of *proved* facts.

A call to Check(F) searches for the alternative derivations of F using backward chaining. The function maintains the set C of *checked* facts, which ensures that each F is checked only once (line 117 and 124). The procedure first calls Saturate(F) to determine whether F follows from the facts considered thus far; we discuss this step in more detail shortly. If F is not proved, the procedure then examines in lines 119–122 each instance r' of a recursive rule that derives F in the 'old' materialisation, and it tries to prove all body atoms of r' from the current stratum. This involves evaluating rules 'backwards' and, as we already discussed in Section 4.3, this is the main advantage of the B/F algorithm over DRed on a number of complex inputs. The function terminates once F is successfully proved (line 122).

Set P accumulates facts that are checked and successfully proved, and it is computed in function Saturate using forward chaining. Given a fact F that is being checked, it first verifies whether F has a nonrecursive derivation. In the original B/F algorithm, this

is done by evaluating the nonrecursive rules 'backwards' in the same way as in line 10 of DRed. In contrast, B/F^c avoids this by simply checking whether the nonrecursive counter is nonzero (line 125): if that is the case, then F is known to have nonrecursive derivations and it is added to P via lines 126 and 128. If F is proved, the procedure propagates its consequences (line 126–131). In particular, the procedure ensures that each consequence F' of P, the facts in the 'new' materialisation in the previous strata, and the recursive rules is added to P if $F' \in C$, or is added to the set Y of *delayed* facts if $F' \notin C$. Intuitively, set Y contains facts that are proved but that have not been checked yet. If a fact in Y is checked at a later point, it is proved in line 125 without having to apply the rules again.

Since the deletion step of B/F^c is 'exact' in the sense that it deletes precisely those facts that no longer hold after the update, rederivation is not needed. Thus, DeleteUnproved is directly followed by Insert, which is the same as in DRed and DRed^c, with the only difference that B/F^c maintains only the nonrecursive counters.

Algorithm 5 is correct in the same way as B/F since checking whether a fact has a nonzero nonrecursive counter is equivalent to checking whether a derivation of the fact exists by evaluating nonrecursve rules 'backwards'.

4.4 Evaluation

We have implemented the unoptimised and optimised variants of DRed and B/F and have compared them empirically.

Benchmarks

We used the following benchmarks for our evaluation: UOBM [48] is a synthetic benchmark that extends the well known LUBM [33] benchmark; Reactome [15] models biological pathways of molecules in cells; Uniprot [14] describes protein sequences and their functional information; ChemBL [26] represents functional and chemical properties of bioactive compounds; and Claros[1] describes archeological artefacts. Each benchmark consists of a

[1] `http://www.clarosnet.org/`

Dataset		$\lvert E\rvert$	S	$\lvert\Pi_{\mathsf{nr}}\rvert$	$\lvert\Pi_{\mathsf{r}}\rvert$
UOBM-U		254.8 M	5	135	144
UOBM-R		254.8 M	6	164	2,215
Reactome-U		12.5 M	9	814	28
Reactome-R		12.5 M	1	0	21,385
Uniprot-R		123.1 M	5	9,312	2,706
ChemBL-R		289.2 M	3	1,766	499
Claros-LE	Best Worst Average	18.8 M	11	1,031	306

Table 4.1: Benchmark statistics

Dataset		DRed^c	DRed	B/F^c	B/F
UOBM-U		179.24	1,185.87	**1.18**	31.96
UOBM-R		0.29	0.56	0.26	**0.24**
Reactome-U		0.06	1.03	**0.05**	1.00
Reactome-R		26.57	62.46	**0.89**	0.90
Uniprot-R		4.31	8.71	**4.13**	4.36
ChemBL-R		7.91	12.22	1.27	**1.26**
Claros-LE	Best	**0.33**	5,788.75	0.43	8.03
	Worst	5,720.57	5,759.92	**2,802.86**	3,227.05
	Average	1,143.55	1,741.66	**560.61**	653.04

Table 4.2: Average running times for deleting 1000 facts (seconds)

set of facts and an OWL 2 DL ontology, which we transformed into datalog programs of different levels of complexity and recursiveness. More specifically, the *upper bound* (U) programs were obtained using the complete but unsound transformation by Zhou et al. [78], and they entail all consequences of the original ontology but may also derive additional facts. The *recursive* (R) programs were obtained using the sound but incomplete transformation by Kaminski, Nenov, and Grau [42], and they tend to be highly recursive. For Claros, the *lower bound extended* (LE) program was obtained by first taking the datalog subset of the ontology and then manually introducing several 'hard' rules, and it was already used by Motik et al. [51] to compare DRed with B/F. Finally, all the tested programs are recursive, although the percentage of the recursive rules varies. Table 4.1 shows the numbers of facts ($\lvert E\rvert$), strata (S), the nonrecursive rules ($\lvert\Pi_{\mathsf{nr}}\rvert$), and the recursive ones ($\lvert\Pi_{\mathsf{r}}\rvert$) for each benchmark.

Dataset	$\lvert E^- \rvert / \lvert E \rvert$	DRed^c	DRed	B/F^c	B/F
UOBM-U	50%	**1.54 k**	3.66 k	1.64 k	3.11 k
UOBM-R	36%	3.28 k	5.75 k	**2.76 k**	2.87 k
Reactome-U	68%	**30.70**	6.32 k	39.16	6.33 k
Reactome-R	31%	1.07 k	1.78 k	**0.92 k**	0.93 k
Uniprot-R	47%	**1.57 k**	3.47 k	1.92 k	2.86 k
ChemBL-R	69%	4.74 k	7.22 k	**3.25 k**	4.18 k
Claros-LE	8%	5.01 k	17.81 k	**3.49 k**	16.74 k

Table 4.3: Running times for handling large deletions (seconds)

Dataset	Remat	Remat-1C	Remat-2C
UOBM-U	1.56 k	1.60 k	1.61 k
UOBM-R	4.14 k	4.16 k	4.19 k
Reactome-U	30.90	31.23	31.32
Reactome-R	0.91 k	0.91 k	0.92 k
Uniprot-R	1.98 k	1.99 k	2.00 k
ChemBL-R	2.56 k	2.57 k	2.59 k
Claros-LE	3.36 k	3.49 k	3.60 k

Table 4.4: Running times for rematerialisation (seconds)

Test Setup

We conducted all experiments on a Dell PowerEdge R720 server with 256GB RAM and two Intel Xeon E5-2670 2.6GHz processors, running Fedora 24, kernel version 4.8.12. All algorithms handle insertions using the seminaïve evaluation. The only overhead is in counter maintenance, which we measured during initial materialisation (which also uses seminaïve evaluation). Hence, the main focus of our tests was on comparing the performance of our algorithms on 'small' and 'large' deletions. In both cases, we first materialised the relevant program on the explicit facts, and then we performed the following tests.

To test small deletions, we measured the performance on ten randomly selected subsets $E^- \subseteq E$ of 1,000 facts. In all apart from Claros-LE, the running times did not depend significantly on the selected subset of E, so in Table 4.2 we report the average times across all ten runs. On Claros-LE, however, the running times varied significantly, so we report in the table the best, the worst, and the average times. The best number in each row is highlighted in bold.

To test large deletions, we identified the largest subset $E^- \subseteq E$ on which either DRed^c or B/F^c takes roughly the same time as computing the 'new' materialisation from scratch. We measured the performance of all algorithms on E^-. This test allows us to assess the scalability of our algorithms. The running times as well as the percentages of the deleted facts are reported in Table 4.3, with the best performance number in each row highlighted in bold. Finally, to assess the overhead of counter maintenance, we measured the performance of rematerialisation from $(E \setminus E^-)$ with no counters (Remat), just the nonrecursive counter (Remat-1C), and both counters (Remat-2C). The results are presented in Table 4.4.

Discussion

DRed^c outperformed DRed on all inputs for small deletions. In particular, on Reactome-U, the improvement is by several orders of magnitude, albeit unoptimised DRed is already quite efficient. The improvement is also significant in many other cases, including UOBM-U, Reactome-R, and ChemBL-R. In fact, Reactome-U exhibits data and rule patterns outlined in Examples 1, which clearly demonstrates the benefits of eliminating 'backward' rule evaluation. Moreover, the program of Claros-LE contains a symmetric and transitive predicate *relatedPlaces*, so the materialisation contains several large cliques of constants connected by this predicate [51]. When a fact $\mathit{relatedPlaces}(a, b)$ is overdeleted, the DRed algorithm overdeletes all $\mathit{relatedPlaces}(c, d)$ where c and d belong to the same clique as a and b, which requires a cubic number of derivations. However, DRed^c can sometimes (but not always) prove that $\mathit{relatedPlaces}(a, b)$ holds using the nonrecursive counter; as one can see, this can considerably improve the performance by avoiding costly overdeletion.

B/F^c also outperformed B/F for small deletions in many cases: B/F^c was more than 20 times faster for UOBM-U, Reactome-U, and the 'best' case of Claros-LE, which is in line with our observation that 'backwards' rule evaluation can be quite costly. In contrast, on the highly recursive datasets (i.e., all R-datasets), the performance of B/F^c and B/F is roughly the same: the main source of difficulty is due to the recursive rules, whose evaluation is unaffected by the optimisations proposed in this paper.

In terms of average running time, B/F^c outperformed DRed^c for small deletions on all datasets. This is so because B/F^c eagerly identifies alternative derivations of facts, which is often easy, and is beneficial since it can considerably reduce overdeletion. However, DRed^c is still very competitive in many cases, e.g., UOBM-R, Reactome-U, and Uniprot-R. DRed^c is even faster than B/F^c in the 'best' case of Claros-LE. As we discussed earlier in Section 4.1, backward rule evaluation can be a dominant source of inefficiency, and DRed^c is efficient since it completely eliminates backward rule evaluation.

The tests for large deletions show that our algorithms can efficiently delete large subsets of the explicit facts on all but one benchmarks, Claros-LE. Claros-LE is difficult due to the presence of cliques as explained earlier. Nevertheless, DRed^c always considerably outperforms DRed; the difference is particularly significant on Reactome-U, where DRed^c is several orders of magnitude faster. Similarly, B/F^c consistently outperforms B/F on all inputs. While both approaches scale well to large deletions, neither algorithm performs uniformly better than the other in all cases. B/F^c can be more efficient than DRed^c when it is easy to identify alternative derivations of facts during deletion. In contrast, DRed^c can be more efficient than B/F^c in cases where backward rule evaluation is a dominant source of inefficiency: DRed^c is able to completely eliminate backward rule evaluation whereas B/F^c only avoids backward rule evaluation on nonrecursive rules.

Finally, the rematerialisation times show that counter maintenance incurs only modest overheads: Remat-2C was in the worst case only several percent slower than Remat.

5

Modular Materialisation and Maintenance

Our optimised approaches to incremental maintenance achieves better performance than existing solutions by reducing or completely eliminating 'backward' rule evaluation. However, there is still room for improvement. In this chapter we present a modular framework for the efficient computation and maintenance of datalog materialisations that allows custom solutions to be integrated with standard datalog reasoning algorithms. As motivation for this work, in Section 5.1 we demonstrate with two examples how seminaïve evaluation can be suboptimal compared to custom solutions. In Section 5.2 we describe our modular materialisation and incremental maintenance algorithms, as well as various requirements regarding the implementation of modules. In Section 5.3 we introduce an oracle function that provides an opportunity for further optimisation. In Sections 5.4 and 5.5 we present customised implementations for two common types of modules. Finally, Section 5.6 presents our evaluation results. Detailed proofs for all the theorems listed in this chapter are provided in Appendix B.

5.1 Motivation

In this section we show how custom algorithms can handle certain rule combinations much more efficiently than seminaïve evaluation. We consider here only materialisation,

but similar observations apply to incremental maintenance algorithms as most of them use variants of seminaïve evaluation.

Although seminaïve evaluation does not repeat derivations, it always considers each applicable rule instance. However, facts are often derived via multiple, distinct rule instances; this is particularly common with recursive rules, but it can also occur with nonrecursive rules only. We are unaware of a general technique that can prevent such derivations. We next present two programs for which materialisation can be computed without considering all applicable rule instances, thus showing how seminaïve evaluation can be suboptimal.

Example 6. *Let* $E = \{R(c_i, c_{i+1}) \mid 0 \leq i < n\}$ *and let* Π *be the program containing rule* (5.1).

$$R(x, y) \wedge R(y, z) \rightarrow R(x, z) \tag{5.1}$$

Clearly, $I = \mathsf{mat}(\Pi, E) = \{R(c_i, c_j) \mid 0 \leq i < j \leq n\}$*, so each rule instance of the form*

$$R(c_i, c_j) \wedge R(c_j, c_k) \rightarrow R(c_i, c_k) \tag{5.2}$$

with $1 \leq i < j < k \leq n$ *is applicable to* I*. Algorithm 1 considers all of these* $O(n^3)$ *rule instances.*

We next present an outline of an approach that is still cubic in general, but on this specific input runs in $O(n^2)$ *time. The key is to distinguish the set* X *of 'external' facts given to* Π *as input from the 'internal' facts derived by* Π*. We can transitively close* R *by iteratively considering pairs of facts* $R(u, v) \in X$ *and* $R(v, w)$*. That is, we require the first fact to be in* X*, but place no restriction on the second fact. (We could have equivalently required the second fact to be in* X*.) In our example, we have* $X = E$*, so the algorithm considers only rule instances of the form*

$$R(c_i, c_{i+1}) \wedge R(c_{i+1}, c_k) \rightarrow R(c_i, c_k) \tag{5.3}$$

for $0 \leq i < k \leq n$*, of which there are* $O(n^2)$ *many. Intuitively, this is analogous to replacing the predicate* R *in all explicit facts with* X*, and using a linear rule*

$$X(x, y) \wedge R(y, z) \rightarrow R(x, z) \tag{5.4}$$

instead of rule (5.1). *In our approach, however, other rules can derive R-facts so the set X is not fixed; thus, rule* (5.1) *cannot be simply replaced with* (5.4). *Our approach 'simulates' such linearisation, and it can be expected to perform well whenever the other rules derive fewer facts than rule* (5.1).

Example 7. *Let* $E = \{R(c_i, c_{i+1}) \mid 1 \leq i < n\} \cup \{R(c_n, c_1)\}$ *and let* Π *consist of rules* (5.1) *and* (5.5).

$$R(x, y) \rightarrow R(y, x) \tag{5.5}$$

Now $I = \mathsf{mat}(\Pi, E) = \{R(c_i, c_j) \mid 1 \leq i, j \leq n\}$, *so each instance of the form* (5.2) *with* $1 \leq i, j, k \leq n$ *is applicable to* I. *Algorithm 1 considers all of these* $O(n^3)$ *rule instances.*

However, we can view any relation R as an undirected graph with n vertices. To compute the symmetric–transitive closure of R, we first compute the connected components of R, and, for each connected component C, we enumerate all $u, v \in C$ *and derive* $R(u, v)$. *The first step is linear in the size of R and the second step requires* $O(n^2)$ *time, so the algorithm runs in* $O(n^2)$ *time on any R.*

5.2 Framework

In this section we present a general framework for materialisation and incremental maintenance that can avoid the deficiencies outlined in Section 5.1. More specifically, in Section 5.2.1 we define the notion of *module* and *module function*. Our framework splits a program into modules that are subsets of the program. Each module is handled using module functions that compute certain consequences of the module. These functions can be implemented as desired, as long as their results satisfy certain properties that guarantee correctness. We present our framework in two steps: in Section 5.2.2 we consider materialisation, and in Section 5.2.3 we focus on incremental maintenance. Finally, we observe that the order in which module functions are called is not arbitrary but follows certain patterns. This leads us to the notion of *call history* and *module correctness*, which we formalise in Section 5.2.4.

5.2.1 Module Functions

Both materialisation and incremental maintenance algorithms typically process a program in a stratum-by-stratum manner. Hence, we require that all rules of a module can be assigned to one stratum so that our framework can handle a specific module in one stratum only. Note that our notion of modules should not be confused with ontology modules [22, 29, 66]: the latter are subsets of an ontology that are semantically independent from each other in a well-defined way, whereas our modules are just arbitrary programs that satisfy the above constraint.

Definition 8. *A* module *is a semipositive program.*

We next define several operators which we use later to characterise properties that module functions should satisfy. Intuitively, $M^{\uparrow}[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}]$ identifies consequences produced by one round of rule application for fact insertion, whereas $M^{\uparrow}_{\infty}[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}]$ identifies consequences produced by iterative application of rules. As we shall see later, these two operators will be used to characterise the lower and upper bounds, respectively, of the module function responsible for fact insertion. In a similar way, $M_{\downarrow}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}]$ and $M^{\infty}_{\downarrow}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}]$ are defined for fact deletion. Note that iterative application of rules is often costly and constitutes a major source of inefficiency, as suggested by examples 6 and 7. Thus, defining the consequences of such rule application, i.e., $M^{\uparrow}_{\infty}[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}]$ and $M^{\infty}_{\downarrow}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}]$, allows us to optimise fact insertion and deletion: we could replace the costly iterative application of rules with a custom algorithm as long as the two produce the same results.

Definition 9. *Let M be a a module and let I, Δ^{p}, and Δ^{n} be datasets such that $\Delta^{\mathsf{p}} \subseteq I$ and $\Delta^{\mathsf{n}} \cap I = \emptyset$ hold. Then, let $M^{\uparrow}[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}] = \Delta_1$, and $M^{\uparrow}_{\infty}[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}] = \bigcup_{1 \leq i \leq k} \Delta_i$ for k the least integer such that $J_k = J_{k-1}$, where $J_0 = \emptyset$, $\Delta_0 = \Delta^{\mathsf{p}}$, and datasets N_i, Δ_i, and J_i are defined for $i \geq 1$ as*

$$
\begin{aligned}
N_i &= \begin{cases} M\left[I \cup J_0 : \Delta_0, \Delta^{\mathsf{n}}\right] & \text{if } i = 1 \\ M\left[I \cup J_{i-1} : \Delta_{i-1}\right] & \text{if } i > 1, \end{cases} \\
\Delta_i &= N_i \setminus (I \cup J_{i-1}), \\
J_i &= J_{i-1} \cup \Delta_i.
\end{aligned}
$$

Definition 10. *Let M be a module and let I^{p}, I^{n}, Δ^{p}, and Δ^{n} be datasets such that $\Delta^{\mathsf{p}} \subseteq I^{\mathsf{p}}$ and $\Delta^{\mathsf{n}} \cap I^{\mathsf{n}} = \emptyset$ both hold. Then, let $M_{\downarrow}[I^{\mathsf{p}}, I^{\mathsf{n}} \colon \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}] = \Delta_1$, and let $M^{\infty}_{\downarrow}[I^{\mathsf{p}}, I^{\mathsf{n}} \colon \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}}] = \bigcup_{1 \leq i \leq k} \Delta_i$ for k the least integer such that $J_k = J_{k-1}$, where $J_0 = \emptyset$, $\Delta_0 = \Delta^{\mathsf{p}}$, and datasets N_i, J_i, and Δ_i are defined for $i \geq 1$ as*

$$
\begin{aligned}
N_i &= \begin{cases} M\Big[I^{\mathsf{p}} \setminus J_0, I^{\mathsf{n}} \colon \Delta_0, \Delta^{\mathsf{n}}\Big] & \text{if } i = 1 \\ M\Big[I^{\mathsf{p}} \setminus J_{i-1}, I^{\mathsf{n}} \colon \Delta_{i-1}\Big] & \text{if } i > 1, \end{cases} \\
J_i &= J_{i-1} \cup \Delta_{i-1}, \\
\Delta_i &= (N_i \cap I^{\mathsf{p}}) \setminus J_i.
\end{aligned}
$$

In Definition 9, despite that $J_0 = \emptyset$ holds, we wrote $M\Big[I \cup J_0 \colon \Delta_0, \Delta^{\mathsf{n}}\Big]$ instead of $M\Big[I \colon \Delta_0, \Delta^{\mathsf{n}}\Big]$ for the sake of symmetry. Similarly, we wrote $M\Big[I^{\mathsf{p}} \setminus J_0, I^{\mathsf{n}} \colon \Delta_0, \Delta^{\mathsf{n}}\Big]$ instead of $M\Big[I^{\mathsf{p}}, I^{\mathsf{n}} \colon \Delta_0, \Delta^{\mathsf{n}}\Big]$ in Definition 10.

Definition 11. *The* implementation *for a module M consists of module functions Add^M, Del^M, and Red^M. These functions can be used in calls shown in the first column of Table 5.1, where datasets I, I^{p}, I^{n}, Δ^{p}, Δ^{n}, and Δ are arguments and J is the result of a call. These datasets must satisfy the conditions from the second column of Table 5.1.*

Given a dataset E and a program Π satisfying $M \subseteq \Pi$, a call to a module function is correct *in the context of E and Π if the condition from Table 5.2 holds.*

Definition 11 describes module functions and gives conditions that a call of a module function should satisfy. Intuitively, function Add^M handles fact insertion for module M and will be used in both our materialisation and incremental update algorithms; functions Del^M and Red^M deal with fact deletion and rederivation, respectively, and will be used in our incremental maintenance algorithm only. More specifically, call $J := \mathsf{Add}^M[I \colon \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} \colon \Delta^{\mathsf{m}}]$ tries to identify facts that need to be added to the materialisation due to the insertion of facts in $\Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}}$ and the removal of facts in Δ^{n}. We distinguish Δ^{p}, the set of inserted facts derived by other modules, from Δ^{m}, the set of inserted facts derived by the module itself, so that it becomes possible for the module to skip some work that it has done before. This separation is unnecessary for facts in Δ^{n} since each fact in Δ^{n} belongs to a lower stratum and is always derived by another module. Similarly, call $J := \mathsf{Del}^M[I^{\mathsf{p}}, I^{\mathsf{n}} \colon \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} \colon \Delta^{\mathsf{m}}]$ tries to identify facts that need to be deleted from the

Call	Conditions on arguments and results
$J{:=}\ \mathsf{Add}^M[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} : \Delta^{\mathsf{m}}]$	$\Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}} \subseteq I$ and $\Delta^{\mathsf{n}} \cap I = \emptyset$ $J \cap I = \emptyset$
$J{:=}\ \mathsf{Del}^M[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} : \Delta^{\mathsf{m}}]$	$\Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}} \subseteq I^{\mathsf{p}}$ and $\Delta^{\mathsf{n}} \cap I^{\mathsf{n}} = \emptyset$ $J \subseteq I^{\mathsf{p}} \setminus (\Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}})$
$J{:=}\ \mathsf{Red}^M[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta]$	$\Delta \cap I^{\mathsf{p}} = \emptyset$ $J \subseteq \Delta$

Table 5.1: Module function calls

Call	Call is correct in the context of E and Π with $M \subseteq \Pi$ if...
$J{:=}\ \mathsf{Add}^M[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} : \Delta^{\mathsf{m}}]$	$J_l \subseteq J \subseteq J_u$ holds where $J_l = M^{\uparrow}[I : \Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}}, \Delta^{\mathsf{n}}]$ is the *lower bound*, and $J_u = M^{\uparrow}_{\infty}[I : \Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}}, \Delta^{\mathsf{n}}]$ is the *upper bound*.
$J{:=}\ \mathsf{Del}^M[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} : \Delta^{\mathsf{m}}]$	$J_l \subseteq J \subseteq J_u$ holds where $J_l = M_{\downarrow}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}}, \Delta^{\mathsf{n}}] \setminus \mathsf{mat}(\Pi, E)$ is the *lower bound*, and $J_u = M^{\infty}_{\downarrow}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}}, \Delta^{\mathsf{n}}]$ is the *upper bound*.
$J{:=}\ \mathsf{Red}^M[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta]$	$J_l \subseteq J \subseteq J_u$ holds where $J_l = M\left[I^{\mathsf{p}}, I^{\mathsf{n}}\right] \cap \Delta$ is the *lower bound*, and $J_u = \mathsf{mat}(\Pi, E) \cap \Delta$ is the *upper bound*.

Table 5.2: Correctness of module function calls

materialisation due to the removal of facts in $\Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}}$ and the insertion of facts in Δ^{n}. Finally, call $J := \mathsf{Red}^M[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta]$ tries to identify facts in Δ that can be recovered.

A module function call is only correct if it achieves the relevant goal as briefly discussed above. We formalise this notion of correctness by providing lower and uppper bounds for each type of module function calls. For a call of type Add^M this is straightforward: the lower bound is obtained by applying the rules of the module for one round, and the upper bound is obtained by applying the rules iteratively until no more facts can be derived. Regarding a call of type Del^M, for the sake of generality, we would like to allow the module function call to delete only those facts that do not hold in the new

Program	Stratum	Module	Rule	
Π	Π^1	$M^{1,1}$	$A(x) \rightarrow B(x)$	(1)
	Π^2	$M^{2,1}$	$P(x,y) \wedge \text{not}\, B(x) \rightarrow R(x,y)$	(2)
			$R(x,y) \rightarrow S(x,y)$	(3)
			$R(x,y) \rightarrow T(x,y)$	(4)
		$M^{2,2}$	$R(x,y) \wedge R(y,z) \rightarrow R(x,z)$	(5)

Table 5.3: A partition of the program Π

materialisation. Hence, we define the lower bound by applying the rules for one round while excluding the consequences that still hold in the new materialisation. For the same reason, we allow calls of type Red^M to recover from the set of deleted facts Δ as many facts as possible so long as they belong to the new materialisation. To achieve such generality, we discuss correctness of a module function call with respect to certain Π and E, which we refer to as context in Definition 11.

In the next two sections we present our framework for datalog materialisation and incremental maintenance. Let Π be a program and let λ be a stratification that partitions Π into strata $\Pi^1, \ldots, \Pi^S$. To apply our framework, for each $1 \leq s \leq S$, stratum Π^s must be further partitioned into modules $M^{s,1}, \ldots, M^{s,n(s)}$; thus, $n(s)$ is the number of modules in stratum s. Each module $M^{s,k}$ is implemented using functions $\mathsf{Add}^{M^{s,k}}$, $\mathsf{Del}^{M^{s,k}}$, and $\mathsf{Red}^{M^{s,k}}$. Note that $M^{s,k} \cap M^{s',k'} = \emptyset$ holds for all distinct modules $M^{s,k}$ and $M^{s',k'}$. Before we move on to the details of our framework, we use Example 12 to demonstrate a partition of a program that satisfies the above requirements.

Example 12. *Let Π be the program containing rules (1)–(5) in Table 5.3, and let λ be a stratification of Π that partitions Π into two strata, Π^1 and Π^2, where Π^1 contains rule (1) and Π^2 contains rules (2)–(5). Then, we can partition these two strata into three modules. For the first stratum, $M^{1,1}$ contains the only rule in Π^1. For the second stratum, modules $M^{2,1}$ and $M^{2,2}$ consist of rules (2)–(4) and rule (5), respectively. Such a partition clearly satisfy the requirements outlined in the previous paragraph.*

5.2.2 Computing the Materialisation

Algorithm 6 formalises our approach to modular materialisation which uses module functions of type Add^M. It takes as input a program Π, a stratification λ of Π, and a set of explicit facts E, and it computes $\mathsf{mat}(\Pi, E)$. The algorithm processes the program in a stratum-by-stratum manner as in Algorithm 1. For each stratum of Π, the algorithm first adds the explicit facts in the current stratum ($E \cap \mathsf{O}^s$) to the materialisation I (line 145). Then, function $\mathsf{Add}^{M^{s,k}}$ is called once for each module to identify consequences of I (line 147). This is followed by lines 148–155 which are responsible for computing the fixpoint. In each iteration, the algorithm first combines the consequences of all modules (line 149) and adds them to the materialisation I (line 151). Then, it computes for each module all the consequences produced by other modules in the previous round (lines 152–153). Finally, function $\mathsf{Add}^{M^{s,k}}$ is called for each module to identify new consequences of $M^{s,k}$ that should be added to I. The loop terminates when no new fact can be derived (line 150). As mentioned earlier in Section 5.2.1, we make the distinction between facts derived by a module itself (Δ^k) and those derived by others (P^k) to provide the module with additional information which could potentially be useful for improving reasoning performance. Theorem 13 states that Algorithm 6 is correct, and its proof is given in Appendix B.1.

Algorithm 6 MAT-MOD(Π, λ, E)

143: $I := \emptyset$
144: **for each** stratum index s with $1 \leq s \leq S$ **do**
145: $\quad I := I \cup (E \cap \mathsf{O}^s)$
146: $\quad$ **for each** k with $1 \leq k \leq n(s)$ **do**
147: $\quad\quad \Delta^k := \mathsf{Add}^{M^{s,k}}[I : I : \emptyset]$
148: $\quad$ **loop**
149: $\quad\quad \Delta := \Delta^1 \cup \cdots \cup \Delta^{n(s)}$
150: $\quad\quad$ **if** $\Delta = \emptyset$ **then break**
151: $\quad\quad I := I \cup \Delta$
152: $\quad\quad$ **for each** k with $1 \leq k \leq n(s)$ **do**
153: $\quad\quad\quad P^k := \bigcup_{1 \leq j \leq n(s), j \neq k} \Delta^j$
154: $\quad\quad$ **for each** k with $1 \leq k \leq n(s)$ **do**
155: $\quad\quad\quad \Delta^k := \mathsf{Add}^{M^{s,k}}[I : P^k : \Delta^k]$

Theorem 13. *Algorithm 6 computes I as $\mathsf{mat}(\Pi, E)$, provided that each call of a module function made during the algorithm's execution is correct in the context of E and Π.*

5.2.3 Incremental Updates

Our modular approach to incremental materialisation maintenance is based on the DRed^c algorithm presented in Section 4.2, and is formalised by Algorithm 7. The algorithm takes as input a program Π, a stratification λ of Π, a set of explicit facts E, the materialisation $I = \mathsf{mat}(\Pi, E)$, two sets of facts E^- and E^+ to delete from and to add to E, respectively. The goal of the algorithm is to update the materialisation $I = \mathsf{mat}(\Pi, E)$ to the new materialisation $\mathsf{mat}(\Pi, (E \setminus E^-) \cup E^+)$. In addition to the module functions of type Add^M already used in Algorithm 6, our modular incremental maintenance algorithm also employs module functions of type Del^M and Red^M. As usual, the algorithm processes the given program stratum by stratum. We next describe the two main steps of the algorithm for a particular stratum.

In the deletion phase, function $\mathsf{Del}^{M^{s,k}}$ is first called for each module to identify facts that should be overdeleted due to changes in the previous strata (lines 162–163). Together with those to delete from the current stratum ($E^- \cap \mathsf{O}^s$), these facts derived by independent modules are combined into Δ, the initial set of facts to overdelete (line 164). Then, the algorithm computes for each module set P^k containing all facts produced by other modules (lines 165–166). As we shall see later, these sets will be used as arguments in subsequent module function calls. Finally, the loop of lines 167–173 compute all consequences of Δ. In each iteration, function $\mathsf{Del}^{M^{s,k}}$ is first called for each module to identify new consequences of module $M^{s,k}$ that should be overdeleted due to the deletion of P^k and Δ^k (lines 168–169). Then, the algorithm merges Δ, which contains facts to overdelete identified in the previous round, into D, which accummulates all the facts to overdelete. Furthermore, the algorithm updates Δ with all consequences identified in the current round, and it prepares P^k for the next round (lines 171–173). The loop terminates when no new fact can be produced by any module (line 167).

In the second step, the algorithm first identifies the rederivable facts by calling $\mathsf{Red}^{M^{s,k}}$ for each module (lines 175–176). These facts are put into A, the set of added

Algorithm 7 DRed-Mod$(\Pi, \lambda, E, I, E^-, E^+)$

156: $D := A := \emptyset, \quad E^- := (E^- \cap E) \setminus E^+, \quad E^+ := E^+ \setminus E$
157: **for each** stratum index s with $1 \leq s \leq S$ **do**
158: Overdelete
159: Rederive-Insert
160: $E := (E \setminus E^-) \cup E^+, \quad I := (I \setminus D) \cup A$
161: **procedure** Overdelete
162: **for each** k with $1 \leq k \leq n(s)$ **do**
163: $\Delta^k := \mathsf{Del}^{M^{s,k}}[I : D \setminus A, A \setminus D : \emptyset]$
164: $\Delta := (E^- \cap \mathsf{O}^s) \cup \Delta^1 \cup \cdots \cup \Delta^{n(s)}$
165: **for each** k with $1 \leq k \leq n(s)$ **do**
166: $P^k := (E^- \cap \mathsf{O}^s) \cup \bigcup_{1 \leq j \leq n(s), j \neq k} \Delta^j$
167: **while** $\Delta \neq \emptyset$ **do**
168: **for each** k with $1 \leq k \leq n(s)$ **do**
169: $\Delta^k := \mathsf{Del}^{M^{s,k}}[I \setminus (D \setminus A), I \cup A : P^k : \Delta^k]$
170: $D := D \cup \Delta$
171: $\Delta := \Delta^1 \cup \cdots \cup \Delta^{n(s)}$
172: **for each** k with $1 \leq k \leq n(s)$ **do**
173: $P^k := \bigcup_{1 \leq j \leq n(s), j \neq k} \Delta^j$
174: **procedure** Rederive-Insert
175: **for each** i with $1 \leq k \leq n(s)$ **do**
176: $\Delta^k := \mathsf{Red}^{M^{s,k}}[I \setminus (D \setminus A), I \cup A : D \setminus A]$
177: $N := [(E \setminus E^-) \cap (D \cap \mathsf{O}^s)] \cup (E^+ \cap \mathsf{O}^s) \setminus [I \setminus (D \setminus A)]$
178: $A := A \cup N \cup \Delta^1 \cup \cdots \cup \Delta^{n(s)}$
179: **for each** k with $1 \leq k \leq n(s)$ **do**
180: $P^k := [(A \setminus D) \cap \mathsf{O}^{<s}] \cup N \cup \bigcup_{1 \leq j \leq n(s), j \neq k} \Delta^j$
181: **for each** k with $1 \leq k \leq n(s)$ **do**
182: $\Delta^k := \mathsf{Add}^{M^{s,k}}[(I \setminus D) \cup A : P^k, D \setminus A : \Delta^k]$
183: **loop**
184: $\Delta := \Delta^1 \cup \cdots \cup \Delta^{n(s)}$
185: **if** $\Delta = \emptyset$ **then break**
186: $A := A \cup \Delta$
187: **for each** k with $1 \leq k \leq n(s)$ **do**
188: $P^k := \bigcup_{1 \leq j \leq n(s), j \neq k} \Delta^j$
189: **for each** k with $1 \leq k \leq n(s)$ **do**
190: $\Delta^k := \mathsf{Add}^{M^{s,k}}[(I \setminus D) \cup A : P^k : \Delta^k]$

facts, together with those rederivable from the remaining explicit facts and those to insert to the current stratum (lines 177–178). Then, function $\mathsf{Add}^{M^{s,k}}$ is called once for each module (lines 181–182), and this identifies consequences of the facts added to A in line 178, as well as consequences of changes in the previous strata. Finally, the fixpoint is computed in the loop of lines 183–190 analogously to Algorithm 6. Theorem 14 states that Algorithm 7 is correct, and its proof is given in Appendix B.2.

Theorem 14. *Algorithm 7 updates dataset I from $\mathsf{mat}(\Pi, E)$ to $\mathsf{mat}(\Pi, (E \setminus E^-) \cup E^+)$, provided that each module function call made during the execution of the algorithm is correct in the context of $(E \setminus E^-) \cup E^+$ and Π.*

5.2.4 Correctness Conditions for Module Functions

To ensure the correctness of our modular materialisation and incremental maintenance algorithms, Theorems 13 and 14 require each module function to be implemented in such a way that each call of the function made during an execution of Algorithm 6 or Algorithm 7 satisfies the relevant lower and upper bounds described in Section 5.2.1. When we check an implementation of a module function against this requirement, there are two issues that prevent us from examining each of its calls independently of the others. We discuss these two issues separately.

First, the definitions of the lower and upper bounds for the outcome of module function calls sometimes involve the use of E and Π. In particular, set $\mathsf{mat}(\Pi, E)$ is used in both the lower bound for Del calls and the upper bound for Red calls. Sets E and Π are arguments of Algorithm 6 and Algorithm 7, rather than arguments of any module function. This is so because (the implementation of) a module should only concern itself with rules that belong to the module and facts that are relevant. As an example, if we implement function Add for a module using iterative rule application as in the seminaïve algorithm, the implementation will apply the rules of the module to facts that have been produced earlier, and it will never make use of E or Π. As sets E and Π are global, to argue about the correctness of each call of a particular module function, we would have to first quantify over all possible executions of the two algorithms with all possible

combinations of modules and then check if the module function calls produced by an arbitrary such execution satisfy the relevant conditions.

Second, as we shall see later, a module may maintain its own state, in which case the correctness of a call depends on the state that the module has at the time of the call, which in turn depends on the behaviour of previous module function calls made both by the same module and by the others. For example, in Section 5.5 we implement a module that axiomatises a relation as symmetric-transitive by treating the relation as an undirected graph and maintaining the connected components of the graph. The connected components are the state that the module has to correctly maintain. To prove the correctness of such an implementation, we would have to show that the internal state of the module is correctly maintained in an arbitrary sequence of module function calls (for this module) produced by Algorithms 6 and 7.

In the remainder of this section, we discuss how these issues are addressed in our framework. Our objectives are twofold. First, we would like to discuss the correctness of a module without explicitly referring to the details of Algorithms 6 and 7. This will allow us to discard the irrelevant aspects of Algorithms 6 and 7 when proving the correctness of a module. Second, we would like to focus on one module at a time and avoid dealing with multiple modules at the same time. This will simplify the proof of correctness for modules and also greatly facilitate the analysis of module states.

To achieve the above objectives, we next analyse the behaviour of Algorithms 6 and 7 and model module function calls produced by executions of these algorithms as sequences that satisfy certain constraints. While the behaviour of the algorithms depend on all modules, we shift the perspective to one module so that the correctness of one module could be discussed without interfering with the other modules. More specifically, Definition 15 describes sequences of module function calls for a module that is generated by one run of Algorithm 6, followed by arbitrary many runs of Algorithm 7. We consider this to be the typical way in which our framework is used: one first applies Algorithm 6 once to obtain the initial materialisation, and then applies Algorithm 7 arbitrarily many times in order to accommodate changes in the explicit facts.

Definition 15 captures the structural aspects of Algorithms 6 and 7. It describes type constraints that apply to each pair of adjacent calls for the same module. For example, a Del^M call may be followed by another Del^M call or a Red^M call, but not by a call of type Add^M. In addition, Definition 15 outlines conditions that the arguments and results of two adjacent module function calls for the same module must satisfy. For example, for two adjacent Add^M calls, the outcome of the first call will be used as the third argument of the second call.

Definition 15. *A* call history H *for a module* M *is a finite, nonempty sequence of the form* (5.6)*. Each* Q_i*,* $0 \leq i \leq m$*, is a finite, nonempty sequence of the form* (5.7)*, and* $C_{i,j}$*,* $1 \leq j \leq n_i$*, is a call to a module function for* M*.*

$$H = Q_0, \ldots, Q_m \tag{5.6}$$

$$Q_i = C_{i,1}, \ldots, C_{i,n_i} \tag{5.7}$$

Each H *must satisfy the following two conditions.*

- *Call* $C_{0,1}$ *is of type* Add^M *and, for each* $1 \leq i \leq m$*, call* $C_{i,1}$ *is of type* Del^M*.*
- *For each* $0 \leq i \leq m$ *and* $1 \leq j \leq n_i$*, calls* $C_{i,j-1}$ *and* $C_{i,j}$ *must be of one of the forms from Table 5.4, with their arguments and results satisfying the respective conditions.*

Intuitively, each Q corresponds to a sequence of module function calls produced by an execution of either Algorithm 6 or Algorithm 7; an H thus corresponds to a sequence of module function calls produced by several such executions. When considering a call $C_{i,j}$, we often identify the arguments and the results using subscripts i and j. For example, when we consider an Add^M-call $C_{i,j}$, we refer to the arguments by $I_{i,j}$, $\Delta^{\mathsf{p}}_{i,j}$, $\Delta^{\mathsf{n}}_{i,j}$, $\Delta^{\mathsf{m}}_{i,j}$, and to the result by $J_{i,j}$.

In addition to Definition 15, which captures the *structural* conditions imposed by Algorithms 6 and 7 on sequences of module function calls, we use Definition 16 to describe the *semantic* conditions imposed by these algorithms. Roughly speaking, the conditions described in Definition 16 will be satisfied if each module function call returns the correct result during the execution of our framework, in which case the materialisation and incremental maintenance algorithms work as expected. As an example, property (5.12)

$C_{i,j-1}$	$C_{i,j}$	Conditions
$J_{i,j-1} := \mathsf{Del}^M[I^{\mathsf{p}}_{i,j-1}, I^{\mathsf{n}}_{i,j-1} \colon \Delta^{\mathsf{p}}_{i,j-1}, \Delta^{\mathsf{n}}_{i,j-1} \colon \Delta^{\mathsf{m}}_{i,j-1}]$	$J_{i,j} := \mathsf{Del}^M[I^{\mathsf{p}}_i, I^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j}, \Delta^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{m}}_{i,j}]$	$\Delta^{\mathsf{m}}_{i,j} = J_{i,j-1}$ $I^{\mathsf{p}}_{i,j} = I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1})$
$J_{i,j-1} := \mathsf{Del}^M[I^{\mathsf{p}}_{i,j-1}, I^{\mathsf{n}}_{i,j-1} \colon \Delta^{\mathsf{p}}_{i,j-1}, \Delta^{\mathsf{n}}_{i,j-1} \colon \Delta^{\mathsf{m}}_{i,j-1}]$	$J_{i,j} := \mathsf{Red}^M[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \colon \Delta_{i,j}]$	$J_{i,j-1} = \emptyset$ $I^{\mathsf{p}}_{i,j} = I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1})$ $\Delta_{i,j} = I^{\mathsf{p}}_1 \setminus I^{\mathsf{p}}_{i,j}$
$J_{i,j-1} := \mathsf{Red}^M[I^{\mathsf{p}}_{i,j-1}, I^{\mathsf{n}}_{i,j-1} \colon \Delta_{i,j-1}]$	$J_{i,j} := \mathsf{Add}^M[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j}, \Delta^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{m}}_{i,j}]$	$\Delta^{\mathsf{m}}_{i,j} = J_{i,j-1}$ $I_{i,j} = I^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$
$J_{i,j-1} := \mathsf{Add}^M[I_{i,j-1} \colon \Delta^{\mathsf{p}}_{i,j-1}, \Delta^{\mathsf{n}}_{i,j-1} \colon \Delta^{\mathsf{m}}_{i,j-1}]$	$J_{i,j} := \mathsf{Add}^M[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j}, \Delta^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{m}}_{i,j}]$	$\Delta^{\mathsf{m}}_{i,j} = J_{i,j-1}$ $I_{i,j} = I_{i,j-1} \cup \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$

Table 5.4: Conditions on the structure of a call sequence

requires that all facts (up to a certain stratum) that should be deleted after the update are indeed deleted.

Definition 16. *Let Π be a program, let λ be a stratification of Π, let M be a module, let H be a call history for M of the form* (5.6)*, and let $\vec{E} = E_0, \ldots, E_m$ be datasets. Then, Π, λ, M, H, and $\vec{E}$ are* compatible *if there exists exactly one stratum s that satisfies $M \subseteq \Pi^s$, and the following properties hold for each i with $0 \leq i \leq m$, where $U_i = \mathsf{O}^{\leq s} \cap \mathsf{mat}(\Pi, E_i)$.*

For $i = 0$, condition (5.8) *must hold. Moreover, if $m > 0$, then conditions* (5.9)*–*(5.10) *must hold as well.*

$$I_{0,1} = \Delta^{\mathsf{p}}_{0,1} \text{ and } \Delta^{\mathsf{m}}_{0,1} = \emptyset \tag{5.8}$$

$$U_0 = \bigcup_{1 \leq k \leq n_0} (\Delta^{\mathsf{p}}_{0,k} \cup \Delta^{\mathsf{m}}_{0,k}) \tag{5.9}$$

$$J_{0,n_0} = \emptyset \tag{5.10}$$

For i with $1 \leq i < m$, condition (5.11) *must hold. Moreover, there exists exactly one j with $1 \leq j \leq n_i$ such that call $C_{i,j}$ is of type* Red*. Thus, all calls before j are of type* Del *and there is at least one such call, all calls after j are of type* Add*, there is at least one such call, and properties* (5.12)*–*(5.14) *must hold.*

$$\mathsf{O}^{\leq s} \cap I_{i-1,n_{i-1}} = \mathsf{O}^{\leq s} \cap I^{\mathsf{p}}_{i,1} \text{ and } \Delta^{\mathsf{m}}_{i,1} = \emptyset \tag{5.11}$$

$$U_{i-1} \setminus U_i \subseteq \bigcup_{1 \leq k \leq j-1} (\Delta^{\mathsf{p}}_{i,k} \cup \Delta^{\mathsf{m}}_{i,k}) \tag{5.12}$$

$$U_i \setminus I^{\mathsf{p}}_{i,j} = \bigcup_{j+1 \leq k \leq n_i} (\Delta^{\mathsf{p}}_{i,k} \cup \Delta^{\mathsf{m}}_{i,k}) \tag{5.13}$$

$$J_{i,n_i} = \emptyset \tag{5.14}$$

For $i = m > 0$, condition (5.11) *must hold. Moreover, there exists at most one j with $1 \leq j \leq n_m$ such that $C_{m,j}$ is of type* Red*; if such a j does exist, then condition* (5.12) *must hold.*

Recall that when module functions are implemented, it is only necessary that they return correct results when used in calls made during the execution of our framework. Since such calls have already been modeled in Definitions 15 and 16, so we are now ready

to define the correctness of a module based on these definitions (and without referring to Algorithms 6 and 7).

Definition 17. *Functions* Add^M, Del^M, *and* Red^M *for a module* M *are* correct *if, for each program* Π*, each stratification* λ *of* Π*, each call history for* M *of the form* (5.6)*, and each vector* $\vec{E} = E_0, \ldots, E_m$ *of datasets such that* Π*,* λ*,* M*,* H*, and* $\vec{E}$ *are compatible, each call* $C_{i,j}$*, for* $0 \leq i \leq m$ *and* $1 \leq j \leq n_i$*, in* H *is correct in the context of* E_i *and* Π*.*

Now that we have all the relevant definitions, we are ready to formulate Theorem 18, which states that our modular framework is correct. The proof is given in Appendix B.3.

Theorem 18. *Consider an arbitrary program* Π*, stratification* λ *of* Π*, dataset* E_0*, and datasets* E_i^- *and* E_i^+ *with* $1 \leq i \leq m$*. Let* $E_i = E_{i-1} \setminus E_i^- \cup E_i^+$ *for* $1 \leq i \leq m$*. Moreover, let* $M^{s,k}$ *be the modules corresponding to* Π *and* λ *as described in Section 5.2.2, and assume that the module functions for each* $M^{s,k}$ *are correct. Let* I_0 *be the result of applying Algorithm 6 to* Π*,* λ*, and* E_0*, and let* I_i *be the result of successively applying Algorithm 7 to* Π*,* λ*,* E_i^-*, and* E_i^+ *for* $1 \leq i \leq m$*. Then,* $I_i = \mathsf{mat}(\Pi, E_i)$ *holds for each* $0 \leq i \leq m$*.*

5.3 Further Optimisation

Recall that both the DRed^c Algorithm and the B/F^c Algorithm presented earlier in Chapter 4 make use of a nonrecursive counter to track the number of nonrecursive derivations for a fact. This counter could be effectively used to limit overdeletion: if the nonrecursive counter of a fact is positive, we know that the fact has an unaffected nonrecursive derivation and thus does not have to deleted. In our work, this means that nonrecursive and recursive modules have to exchange information about whether a fact is derivable or not. We achieve this by introducing an oracle function (Definition 19) that is able to decide whether a fact still holds after an update. It could be straightforwardly implemented using nonrecursive counters, so in the remainder of this chapter, we assume the existence of such an oracle function when implementing a module.

Definition 19. *An* oracle function T *is* correct *if, whenever some module M makes a call to some module function in the context of some E and Π, and $T(F)$ returns true during the call, then $F \in mat(\Pi, E)$ holds.*

5.4 Transitive Closure

We now consider a module consisting of a single rule (5.1) axiomatising a relation R as transitive. Following the ideas from Example 6, we distinguish the 'internal' facts produced by rule (5.1) from the 'external' facts produced by other rules. We keep track of the latter in a global set X_R that is initialised to the empty set. A key invariant of our approach is that each fact $R(a_0, a_n)$ is produced by a chain $\{R(a_0, a_1), \ldots, R(a_{n-1}, a_n)\} \subseteq X_R$ of 'external' facts. Thus, we can transitively close R by considering pairs of R-facts where at least one of them is contained in X_R, which can greatly reduce the number of inferences. A similar effect could be achieved by rewriting the input program: we introduce a fresh predicate X_R, and we replace by X_R each occurrence of R in the head of a rule, as well as one of the two occurrences of R in the body of rule (5.1). Such an approach, however, introduces the facts containing the auxiliary predicate X_R into the materialisation and thus reveals implementation details to the users. Moreover, the rederivation step can be realised very efficiently in our approach.

Based on the above idea, function $\mathsf{Add}^{\mathsf{tc}(R)}$, shown in Algorithm 8, essentially implements seminaïve evaluation for rule (5.4): the loops in lines 192–193 and 194–197 handle the two delta rules derived from (5.4). Function $\mathsf{Del}^{\mathsf{tc}(R)}$, shown in Algorithm 9, implements seminaïve evaluation for rule (5.4) analogously to $\mathsf{Add}^{\mathsf{tc}(R)}$; it also implements seminaïve evaluation for the rule $R(x, y) \wedge X(y, z) \rightarrow R(x, z)$. Another difference is that only facts whose nonrecursive counter is zero are overdeleted, which mimics overdeletion in DRed^c. As a result, not all facts processed in lines 203 and 207 are added to J so, to avoid repeatedly considering such facts, the algorithm maintains the set S of 'seen' facts. Finally, function $\mathsf{Red}^{\mathsf{tc}(R)}$, shown in Algorithm 10, identifies for each source vertex u all vertices reachable by the external facts in X_R.

Theorem 20. *Functions* $\mathsf{Add}^{\mathsf{tc}(R)}$, $\mathsf{Del}^{\mathsf{tc}(R)}$, *and* $\mathsf{Red}^{\mathsf{tc}(R)}$ *are correct for the module that axiomatises relation* R *as transitive, provided that the oracle function* T *is correct.*

Algorithm 8 $\mathsf{Add}^{\mathsf{tc}(R)}[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} : \Delta^{\mathsf{m}}]$

191: $J := \emptyset, \quad Q := R[\Delta^{\mathsf{p}}], \quad X_R := X_R \cup R[\Delta^{\mathsf{p}}]$
192: **for each** $R(u, v) \in \Delta^{\mathsf{p}}$ and **each** $R(v, w) \in I \setminus \Delta^{\mathsf{p}}$ **do**
193: **if** $R(u, w) \notin I \cup J$ **then add** $R(u, w)$ to Q and J
194: **while** $Q \neq \emptyset$ **do**
195: **remove** an arbitrarily chosen fact $R(v, w)$ from Q
196: **for each** $R(u, v) \in X_R$ such that $R(u, w) \notin I \cup J$ **do**
197: **add** $R(u, w)$ to Q and J
198: **return** J

Algorithm 9 $\mathsf{Del}^{\mathsf{tc}(R)}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} : \Delta^{\mathsf{m}}]$

199: $J := \emptyset, \quad Q := S := R[\Delta^{\mathsf{p}}]$
200: **while** $Q \neq \emptyset$ **do**
201: **remove** an arbitrarily chosen fact $R(u, v)$ from Q
202: **for each** $R(v, w) \in X_R$ such that $R(u, w) \in I^{\mathsf{p}} \setminus S$ **do**
203: **add** $R(u, w)$ to Q and S
204: **if** $T(R(u, w)) = \mathsf{f}$ **then add** $R(u, w)$ to J
205: **else add** $R(u, w)$ to Y_R
206: **for each** $R(t, u) \in X_R$ such that $R(t, v) \in I^{\mathsf{p}} \setminus S$ **do**
207: **add** $R(t, v)$ to Q and S
208: **if** $T(R(t, v)) = \mathsf{f}$ **then add** $R(t, v)$ to J
209: **else add** $R(t, v)$ to Y_R
210: $X_R := X_R \setminus (\Delta^{\mathsf{p}} \cup J)$
211: **return** $J \cap (I^{\mathsf{p}} \setminus (\Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}}))$

Algorithm 10 $\mathsf{Red}^{\mathsf{tc}(R)}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta]$

212: $X_R := X_R \cup Y_R, \quad J := Y_R := \emptyset$
213: **for each** u such that there exist v with $R(u, v) \in \Delta$ **do**
214: **for each** w reachable from u via R facts in X_R **do**
215: **add** $R(u, w)$ to J
216: **return** $J \cap \Delta$

5.5 Symmetric–Transitive Closure

We now consider a module consisting of two rules, (5.1) and (5.5), axiomatising a relation R as transitive and symmetric. As in Example 7, we can view relation R as an undirected graph. To compute the materialisation, we extract the set C_R of connected components—that is, each $U \in C_R$ is a set of mutually connected vertices in the symmetric–transitive

closure of R; finally, we derive $R(u, v)$ for all u and v in each component $U \in C_R$. Set C_R is global and is initially empty.

Based on this idea, function $\mathsf{Add}^{\mathsf{stc}(R)}$, shown in Algorithm 11, uses an auxiliary function CLOSEEDGES to incrementally update the set C_R by processing each fact $R(u, v) \in \Delta$ in lines 220–228: if either u or v does not occur in a component in C_R, then the respective component is created in C_R (lines 222 and 224); and if u and v belong to distinct components U and V, then U and V are merged into a single component and all R-facts connecting U and V are added (lines 225–228). Function $\mathsf{Del}^{\mathsf{stc}(R)}$, shown in Algorithm 12, simply overdeletes all facts $R(u', v')$ whose nonrecursive counter is zero and where both u' and v' belong to a component U containing both vertices of a fact $R(u, v)$ in Δ. Those facts $R(u', v')$ for which the nonrecursive counter is nonzero will hold after overdeletion, so they are kept in an initially empty global set Y_R so that they can be used for rederivation later. Finally, function $\mathsf{Red}^{\mathsf{stc}(R)}$, shown in Algorithm 13, simply closes the set Y_R in the same way as during addition, and it empties the set Y_R. While this creates a dependency between $\mathsf{Del}^{\mathsf{stc}(R)}$ and $\mathsf{Red}^{\mathsf{stc}(R)}$, the order in which these functions are called in Algorithm 7 ensures that the set Y_R is maintained correctly.

Theorem 21. *Functions* $\mathsf{Add}^{\mathsf{stc}(R)}$, $\mathsf{Del}^{\mathsf{stc}(R)}$, *and* $\mathsf{Red}^{\mathsf{stc}(R)}$ *are correct for the module that axiomatises R as symmetric–transitive, provided that the oracle function T is correct.*

Algorithm 11 $\mathsf{Add}^{\mathsf{stc}(R)}[I : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} : \Delta^{\mathsf{m}}]$

```
217: return CloseEdges(Δ^p \ Δ^m) \ I
218: function CloseEdges(Δ)
219:     J := ∅
220:     for each R(u, v) ∈ Δ do
221:         if no U ∈ C_R exists such that u ∈ U then
222:             add {u} to C_R, and R(u, u) to J
223:         if no V ∈ C_R exists such that v ∈ V then
224:             add {v} to C_R, and R(v, v) to J
225:         if u and v belong to distinct U, V ∈ C_R, resp. then
226:             remove U and V from C_R, and add U ∪ V to C_R
227:             for each u' ∈ U and each v' ∈ V do
228:                 add R(u', v') and R(v', u') to J
229:     return J
```

Algorithm 12 $\mathsf{Del}^{\mathsf{stc}(R)}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta^{\mathsf{p}}, \Delta^{\mathsf{n}} : \Delta^{\mathsf{m}}]$

230: $J := \emptyset$
231: **for each** $U \in C_R$ where $\exists R(u, v) \in \Delta^{\mathsf{p}} \setminus \Delta^{\mathsf{m}}$ s.t. $u, v \in U$ **do**
232: **for each** $u' \in U$ and **each** $v' \in U$ **do**
233: **if** $T(R(u', v')) = \mathsf{f}$ **then add** $R(u', v')$ to J
234: **else add** $R(u', v')$ to Y_R
235: **remove** U from C_R
236: **return** $J \cap (I^{\mathsf{p}} \setminus (\Delta^{\mathsf{p}} \cup \Delta^{\mathsf{m}}))$

Algorithm 13 $\mathsf{Red}^{\mathsf{stc}(R)}[I^{\mathsf{p}}, I^{\mathsf{n}} : \Delta]$

237: $J := \textsc{CloseEdges}(Y_R) \setminus I^{\mathsf{p}}$
238: $Y_R := \emptyset$
239: **return** $J \cap \Delta$

Theorem 22 states that the module functions for symmetric-transitive modules have quadratic running time. For each of these module functions, we consider as input the arguments of the function. For example, the size of the input for module function $\mathsf{Add}^{\mathsf{stc}(R)}$ is simply $n = |I| + |\Delta^{\mathsf{p}}| + |\Delta^{\mathsf{n}}| + |\Delta^{\mathsf{m}}|$.

Theorem 22. *Functions* $\mathsf{Add}^{\mathsf{stc}(R)}$, $\mathsf{Del}^{\mathsf{stc}(R)}$, *and* $\mathsf{Red}^{\mathsf{stc}(R)}$ *have running time* $O(n^2)$ *for* n *the size of the input.*

5.6 Evaluation

We have implemented our modular materialisation and incremental maintenance algorithms, as well as the seminaïve materialisation and the DRed^c algorithms, and we have compared their performance empirically.

Test Benchmarks

We used the following real-world and synthetic benchmarks in our tests. Claros-LE was already described in Section 4.4. LUBM [33] is a well-known benchmark that models individuals and organisations in a university domain. Similarly to Claros-LE, we obtained the *lower bound extended* (-LE) program for LUBM by converting a subset of the accompanying OWL ontology into datalog rules and manually extended them with several 'difficult rules'. We chose this dataset over the UOBM datasets used in

Benchmark	$\lvert E\rvert$	$\lvert I\rvert$	S	$\lvert\Pi_{\mathsf{nr}}\rvert$	$\lvert\Pi_{\mathsf{r}}\rvert$	$\lvert\mathrm{TC}\rvert$	$\lvert\mathrm{STC}\rvert$
Claros-LE	18.8 M	533.3 M	11	1031	306	27	2
LUBM-LE	133.6 M	332.6 M	5	85	22	1	2
DBpedia-SKOS	5.0 M	97.0 M	5	26	15	2	1
DAG-R	0.1 M	22.9 M	1	1	1	1	0

Table 5.5: Benchmark statistics

Benchmark	Mat-Mod	Mat
Claros-LE	896.67	4034.92
LUBM-LE	404.99	1352.37
DBpedia-SKOS	121.60	4315.10
DAG-R	34.56	3727.62

Table 5.6: Running times for materialisation computation (seconds)

Benchmark	Small Deletions		Small Insertions		Large Deletions	
	DRed^c-Mod	DRed^c	DRed^c-Mod	DRed^c	DRed^c-Mod	DRed^c
Claros-LE	1.22	1216.86	0.25	1.11	358.47	4288.69
LUBM-LE	0.64	4.80	0.02	0.02	213.49	1612.28
DBpedia-SKOS	29.80	856.92	0.29	3.36	205.76	4669.25
DAG-R	101.10	3493.52	19.70	128.50	160.37	4807.37

Table 5.7: Running times for incremental maintenance (seconds)

Section 4.4 since LUBM-LE contains rules that axiomatise a relation as both symmetric and transitive, which allows us to evaluate the performance of our implementation for symmetric-transitive modules. DBpedia [44] contains structured information extracted from Wikipedia. DBpedia represents Wikipedia categories using the SKOS vocabulary [50], which defines several transitive properties. We used the datalog subset of the SKOS RDF schema. Moreover, the materialisation of DBpedia-SKOS is too large to fit into the memory of our test server, so we used a random sample of the DBpedia dataset consisting of five million facts. Finally, DAG-R is a synthetic benchmark consisting of a randomly generated dataset containing a directed acyclic graph with 10k nodes and 100k edges, and a program that axiomatises the path relation as transitive. Table 5.5 shows the numbers of explicit facts ($\lvert E\rvert$), derived facts ($\lvert I\rvert$), strata (S), nonrecursive rules ($\lvert\Pi_{\mathsf{nr}}\rvert$), recursive rules ($\lvert\Pi_{\mathsf{r}}\rvert$), transitivity modules ($\lvert\mathrm{TC}\rvert$), and symmetric–transitive modules ($\lvert\mathrm{STC}\rvert$) for each benchmark.

Test Setup and Results

We conducted all experiments on a Dell PowerEdge R720 server with 256GB RAM and two Intel Xeon E5-2670 2.6GHz processors, running Fedora 30, kernel version 5.0.17. For each benchmark, we loaded the test data into our system and then compared the performance of our modular algorithms with the seminaïve and DRed^c algorithms using the following methodology.

We first computed the materialisation and measured the wall-clock time. The results are shown in Table 5.6. We then conducted two groups of incremental reasoning tests.

In the first group, we tested the performance of our incremental algorithms on small changes. To this end, we used uniform sampling to select ten subsets $E_i \subseteq E$, $1 \leq i \leq 10$, each consisting of 1000 facts from the input dataset. We deleted and then reinserted E_i for each i while measuring the wall-clock times, and then we computed the average times for deletion and insertion over the ten samples. The results are shown in the 'Small Deletions' and 'Small Insertions' columns of Table 5.7, respectively.

In the second group, we tested the performance of incremental algorithms on large deletions. To this end, we used uniform sampling to select a subset $E^- \subseteq E$ containing 25% of the explicit facts, and we measured the wall-clock time needed to delete E^- from the materialisation. The results are shown in the 'Large Deletions' column of Table 5.7. We did not consider large insertions because our algorithms handle insertion in the same way as materialisation, so the relative performance of our algorithms should be similar to the performance of materialisation shown in Table 5.6.

Discussion

Mat-Mod significantly outperformed Mat on all test inputs. For example, Mat-Mod was several times faster than Mat on Claros-LE and LUBM-LE. The programs of both benchmarks contain transitivity and symmetric–transitivity modules, which are efficiently handled by our custom algorithm. The performance improvement is even more significant for DBpedia-SKOS and DAG-R: Mat-Mod is more than 30 times faster than Mat on

DBpedia-SKOS, and the difference reaches two orders of magnitude on DAG-R. In fact, DBpedia contains long chains/cycles over the *skos:broader* relation [9], which is axiomatised as transitive in SKOS. Mat-Mod outperforms Mat in this case since our custom algorithm for transitivity skips a large number of rule instances. The same observation explains the superior performance of DAG-R.

Similarly, DRed^c-Mod considerably outperformed DRed^c on small deletions: the performance speedup ranges around eight times on LUBM-LE to three orders of magnitude on Claros-LE. The program of Claros-LE contains a symmetric–transitive closure module for the predicate *relatedPlaces*, and the materialisation contains large cliques of constants connected to each other via this predicate. Thus, when a *relatedPlaces(a,b)* fact is deleted, DRed^c can end up considering up to n^3 rule instances where n is the number of constants in the clique containing a and b. In contrast, our custom algorithm for this module maintains a connected component for the clique and requires only up to n^2 steps. It is worth noticing that, while DRed^c-Mod significantly outperforms DRed^c on DAG-R, the incremental update times for small deletion were larger than the update times for the initial materialisation. This is because deleting one thousand edges from the graph ('Small Deletion') caused a large part of the materialisation to be overdeleted and rederived again. For DRed^c the situation is similar, but rederivation in DRed^c benefits from a global recursive counter (at the expense of considering each applicable rule instance), which makes small deletion still faster than initial materialisation. Finally, as shown in Table 5.7, DRed^c-Mod scaled well and maintained its advantage over DRed^c on large deletions.

Incremental insertions are in general easier to handle than deletions since during insertion the algorithms can rely on the whole materialisation to prune the propagation of facts whereas during deletion the algorithms can only rely on the nonrecursive counters of facts to do the same. This is clearly reflected in Table 5.7. Nevertheless, in our tests for small insertions, DRed^c-Mod was several times faster than DRed^c in all cases but LUBM-LE, for which both algorithms updated the materialisation instantateously.

6
Conclusion and Outlook

In this thesis we have proposed several novel approaches to the computation and maintenance of datalog materialisations. Existing incremental maintenance algorithms either lack generality, or involve costly 'backward' rule evaluation, so we have presented two new algorithms, DRed^c and B/F^c, which avoid these drawbacks. Moreover, we have observed that for both materialisation computation and incremental maintenance, certain types of rules and rule combinations can be handled much more efficiently using custom solutions than using general purpose algorithms. To enable the integration of such custom solutions into standard datalog reasoning methods, we have introduced a modular framework that deals with both materialisation computation and incremental maintenance. In addition, we have implemented custom solutions for programs axiomatising the transitive and symmetric-transitive closure of a relation.

Both DRed^c and B/F^c are capable of updating materialisations of general datalog programs. The DRed^c algorithm combines DRed with Counting. It maintains two counters for each fact: the maintenance of the nonrecursive counters captures the behaviour of Counting on nonrecursive rules and also helps reduce overdeletion, whereas the maintenance of the recursive counters allows the algorithm to completely eliminate 'backward' rule evaluation. The B/F^c algorithm can be viewed as a combination of B/F and Counting. It eliminates 'backward' rule evaluation for nonrecursive rules while

still allowing backward chaining to be used on recursive rules in order to make deletion exact. Our evaluation shows that both DRed^c and B/F^c are generally more efficient than their optimised counterparts, DRed and B/F, respectively, often by orders of magnitude. Moreover, our algorithms could handle both small and large updates efficiently, and have thus been shown to be ready for practical use.

Building upon the seminaïve algorithm for materialisation computation and the DRed^c algorithm for incremental maintenance, we have presented a modular framework that allows specialised algorithms for certain types of programs to work in conjunction with standard datalog reasoning methods. Our framework provides a clear interface to application developers and thus permits smooth integration of custom solutions. More specifically, it partitions a datalog program into separate modules and requires three functions to be implemented for each of these modules: function Add is responsible for fact insertion during both materialisation and incremental update; function Red and Del deal with rederivation and deletion, respectively, during incremental update. There is no restriction on how these functions are implemented for a module, so long as they return results satisfying relevant lower and upper bounds when called by our framework. If no custom solution is available for a module, these functions can always be implemented with the seminaïve evaluation strategy, which means that our framework is general enough to handle arbitrary combinations of rules.

We demonstrate with two types of modules that the integration of custom solutions and general datalog reasoning algorithms could indeed be achieved in a straightforward manner through the use of our modular framework. In particular, we implement the first type of modules, which axiomatises a relation as transitive, by distinguishing facts derived by the module itself from facts derived by other modules. This allows us to simulate the behaviour of linear rules and thus skip many superfluous derivations. Our second type of modules, which axiomatises a relation as symmetric-transitive, is implemented by treating the relation as an undirected graph and maintaining its connected components. This custom solution needs $O(n^2)$ steps in the worst case, instead of $O(n^3)$ steps required by the seminaïve algorithm. It is worth pointing out that our modular framework could work with any type of modules, but we chose to implement these two specific

types of modules as a proof of concept for two main reasons. First, they represent two basic types of recursive definitions that datalog is capable of expressing. Second, there already exist (custom) algorithms that compute and maintain transive and symmetric-transitive relations, so deemed it feasible to develop efficient module implementations for transitive and symmetric-transitive modules. And this was indeed the case: we have shown empirically that, with our custom solutions integrated, the materialisation and incremental maintenance algorithms significantly outperform the existing ones, sometimes by orders of magnitude.

We conclude by outlining several directions for future work. First, the recently proposed FBF algorithms [52] can be seen as a generalisation of both the DRed and the B/F algorithm. Since our DRed^c and B/F^c algorithms build upon DRed and B/F, respectively, it would be interesting to see if a generalisation similar to FBF could be obtained for these two new algorithms. Moreover, the evaluation results presented in Section 4.4 suggest that DRed^c performs better than B/F^c on some input while B/F^c achieves better performance on others. When derivations of facts could be easily identified, B/F^c usually performs better since it avoids overdeletion by eagerly checking alternative derivations of facts. In contrast, when it is difficult to identify derivations of facts and backward chaining becomes a dominant source of inefficiency, DRed^c is more efficient since it does not involve any 'backward' rule evaluation. Thus, devising a strategy that automatically decides which algorithm to choose for a given dataset might improve the performance of datalog systems.

In this thesis we have implemented two types of modules as a starting point. To widen the applicability of our modular framework, we hope to be able to implement other types of modules in future. A particularly interesting type of modules is the class of regular chain programs, whose materialisation and incremental maintenance can be handled using custom algorithms [21, 49, 76]. Whether these custom algorithms could be readily adapted to implementations of module functions that satisfy the relevant constraints remains to be investigated.

Appendices

A

Proof of Theorems in Chapter 4

A.1 Proof of Theorem 4

Theorem 4. *Algorithm 4 updates dataset I from* $\mathsf{mat}(\Pi, E)$ *to* $\mathsf{mat}(\Pi, (E \setminus E^-) \cup E^+)$*, and it updates* C_{nr} *and* C_{r} *so they are compatible with* Π*,* λ*, and* $(E \setminus E^-) \cup E^+$*.*

Proof. Let $\ominus$ be the multiset subtraction operator, and let $\mathsf{Occ}(F, M)$ be the multiplicity of F in multiset M. Due to line 67 of Algorithm 4, without loss of generality we assume that $E^- \subseteq E$ and $E^+ \cap E = \emptyset$. Now let $E|_o = E$ and let $I^0|_o = \emptyset$. Moreover, for each $1 \leq s \leq S$, let $I^s_0|_o, I^s_1|_o, \ldots$ be the sequence of sets where $I^s_0|_o = I^{s-1}|_o \cup (E|_o \cap \mathsf{O}^s)$, and for $i > 0$, $I^s_i|_o = I^s_{i-1}|_o \cup \Pi^s\left[I^s_{i-1}|_o\right]$. Index k clearly exists at which the sequence reaches the fixpoint (i.e., $I^s_k|_o = I^s_{k+1}|_o$), so let $I^s|_o = I^s_k|_o$. Finally, let $I|_o = I^S|_o$; we clearly have $I|_o = \mathsf{mat}(\Pi, E|_o)$—that is, $I|_o$ is the 'old' materialisation. Now let $E|_n = (E|_o \setminus E^-) \cup E^+$, and let $I^s_i|_n$, $I^s|_n$, and $I|_n$ be defined analogously, so $I|_n$ is the 'new' materialisation.

For each $1 \leq s \leq S$ and each $F \in I|_o \cap \mathsf{O}^s$, let $C_{\mathsf{nr}}[F]|_o = \mathsf{Occ}(F, E|_o \oplus \Pi^s_{\mathsf{nr}}[\![I|_o]\!])$ and $C_{\mathsf{r}}[F]|_o = \mathsf{Occ}(F, \Pi^s_{\mathsf{r}}[\![I|_o]\!])$. For each $1 \leq s \leq S$ and each $F \in I|_n \cap \mathsf{O}^s$, we define $C_{\mathsf{nr}}[F]|_n$ and $C_{\mathsf{r}}[F]|_n$ analogously using $E|_n$ and $I|_n$.

Now consider a run of Algorithm 4 on $I|_o$, E^-, and E^+. Let $D^0 = A^0 = R^0 = \emptyset$, and for each s with $1 \leq s \leq S$, let D^s, A^s, and R^s be the values of D, A, and R, respectively,

after the loop in lines 68–71 finishes for stratum index s. Note that during the execution of Algorithm 4, the set I is equal to $I|_o$ up to before line 72. Furthermore, for each fact $F \in I^s|_o \cap \mathsf{O}^s$, let $C_{\mathsf{nr}}[F]|_d$ and $C_{\mathsf{r}}[F]|_d$ be the values of $C_{\mathsf{nr}}[F]$ and $C_{\mathsf{r}}[F]$, respectively, at the point when OVERDELETE finishes for stratum index s; similarly, for each fact $F \in ((I^s|_o \setminus D^s) \cup A^s) \cap \mathsf{O}^s$, let $C_{\mathsf{nr}}[F]|_a$ and $C_{\mathsf{r}}[F]|_a$ be the values of $C_{\mathsf{nr}}[F]$ and $C_{\mathsf{r}}[F]$, respectively, at the point when INSERT finishes for stratum index s.

$$\Pi^s_{\mathsf{nr}}\left[\!\left[I^s|_o\right]\!\right] = \Pi^s_{\mathsf{nr}}\left[\!\left[I^s|_o \colon D^s \setminus A^{s-1}, A^{s-1} \setminus D^s\right]\!\right] \oplus \Pi^s_{\mathsf{nr}}\left[\!\left[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\right]\!\right] \tag{A.1}$$

$$\Pi^s_{\mathsf{r}}\left[\!\left[I^s|_o\right]\!\right] = \Pi^s_{\mathsf{r}}\left[\!\left[I^s|_o \colon D^s \setminus A^{s-1}, A^{s-1} \setminus D^s\right]\!\right] \oplus \Pi^s_{\mathsf{r}}\left[\!\left[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\right]\!\right] \tag{A.2}$$

$$C_{\mathsf{nr}}[F]|_o - C_{\mathsf{nr}}[F]|_d = \mathsf{Occ}(F, (E^- \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}\left[\!\left[I^s|_o \colon D^s \setminus A^{s-1}, A^{s-1} \setminus D^s\right]\!\right]) \quad \text{for each } F \in I^s|_o \cap \mathsf{O}^s \tag{A.3}$$

$$C_{\mathsf{r}}[F]|_o - C_{\mathsf{r}}[F]|_d = \mathsf{Occ}(F, \Pi^s_{\mathsf{r}}\left[\!\left[I^s|_o \colon D^s \setminus A^{s-1}, A^{s-1} \setminus D^s\right]\!\right]) \quad \text{for each } F \in I^s|_o \cap \mathsf{O}^s \tag{A.4}$$

$$I^s|_o \setminus I^s|_n \subseteq D^s \subseteq I^s|_o \tag{A.5}$$

$$\mathsf{O}^s \cap D^s \cap \Pi^s_{\mathsf{r}}\left[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\right] \subseteq R^s \subseteq I^s|_n \tag{A.6}$$

$$I^s|_o \cap A^s \subseteq D^s \tag{A.7}$$

$$(I^s|_o \setminus D^s) \cup A^s = I^s|_n \tag{A.8}$$

$$C_{\mathsf{nr}}[F]|_a - C_{\mathsf{nr}}[F]|_d = \mathsf{Occ}(F, (E^+ \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}\left[\!\left[(I^s|_o \setminus D^s) \cup A^s\right]\!\right] \ominus \Pi^s_{\mathsf{nr}}\left[\!\left[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\right]\!\right]) \quad \text{for each } F \in ((I^s|_o \setminus D^s) \cup A^s) \cap I^s|_o \cap \mathsf{O}^s \tag{A.9}$$

$$C_{\mathsf{nr}}[F]|_a = \mathsf{Occ}(F, ((E|_o \setminus E^-) \cup E^+) \oplus \Pi^s_{\mathsf{nr}}\left[\!\left[(I^s|_o \setminus D^s) \cup A^s\right]\!\right]) \quad \text{for each } F \in (((I^s|_o \setminus D^s) \cup A^s) \setminus I^s|_o) \cap \mathsf{O}^s \tag{A.10}$$

$$C_{\mathsf{r}}[F]|_a - C_{\mathsf{r}}[F]|_d = \mathsf{Occ}(F, \Pi^s_{\mathsf{r}}\left[\!\left[(I^s|_o \setminus D^s) \cup A^s\right]\!\right] \ominus \Pi^s_{\mathsf{r}}\left[\!\left[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\right]\!\right]) \quad \text{for each } F \in ((I^s|_o \setminus D^s) \cup A^s) \cap I^s|_o \cap \mathsf{O}^s \tag{A.11}$$

$$C_{\mathsf{r}}[F]|_a = \mathsf{Occ}(F, \Pi^s_{\mathsf{r}}\left[\!\left[(I^s|_o \setminus D^s) \cup A^s\right]\!\right]) \quad \text{for each } F \in (((I^s|_o \setminus D^s) \cup A^s) \setminus I^s|_o) \cap \mathsf{O}^s \tag{A.12}$$

We shall prove that properties (A.5), (A.7), and (A.8) hold for each s with $0 \leq s \leq S$, and that the other properties hold for each s with $1 \leq s \leq S$. Then, property (A.8) for $s = S$ and line 72 of Algorithm 4 imply that I is correctly updated from $I|_o$ to $I|_n$. Moreover, $C_{\mathsf{nr}}[F]|_o = \mathsf{Occ}(F, E|_o \oplus \Pi^s_{\mathsf{nr}}\left[\!\left[I|_o\right]\!\right])$ and properties (A.8), (A.1), (A.3) and (A.9)

for $1 \leq s \leq S$ jointly imply the following:

$$\begin{aligned} C_{\mathsf{nr}}[F]|_a &= C_{\mathsf{nr}}[F]|_o - (C_{\mathsf{nr}}[F]|_o - C_{\mathsf{nr}}[F]|_d) + (C_{\mathsf{nr}}[F]|_a - C_{\mathsf{nr}}[F]|_d) \\ &= \mathsf{Occ}(F, E|_n \oplus \Pi^s_{\mathsf{nr}}[\![I|_n]\!]) = C_{\mathsf{nr}}[F]|_n \\ &\quad \text{for each } F \in I^s|_n \cap I^s|_o \cap \mathsf{O}^s \end{aligned} \tag{A.13}$$

In addition, (A.8) and (A.10) jointly imply that $C_{\mathsf{nr}}[F]|_a = C_{\mathsf{nr}}[F]|_n$ holds for each $F \in (I^s|_n \setminus I^s|_o) \cap \mathsf{O}^s$. Therefore, we have $C_{\mathsf{nr}}[F]|_a = C_{\mathsf{nr}}[F]|_n$ for each $F \in I^s|_n \cap \mathsf{O}^s$, which means the nonrecursive counts are correctly updated after the execution of the algorithm. $C_{\mathsf{r}}[F]|_a = C_{\mathsf{r}}[F]|_n$ can be shown analogously using properties (A.8), (A.2), (A.4), (A.11), and (A.12).

We prove properties (A.1)–(A.12) by induction on s. The base case where $s = 0$ is trivial since all relevant sets in (A.5), (A.7), and (A.8) are empty. For the inductive step, we consider an arbitrary s with $1 \leq s \leq S$ such that (A.5), (A.7), and (A.8) hold for $s - 1$, and we show that properties (A.1)–(A.12) hold for s. The proof is lengthy so we break it into several claims. □

Claim 23. *Properties (A.1) and (A.2) hold.*

Proof. The way sets D and A are constructed ensures that $(D^s \setminus D^{s-1}) \cap \mathsf{O}^{<s} = \emptyset$ and $A^{s-1} \subseteq \mathsf{O}^{<s}$ hold, and so $A^{s-1} \setminus D^s = A^{s-1} \setminus D^{s-1}$ holds as well, which in turn implies $I^s|_o \cup (A^{s-1} \setminus D^s) = I^s|_o \cup (A^{s-1} \setminus D^{s-1})$. Moreover, the induction assumption $D^{s-1} \subseteq I^{s-1}|_o$ ensures $I^s|_o \cup (A^{s-1} \setminus D^{s-1}) = I^s|_o \cup A^{s-1}$. Therefore, we have $I^s|_o \cup (A^{s-1} \setminus D^s) = I^s|_o \cup A^{s-1}$, which together with the definition of rule application ensures the correctness of the two properties. □

Claim 24. *Property (A.3) holds.*

Proof. Consider an arbitrary $F \in I^s|_o \cap \mathsf{O}^s$, we have two cases here.

If $F \notin (E^- \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}[\![I^s|_o : D^s \setminus A^{s-1}, A^{s-1} \setminus D^s]\!]$ holds, then the right-hand side of the equation in (A.3) equals zero. Moreover, Π^s_{nr} contains only nonrecursive rules, and $A^{s-1} \setminus D^s = A^{s-1} \setminus D^{s-1}$ holds for the same reason as explained in the proof of claim 23; thus we have $F \notin (E^- \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}[\![I^s|_o : D^{s-1} \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}]\!]$; but then, line 75 of Algorithm 4 ensures that the nonrecursive count of F is not decremented during the

execution of the OVERDELETE procedure, so the left-hand side of the equation is equal to zero as well. Therefore the property holds in this case.

If $F \in (E^- \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}\llbracket I^s|_o \colon D^s \setminus A^{s-1}, A^{s-1} \setminus D^s\rrbracket$ holds, then we clearly have $F \in (E^- \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}\llbracket I^s|_o \colon D^{s-1} \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\rrbracket$ since the application of nonrecursive rules will only match body atoms to facts from lower strata. Line 75 and line 76 guarantee that each distinct occurrence of F in $(E^- \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}\llbracket I^s|_o \colon D^{s-1} \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\rrbracket$ results in decrementing the nonrecursive count of F by one. Thus the property also holds in this case. □

Claim 25. *Property (A.4) holds.*

Proof. Line 80 and line 84 ensure that Δ_D used in line 82 is different between two iterations of the loop in lines 79-84, so the rule instances considered in line 82 are different between iterations. The total number of these rule instances is finite and is bounded by the number of rule instances in $\bigcup_{r' \in \Pi^s_{\mathsf{r}}} \mathsf{inst}_{r'}\left[I^s|_o\right]$. Thus the loop must terminate, and we let T be the total number of iterations. Moreover, for each $1 \leq i \leq T$, let D^s_i be the value of D at the beginning of the ith iteration of the loop, and let $C_{\mathsf{r}}[F]|^i_d$ be the value of $C_{\mathsf{r}}[F]$ for each $F \in I^s|_o \cap \mathsf{O}^s$ at the same time point. We prove by induction on i that (A.14) holds for $1 \leq i \leq T$. Then (A.14) for $i = T$ and $A^{s-1} \setminus D^{s-1} = A^{s-1} \setminus D^s$ ensure the correctness of property (A.4).

$$C_{\mathsf{r}}[F]|_o - C_{\mathsf{r}}[F]|^i_d = \mathsf{Occ}(F, \Pi^s_{\mathsf{r}}\llbracket I^s|_o \colon D^s_i \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\rrbracket) \quad \text{for each } F \in I^s|_o \cap \mathsf{O}^s \tag{A.14}$$

For the base case, consider an arbitrary $F \in I^s|_o \cap \mathsf{O}^s$. It is easy to see that the recursive count of F has never been changed and should be equal to $C_{\mathsf{r}}[F]|_o$ before line 77 of procedure OVERDELETE for stratum s. But then, line 77 and line 78 ensure that if $F \in \Pi^s_{\mathsf{r}}\llbracket I^s|_o \colon D^{s-1} \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\rrbracket$, then each distinct occurrence of F in the multiset results in decrementing the corresponding recursive count by one; moreover, if $F \notin \Pi^s_{\mathsf{r}}\llbracket I^s|_o \colon D^{s-1} \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\rrbracket$, then the corresponding recursive count will not be changed; either way, $D^s_1 = D^{s-1}$ implies that $C_{\mathsf{r}}[F]|_o - C_{\mathsf{r}}[F]|^1_d$ is equal to $\mathsf{Occ}(F, \Pi^s_{\mathsf{r}}\llbracket I^s|_o \colon D^s_1 \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\rrbracket)$, as required.

For the inductive step, assume that (A.14) holds for $i-1$ where $1 < i \leq T$, and consider arbitrary $F \in I^s|_o \cap \mathsf{O}^s$. Lines 80, 82 and 84 jointly imply (A.15).

$$C_{\mathsf{r}}[F]|_d^{i-1} - C_{\mathsf{r}}[F]|_d^i = \mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket I^s|_o \setminus (D_{i-1}^s \setminus A^{s-1}), I^s|_o \cup A^{s-1} \colon D_i^s \setminus D_{i-1}^s \rrbracket) \quad \text{(A.15)}$$

We now show that the following holds:

$$\begin{aligned} &\mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket I^s|_o \colon D_{i-1}^s \setminus A^{s-1}, A^{s-1} \setminus D^{s-1} \rrbracket) \\ &+ \mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket I^s|_o \setminus (D_{i-1}^s \setminus A^{s-1}), I^s|_o \cup A^{s-1} \colon D_i^s \setminus D_{i-1}^s \rrbracket) \\ &= \mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket I^s|_o \colon D_i^s \setminus A^{s-1}, A^{s-1} \setminus D^{s-1} \rrbracket) \end{aligned} \quad \text{(A.16)}$$

There is no rule instance repetition between $\bigcup_{r' \in \Pi_{\mathsf{r}}^s} \mathsf{inst}_{r'}\left[I^s|_o \colon D_{i-1}^s \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\right]$ and $\bigcup_{r' \in \Pi_{\mathsf{r}}^s} \mathsf{inst}_{r'}\left[I^s|_o \setminus (D_{i-1}^s \setminus A^{s-1}), I^s|_o \cup A^{s-1} \colon D_i^s \setminus D_{i-1}^s\right]$, so it is sufficient to show that (A.17) holds.

$$\begin{aligned} &\bigcup_{r' \in \Pi_{\mathsf{r}}^s} \mathsf{inst}_{r'}\left[I^s|_o \colon D_{i-1}^s \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\right] \\ &\cup \bigcup_{r' \in \Pi_{\mathsf{r}}^s} \mathsf{inst}_{r'}\left[I^s|_o \setminus (D_{i-1}^s \setminus A^{s-1}), I^s|_o \cup A^{s-1} \colon D_i^s \setminus D_{i-1}^s\right] \\ &= \bigcup_{r' \in \Pi_{\mathsf{r}}^s} \mathsf{inst}_{r'}\left[I^s|_o \colon D_i^s \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\right] \end{aligned} \quad \text{(A.17)}$$

The $\subseteq$ direction of (A.17) trivially holds. Now consider the $\supseteq$ direction, let r'' be an arbitrary rule instance contained in the right-hand side of (A.17), then there exists rule $r' \in \Pi_{\mathsf{r}}^s$ such that $r'' \in \mathsf{inst}_{r'}\left[I^s|_o \colon D_i^s \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\right]$ holds. If we have $r'' \in \mathsf{inst}_{r'}\left[I^s|_o \colon D_{i-1}^s \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\right]$, then clearly r'' is also contained in the left-hand side of (A.17). Otherwise, r'' has all positive atoms in $I^s|_o$ but no positive atom in $D_{i-1}^s \setminus A^{s-1}$; moreover, all negative body atoms of r'' have to be matched in $I^s|_o \cup (A^{s-1} \setminus D^{s-1}) = I^s|_o \cup A^{s-1}$; furthermore, r'' has at least one positive body atom in $D_i^s \setminus D_{i-1}^s$; therefore, we have $r'' \in \mathsf{inst}_{r'}\left[I^s|_o \setminus (D_{i-1}^s \setminus A^{s-1}), I^s|_o \cup A^{s-1} \colon D_i^s \setminus D_{i-1}^s\right]$, so r'' is contained in the left-hand side of (A.17) in this case as well.

(A.15), (A.16) and the induction assumption that (A.14) holds for $i-1$ imply the correctness of (A.14) for i, and this completes our proof. □

Claim 26. *The right-hand inclusion of property (A.5) holds.*

Proof. For each rule $r \in \Pi^s$, procedure OVERDELETE for stratum index s considers in lines 75, 77, and 82 only instances of r that are contained in $\mathsf{inst}_r\left[I^s|_o\right]$, so the facts derived by these rule instances are in $I^s|_o$. Thus, the claim holds by a straightforward induction on the construction of the set D^s. □

Claim 27. *The left-hand inclusion of property (A.5) holds.*

Proof. We show by induction that (A.18) holds for each i.

$$I_i^s|_o \setminus I^s|_n \subseteq D^s \tag{A.18}$$

For the base case, note that $I_0^s|_o = I^{s-1}|_o \cup (E|_o \cap \mathsf{O}^s)$ and that $I^{s-1}|_o \setminus I^{s-1}|_n \subseteq D^{s-1}$ holds for $s-1$ by the induction assumption. Now consider an arbitrary fact $F \in E|_o \cap \mathsf{O}^s$ such that $F \notin I^s|_n$ holds. Then, the latter ensures $F \notin E|_o \setminus E^-$, which implies $F \in E^-$ and so F is added to N_D in line 76. We next show that F is added to D in line 84. $F \notin I^s|_n$ and $I^s|_n = I^s|_n \cup \Pi^s\Big[I^s|_n\Big]$ imply $\mathsf{Occ}(F, \Pi^s_{\mathsf{nr}}[\![I^s|_n]\!]) = 0$, which in turn implies $\mathsf{Occ}(F, \Pi^s_{\mathsf{nr}}[\![I^{s-1}|_n]\!]) = 0$; but then, $I^{s-1}|_o \setminus (D^{s-1} \setminus A^{s-1}) \subseteq (I^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1}$ and the induction assumption $(I^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1} = I^{s-1}|_n$ ensure the following property.

$$\mathsf{Occ}(F, \Pi^s_{\mathsf{nr}}[\![I^{s-1}|_o \setminus (D^{s-1} \setminus A^{s-1})), I^{s-1}|_o \cup A^{s-1}]\!] = 0$$

Together with (A.1), (A.3), and the definition of $C_{\mathsf{nr}}[F]|_o$ this implies $C_{\mathsf{nr}}[F]|_d = 0$, so the condition in line 80 is satisfied. Hence, $F \in D^s$ holds, as required.

For the inductive step, assume that $I^s_{i-1}|_o$ satisfies (A.18) for $i > 0$, and consider arbitrary $F \in I^s_i|_o \setminus I^s|_n$. If $F \in I^s_{i-1}|_o$, then $F \in D^s$ holds by the induction assumption. Otherwise, there exist a rule $r \in \Pi^s$ and its instance $r' \in \mathsf{inst}_r\Big[I^s_{i-1}|_o\Big]$ such that $F \in \mathsf{h}(r')$. Definition (3.3) ensures $\mathsf{b}^+(r') \subseteq I^s_{i-1}|_o \subseteq I^s|_o$ and $\mathsf{b}^-(r') \cap I^s_{i-1}|_o = \emptyset$, and $\mathsf{b}^-(r') \subseteq \mathsf{O}^{<s}$ implies $\mathsf{b}^-(r') \cap I^{s-1}|_o = \mathsf{b}^-(r') \cap I^s|_o = \emptyset$. Finally, $F \notin I^s|_n$ implies $r' \notin \mathsf{inst}_r\Big[I^s|_n\Big]$, so by definition (3.3) we have one of the following two possibilities.

- $\mathsf{b}^+(r') \not\subseteq I^s|_n$. There exists $G \in \mathsf{b}^+(r') \cap (I^s_{i-1}|_o \setminus I^s|_n)$. The induction assumption for (A.18) implies $G \in D^s$, and $G \notin I^s|_n$ implies $G \notin I^{s-1}|_n$, so the induction assumption for (A.8) ensures $G \notin A^{s-1}$; hence, $G \in D^s \setminus A^{s-1}$.
- $\mathsf{b}^-(r') \cap I^s|_n = \mathsf{b}^-(r') \cap I^{s-1}|_n \neq \emptyset$. There exists $G \in \mathsf{b}^-(r') \cap (I^{s-1}|_n \setminus I^{s-1}|_o)$; but then, the right-hand inclusion of (A.5) implies $G \notin D^{s-1}$, and the induction assumption for (A.8) implies $G \in A^{s-1}$; therefore, we have $G \in A^{s-1} \setminus D^{s-1} = A^{s-1} \setminus D^s$.

Either way, the right-hand side of (A.3) or (A.4) is larger than zero. Hence, the count for F is decremented in OVERDELETE and F is added to N_D. Then, in the same way as in the proof of the base case we have $F \in D^s$, as required. □

Claim 28. *The right-hand inclusion of property* (A.6) *holds.*

Proof. Consider arbitrary $F \in R^s$. F can only be added to R^s in line 70, so we have $F \in D^s \cap \mathsf{O}^s$ and $C_\mathsf{r}[F]|_d > 0$; $F \in D^s \cap \mathsf{O}^s$ and property (A.5) ensure $F \in I^s|_o \cap \mathsf{O}^s$; but then, properties (A.4), (A.2) and $C_\mathsf{r}[F]|_o = \mathsf{Occ}(F, \Pi^s_\mathsf{r}[\![I^s|_o]\!])$ jointly imply that $C_\mathsf{r}[F]|_d = \mathsf{Occ}(F, \Pi^s_\mathsf{r}[\![I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}]\!])$ holds, and so $C_\mathsf{r}[F]|_d > 0$ implies that $\mathsf{Occ}(F, \Pi^s_\mathsf{r}[\![I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}]\!]) > 0$ holds.

Next we show that $\Pi^s_\mathsf{r}[\![I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}]\!] \subseteq \Pi^s_\mathsf{r}[\![I^s|_n]\!]$ holds: due to stratification negative body atoms will only be matched against facts from lower strata, so we have $\Pi^s_\mathsf{r}[\![I^s|_n]\!] = \Pi^s_\mathsf{r}[\![I^s|_n, I^{s-1}|_n]\!]$; moreover, the left-hand inclusion of property (A.5) implies $I^s|_o \setminus D^s \subseteq I^s|_n$, which together with the induction assumption for property (A.8) implies $I^s|_o \setminus (D^s \setminus A^{s-1}) \subseteq (I^s|_o \setminus D^s) \cup A^{s-1} \subseteq I^s|_n$; furthermore, the induction assumption for property (A.8) ensures $I^{s-1}|_n = (I^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1} \subseteq I^s|_o \cup A^{s-1}$; therefore we clearly have $\Pi^s_\mathsf{r}[\![I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}]\!] \subseteq \Pi^s_\mathsf{r}[\![I^s|_n]\!]$. But then, $\mathsf{Occ}(F, \Pi^s_\mathsf{r}[\![I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}]\!]) > 0$ implies $\mathsf{Occ}(F, \Pi^s_\mathsf{r}[\![I^s|_n]\!]) > 0$, so we have $F \in I^s|_n$, as required. □

Claim 29. *The left-hand inclusion of property* (A.6) *holds.*

Proof. Consider arbitrary $F \in \mathsf{O}^s \cap D^s \cap \Pi^s_\mathsf{r}\big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\big]$, and we show that $F \in R^s$ holds. $F \in \mathsf{O}^s \cap D^s$ implies that $\mathsf{Occ}(F, \Pi^s_\mathsf{r}[\![I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}]\!])$ is equal to $C_\mathsf{r}[F]|_d$ in the same way as in the proof of claim 28; but then, the fact that $F \in \Pi^s_\mathsf{r}\big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\big]$ holds implies $C_\mathsf{r}[F]|_d > 0$, so line 70 of Algorithm 4 ensures that F is added to R^s, as required. □

Claim 30. *Property* (A.7) *holds.*

Proof. Consider an arbitrary fact $F \in I^s|_o \cap A^s$: if $F \in I^s|_o \cap A^{s-1}$ holds, then the induction assumption ensures that we have $F \in D^{s-1} \subseteq D^s$; if $F \in I^s|_o \cap (A^s \setminus A^{s-1})$ holds, then F is added to Δ_A in line 92 of procedure INSERT for stratum index s, which ensures $F \notin (I^s|_o \setminus D^s) \cup A^{s-1}$; but then, $F \in I^s|_o$ implies $F \in D^s$, as required. □

Claim 31. *The $\subseteq$ direction of property* (A.8) *holds.*

Proof. We prove by induction on the construction of A that $(I^s|_o \setminus D^s) \cup A \subseteq I^s|_n$ holds. We first consider the base case. Set A is equal to A^{s-1} before the loop in lines 91–96; thus, property (A.8) is equivalent to $(I^s|_o \setminus D^s) \cup A^{s-1} \subseteq I^s|_n$; now $I^s|_o \setminus D^s \subseteq I^s|_n$ is implied by the left-hand inclusion of property (A.5), whereas $A^{s-1} \subseteq I^s|_n$ holds by the induction assumption.

For the inductive step, we assume that $(I^s|_o \setminus D^s) \cup A \subseteq I^s|_n$ holds, and we consider ways in which Algorithm 4 can add a fact F to A. If $F \in E^+ \cap \mathsf{O}^s$, then $F \in I^s|_n$ clearly holds. Moreover, if $F \in R^s$ holds, then $F \in I^s|_n$ holds by (A.6). Otherwise, F is derived in line 87, 89, or 95, so a rule $r \in \Pi^s$ and its instance $r' = \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A\Big]$ exist such that $F \in \mathsf{h}(r')$ holds. But then, definition (3.3) ensures $\mathsf{b}^+(r') \subseteq (I^s|_o \setminus D^s) \cup A \subseteq I^s|_n$ and $\mathsf{b}^-(r') \cap ((I^s|_o \setminus D^s) \cup A) = \emptyset$, which together with $\mathsf{b}^-(r') \subseteq \mathsf{O}^{<s}$ and the induction assumption for (A.8) implies $\mathsf{b}^-(r') \cap I^s|_n = \mathsf{b}^-(r') \cap I^{s-1}|_n = \emptyset$. Consequently, we have $r' \in \mathsf{inst}_r\Big[I^s|_n\Big]$, so $F \in I^s|_n$ holds, as required. □

Claim 32. *The $\supseteq$ direction of property* (A.8) *holds.*

Proof. We show by induction that (A.19) holds for each i.

$$(I^s|_o \setminus D^s) \cup A^s \supseteq I^s_i|_n \tag{A.19}$$

For the base case, we have $I^s_0|_n = I^{s-1}|_n \cup (E|_n \cap \mathsf{O}^s) = (I^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1} \cup (E|_n \cap \mathsf{O}^s)$ by the induction assumption for (A.8). $I^{s-1}|_o \setminus D^{s-1} \subseteq I^s|_o \setminus D^s$ and $A^{s-1} \subseteq A^s$ clearly hold. Now consider arbitrary $F \in E|_n \cap \mathsf{O}^s$. If $F \in E^+$, then lines 87, 88, 92, and 94 ensure $F \in (I^s|_o \setminus D^s) \cup A^s$. If $F \in E|_n \setminus E^+ = E|_o \setminus E^-$, then we clearly have $F \in I^s|_o$; but then, $C_{\mathsf{nr}}[F]|_o = \mathsf{Occ}(F, E|_o \oplus \Pi^s_{\mathsf{nr}}[\![I|_o]\!])$ and property (A.3) imply $C_{\mathsf{nr}}[F]|_d \geq 1$; thus, line 80 ensures $F \notin D^s$.

For the inductive step, assume that $I^s_{i-1}|_n$ satisfies (A.19) for $i > 0$, and consider arbitrary $F \in I^s_i|_n$. If $F \in I^s_{i-1}|_n$, then (A.19) holds by the induction assumption. Otherwise, by definition a rule r and its instance $r' \in \mathsf{inst}_r\Big[I^s_{i-1}|_n\Big]$ exist where $\mathsf{h}(r') = F$. Definition (3.3) and (A.19) for $i-1$ ensure $\mathsf{b}^+(r') \subseteq I^s_{i-1}|_n \subseteq (I^s|_o \setminus D^s) \cup A^s$. Moreover, Definition (3.3) also ensures $\mathsf{b}^-(r') \cap I^s_{i-1}|_n = \emptyset$, which together with the induction assumption that property (A.8) holds for $s-1$ and the definition of $I^s_{i-1}|_n$ implies $\mathsf{b}^-(r') \cap ((I^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1}) = \emptyset$; but then, $\mathsf{b}^-(r') \cap ((I^s|_o \setminus D^s) \cup A^s) = \emptyset$ clearly holds since we have $\mathsf{b}^-(r') \subseteq \mathsf{O}^{<s}$. Now we consider the following cases.

- $\mathsf{b}^+(r') \cap (A^s \setminus A^{s-1}) \neq \emptyset$. Facts in $A^s \setminus A^{s-1}$ are added to A via lines 92 and 94, so there is a point in the execution of the algorithm where $\mathsf{b}^+(r') \cap (A^s \setminus A^{s-1}) \cap \Delta_A \neq \emptyset$ holds in line 95 for the last time for Δ_A. Since $\Delta_A \subseteq A$ holds at this point, we clearly have $\mathsf{b}^+(r') \subseteq (I^s|_o \setminus D^s) \cup A$; moreover, $A \subseteq A^s$ and $\mathsf{b}^-(r') \cap ((I^s|_o \setminus D^s) \cup A^s) = \emptyset$ ensure $\mathsf{b}^-(r') \cap ((I^s|_o \setminus D^s) \cup A) = \emptyset$. But then, $r' \in \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A \,\vdots\, \Delta_A\Big]$ holds; $F = \mathsf{h}(r')$ will be added to N_A in line 96; and lines 92 and 94 ensure $F \in (I^s|_o \setminus D^s) \cup A^s$.
- $\mathsf{b}^+(r') \cap (A^s \setminus A^{s-1}) = \emptyset$, so $\mathsf{b}^+(r') \subseteq (I^s|_o \setminus D^s) \cup A^{s-1}$ holds. We have the following two possibilities.
  - $\mathsf{b}^+(r') \cap (A^{s-1} \setminus D^s)$ or $\mathsf{b}^-(r') \cap (D^s \setminus A^{s-1})$. By the definition of rule matching, we clearly have $r' \in \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^{s-1} \,\vdots\, A^{s-1} \setminus D^s, D^s \setminus A^{s-1}\Big]$. But then, lines 87–90 ensure that $F = \mathsf{h}(r')$ is added to N_A; and lines 92 and 94 ensure $F \in (I^s|_o \setminus D^s) \cup A^s$.
  - $\mathsf{b}^+(r') \cap (A^{s-1} \setminus D^s) = \mathsf{b}^-(r') \cap (D^s \setminus A^{s-1}) = \emptyset$. Then, $\mathsf{b}^-(r') \cap (D^s \setminus A^{s-1}) = \emptyset$ and $\mathsf{b}^-(r') \cap (I^s|_o \setminus D^s) \cap A^s = \emptyset$ imply $\mathsf{b}^-(r') \cap (I^s|_o \cup A^{s-1}) = \emptyset$. Moreover, we argue that each fact G with $G \in \mathsf{b}^+(r') \subseteq (I^s|_o \setminus D^s) \cup A^{s-1}$ satisfies $G \in I^s|_o \setminus (D^s \setminus A^{s-1})$. This clearly holds if $G \in I^s|_o \setminus D^s$. If $G \in A^{s-1}$, then $G \notin A^{s-1} \setminus D^s$ implies $G \in D^s$, which in turn implies $G \in I^s|_o$ by claim 26; thus, $G \in I^s|_o \setminus (D^s \setminus A^{s-1})$ holds. Now, definition (3.3) ensures $r' \in \mathsf{inst}_r\Big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\Big]$, which implies $F = \mathsf{h}(r') \in I^s|_o$. If $F \notin D^s$, then $F \in (I^s|_o \setminus D^s) \cup A^s$ trivially holds. If $F \in D^s$, property (A.6) implies $F \in R^s$; then, lines 86, 92, and 94 ensure $F \in (I^s|_o \setminus D^s) \cup A^s$. □

Claim 33. *The following two properties hold.*

$$\begin{aligned}\Pi^s_{\mathsf{nr}}\big[\!\big[(I^s|_o \setminus D^s) \cup A^s\big]\!\big] &= \Pi^s_{\mathsf{nr}}\big[\!\big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\big]\!\big] \\ &\quad \oplus \Pi^s_{\mathsf{nr}}\big[\!\big[(I^s|_o \setminus D^s) \cup A^s : A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}\big]\!\big]\end{aligned} \tag{A.20}$$

$$\begin{aligned}\Pi^s_{\mathsf{r}}\big[\!\big[(I^s|_o \setminus D^s) \cup A^s\big]\!\big] &= \Pi^s_{\mathsf{r}}\big[\!\big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\big]\!\big] \\ &\quad \oplus \Pi^s_{\mathsf{r}}\big[\!\big[(I^s|_o \setminus D^s) \cup A^s : A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}\big]\!\big]\end{aligned} \tag{A.21}$$

Proof. By the definition of $\Pi\big[\!\big[I^{\mathsf{p}}, I^{\mathsf{n}} : P, N\big]\!\big]$ it is sufficient to show that for each rule $r \in \Pi^s$, properties (A.22) and (A.23) hold.

$$\begin{aligned}&\mathsf{inst}_r\Big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\Big] \cap \\ &\mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s : A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}\Big] = \emptyset\end{aligned} \tag{A.22}$$

$$\begin{aligned}\mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s\Big] &= \mathsf{inst}_r\Big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\Big] \\ &\quad \cup\ \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s : A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}\Big]\end{aligned} \tag{A.23}$$

To prove (A.22), consider arbitrary $r' \in \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s : A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}\Big]$. By Definition (3.3) we have $\mathsf{b}^+(r') \cap (A^s \setminus D^{s-1}) \neq \emptyset$ or $\mathsf{b}^-(r') \cap (D^{s-1} \setminus A^{s-1}) \neq \emptyset$; now we examine these two cases separately.

For the first case, let F be an arbitrary fact in $\mathsf{b}^+(r') \cap (A^s \setminus D^{s-1})$. Now if $F \in A^{s-1} \setminus D^{s-1}$ holds, then the induction assumption for (A.7) ensures $F \notin I^{s-1}|_o$, which in turn implies $F \notin I^s|_o \setminus (D^s \setminus A^{s-1})$; if $F \in A^s \setminus A^{s-1}$ holds, then in the same way as in the proof of claim 30 we have $F \notin (I^s|_o \setminus D^s) \cup A^{s-1}$, which implies $F \notin I^s|_o \setminus (D^s \setminus A^{s-1})$; either way, we have $\mathsf{b}^+(r') \not\subseteq I^s|_o \setminus (D^s \setminus A^{s-1})$, so $r' \notin \mathsf{inst}_r\Big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\Big]$ holds.

For the second case, $\mathsf{b}^-(r') \cap (D^{s-1} \setminus A^{s-1}) \neq \emptyset$ and the induction assumption for property (A.5) imply $\mathsf{b}^-(r') \cap I^{s-1}|_o \neq \emptyset$, which clearly implies $\mathsf{b}^-(r') \cap (I^s|_o \cup A^{s-1}) \neq \emptyset$; thus, by Definition (3.3) we have $r' \notin \mathsf{inst}_r\Big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\Big]$; this completes our proof for (A.22).

Next we prove the $\supseteq$ direction of property (A.23). Consider an arbitrary rule instance r' contained in the right-hand side of (A.23).

- If $r' \in \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s : A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}\Big]$ holds, then by definition (3.3) we clearly have $r' \in \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s\Big]$.

- If $r' \in \mathsf{inst}_r\Big[I^s|_o \backslash (D^s \backslash A^{s-1}), I^s|_o \cup A^{s-1}\Big]$ holds, then $\mathsf{b}^+(r') \subseteq I^s|_o \setminus (D^s \setminus A^{s-1})$ implies $\mathsf{b}^+(r') \subseteq (I^s|_o \setminus D^s) \cup A^s$; moreover, $\mathsf{b}^-(r') \cap (I^s|_o \cup A^{s-1}) = \emptyset$ and $\mathsf{b}^-(r') \subseteq \mathsf{O}^{<s}$ jointly imply $\mathsf{b}^-(r') \cap ((I^s|_o \setminus D^s) \cup A^s) = \mathsf{b}^-(r') \cap ((I^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1}) = \emptyset$; therefore, $r' \in \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s\Big]$ holds by Definition (3.3).

Finally, for the $\subseteq$ direction, consider arbitrary $r' \in \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s\Big]$. If $\mathsf{b}^+(r') \cap (A^s \setminus D^{s-1}) \neq \emptyset$ or $\mathsf{b}^-(r') \cap (D^{s-1} \setminus A^{s-1}) \neq \emptyset$ holds, then we clearly have $r' \in \mathsf{inst}_r\Big[(I^s|_o \setminus D^s) \cup A^s : A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}\Big]$ by Definition (3.3). Otherwise, let F be an arbitrary fact in $\mathsf{b}^+(r')$ and let G be an arbitrary fact in $\mathsf{b}^-(r')$, then we have $F \in ((I^s|_o \setminus D^s) \cup A^s) \setminus (A^s \setminus D^{s-1})$ and $G \notin (I^s|_o \setminus D^s) \cup A^s \cup (D^{s-1} \setminus A^{s-1})$. We next show that $F \in I^s|_o \setminus (D^s \setminus A^{s-1})$ and $G \notin I^s|_o \cup A^{s-1}$ hold.

If $F \in (I^s|_o \setminus D^s) \setminus (A^s \setminus D^{s-1})$ holds, then $F \in I^s|_o \setminus (D^s \setminus A^{s-1})$ trivially holds; if $F \in A^s \setminus (A^s \setminus D^{s-1})$ holds, then we have $F \in A^s \cap D^{s-1} = A^{s-1} \cap D^{s-1} = A^{s-1} \cap D^s$, which in turn implies $F \notin D^s \setminus A^{s-1}$; moreover, $F \in D^s$ and property (A.5) imply $F \in I^s|_o$; therefore, $F \in I^s|_o \setminus (D^s \setminus A^{s-1})$ holds, as required.

$G \notin A^s$ and $G \notin D^{s-1} \setminus A^{s-1}$ jointly imply $G \notin D^{s-1}$; but then, $G \in \mathsf{O}^{<s}$ implies $G \notin D^s$, which together with $G \notin I^s|_o \setminus D^s$ ensures $G \notin I^s|_o$; thus, $G \notin I^s|_o \cup A^{s-1}$ holds, as required.

F and G are chosen arbitrarily, so $\mathsf{b}^+(r') \in I^s|_o \setminus (D^s \setminus A^{s-1})$ and $\mathsf{b}^-(r') \notin I^s|_o \cup A^{s-1}$ hold; by Definition (3.3) we have $r' \in \mathsf{inst}_r\Big[I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}\Big]$. This completes our proof for (A.23). □

Claim 34. *Property* (A.9) *holds.*

Proof. Due to claim 33 it is sufficient to show that the following property holds for each $F \in ((I^s|_o \setminus D^s) \cup A^s) \cap I^s|_o \cap \mathsf{O}^s$.

$$C_{\mathsf{nr}}[F]|_a - C_{\mathsf{nr}}[F]|_d = \mathsf{Occ}(F, (E^+ \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}[\![(I^s|_o \setminus D^s) \cup A^s : A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}]\!]) \tag{A.24}$$

Π^s_{nr} contains only nonrecursive rules, so it is sufficient to prove

$$C_{\mathsf{nr}}[F]|_a - C_{\mathsf{nr}}[F]|_d = \mathsf{Occ}(F, (E^+ \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}[\![(I^s|_o \setminus D^s) \cup A^{s-1} : A^{s-1} \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}]\!]), \tag{A.25}$$

which is ensured by line 87 and line 88 of the algorithm. □

Claim 35. *Property* (A.10) *holds.*

Proof. $F \notin I^s|_o \cap \mathsf{O}^s$ ensures $C_{\mathsf{nr}}[F]|_d = C_{\mathsf{nr}}[F]|_o = 0$. Moreover, $F \notin \Pi^s_{\mathsf{nr}}[\![I^s|_o]\!]$ and property (A.1) jointly imply $F \notin \Pi^s_{\mathsf{nr}}[\![I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}]\!]$. Now if $F \notin (E^+ \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}[\![(I^s|_o \setminus D^s) \cup A^{s-1} \colon A^{s-1} \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}]\!]$, then by (A.20) and $F \notin (E|_o \setminus E^-) \cap \mathsf{O}^s$ we clearly have $\mathsf{Occ}(F, ((E|_o \setminus E^-) \cup E^+) \oplus \Pi^s_{\mathsf{nr}}[\![(I^s|_o \setminus D^s) \cup A^s]\!]) = 0$; moreover, line 87 ensures that the nonrecursive count for F is not incremented in this case, so the left-hand side of (A.10) equals zero as well. Otherwise, each occurrence of F in the multiset $(E^+ \cap \mathsf{O}^s) \oplus \Pi^s_{\mathsf{nr}}[\![(I^s|_o \setminus D^s) \cup A^{s-1} \colon A^{s-1} \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}]\!]$ results in incrementing the nonrecursive count of F by one; together with $F \notin (E|_o \setminus E^-) \cap \mathsf{O}^s$ and $F \notin \Pi^s_{\mathsf{nr}}[\![I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1}]\!]$ this ensures the correctness of the property. □

Claim 36. *Property* (A.11) *holds.*

Proof. Line 92 and line 94 ensure that Δ_A used in line 95 is different between iterations of the loop in lines 91–96, so the rle instances considered in line 95 are different between iterations. All these rule instances are in $\bigcup_{r' \in \Pi^s_{\mathsf{r}}} \mathsf{inst}_{r'}\left[(I^s|_o \setminus D^s) \cup A^s\right]$, which is equal to $\bigcup_{r' \in \Pi^s_{\mathsf{r}}} \mathsf{inst}_{r'}\left[I^s|_n\right]$ by property (A.8). Therefore, the loop will terminate. Now let T be the total number of iterations; moreover, for each $1 \leq i \leq T$, let A^s_i be the value of A at the beginning of the ith iteration of the loop, and let $C_{\mathsf{r}}[F]|^i_a$ be the value of $C_{\mathsf{r}}[F]$ for each $F \in ((I^s|_o \setminus D^s) \cup A^s) \cap I^s|_o \cap \mathsf{O}^s$ at the same time point. We next prove by induction on i that (A.26) holds for $1 \leq i \leq T$; then (A.26) for $i = T$ and property (A.21) ensure the correctness of the claim.

$$C_{\mathsf{r}}[F]|^i_a - C_{\mathsf{r}}[F]|_d = \mathsf{Occ}(F, \Pi^s_{\mathsf{r}}[\![(I^s|_o \setminus D^s) \cup A^s_i \colon A^s_i \setminus D^{s-1}, D^{s-1} \setminus A^s_i]\!]) \tag{A.26}$$

For the base case, we have $A^s_1 = A^{s-1}$ by the definition of A^s_i. Consider arbitrary $F \in ((I^s|_o \setminus D^s) \cup A^s) \cap I^s|_o \cap \mathsf{O}^s$. Lines 89 and 90 ensure that each occurrence of F in $\Pi^s_{\mathsf{r}}[\![(I^s|_o \setminus D^s) \cup A^{s-1} \colon A^{s-1} \setminus D^s, D^s \setminus A^{s-1}]\!]$ results in incrementing the corresponding recursive count by one. Moreover, facts in A^{s-1} belong to $\mathsf{O}^{\leq s-1}$ whereas facts in $D^s \setminus D^{s-1}$ belong to O^s, so we have $A^{s-1} \setminus D^s = A^{s-1} \setminus D^{s-1}$. In addition, negative body atoms will only be matched to facts from lower strata, i.e., facts in $D^{s-1} \setminus A^{s-1}$. Consequently, (A.26) holds for $i = 1$, as required.

For the inductive step, assume that (A.26) holds for $i-1$ where $1 < i \leq T$, and consider arbitrary $F \in ((I^s|_o \setminus D^s) \cup A^s) \cap I^s|_o \cap \mathsf{O}^s$. Lines 92, 94, and 95 jointly imply (A.27).

$$C_{\mathsf{r}}[F]|_a^i - C_{\mathsf{r}}[F]|_a^{i-1} = \mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket (I^s|_o \setminus D^s) \cup A_i^s \colon A_i^s \setminus A_{i-1}^s \rrbracket) \tag{A.27}$$

We now show that the following holds.

$$\begin{aligned} &\mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket (I^s|_o \setminus D^s) \cup A_{i-1}^s \colon A_{i-1}^s \setminus D^{s-1}, D^{s-1} \setminus A_{i-1}^s \rrbracket \\ &+ \mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket (I^s|_o \setminus D^s) \cup A_i^s \colon A_i^s \setminus A_{i-1}^s \rrbracket) \\ &= \mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket (I^s|_o \setminus D^s) \cup A_i^s \colon A_i^s \setminus D^{s-1}, D^{s-1} \setminus A_i^s \rrbracket) \end{aligned} \tag{A.28}$$

To this end, it is sufficient to show that for each $r \in \Pi_{\mathsf{r}}^s$, property (A.29) holds.

$$\begin{aligned} &\mathsf{inst}_r \Big[(I^s|_o \setminus D^s) \cup A_{i-1}^s \colon A_{i-1}^s \setminus D^{s-1}, D^{s-1} \setminus A_{i-1}^s \Big] \\ &\cup \mathsf{inst}_r \Big[(I^s|_o \setminus D^s) \cup A_i^s \colon A_i^s \setminus A_{i-1}^s \Big] \\ &= \mathsf{inst}_r \Big[(I^s|_o \setminus D^s) \cup A_i^s \colon A_i^s \setminus D^{s-1}, D^{s-1} \setminus A_i^s \Big] \end{aligned} \tag{A.29}$$

$(A_{i-1}^s \setminus D^{s-1}) \cup (A_i^s \setminus A_{i-1}^s) = A_i^s \setminus D^{s-1}$ and $D^{s-1} \setminus A_{i-1}^s = D^{s-1} \setminus A_i^s \subseteq \mathsf{O}^{<s}$ ensure that the $\subseteq$ direction of (A.29) holds. To see that the $\supseteq$ direction holds as well, consider arbitrary $r' \in \mathsf{inst}_r \Big[(I^s|_o \setminus D^s) \cup A_i^s \colon A_i^s \setminus D^{s-1}, D^{s-1} \setminus A_i^s \Big]$. If $\mathsf{b}^+(r') \cap (A^s \setminus A^{s-1}) \neq \emptyset$, then we clearly have $r' \in \mathsf{inst}_r \Big[(I^s|_o \setminus D^s) \cup A_i^s \colon A_i^s \setminus A_{i-1}^s \Big]$. If $\mathsf{b}^+(r') \cap (A_i^s \setminus A_{i-1}^s) = \emptyset$, then definition (3.3) ensures that we have $\mathsf{b}^+(r') \subseteq ((I^s|_o \setminus D^s) \cup A_i^s) \setminus (A_i^s \setminus A_{i-1}^s) = (I^s|_o \setminus D^s) \cup A_{i-1}^s$. There are two possibilities here: if $\mathsf{b}^+(r') \cap (A_i^s \setminus D^{s-1}) \neq \emptyset$, then $\mathsf{b}^+(r') \cap (A_i^s \setminus A_{i-1}^s) = \emptyset$ implies $\mathsf{b}^+(r') \cap (A_{i-1}^s \setminus D^{s-1}) \neq \emptyset$; if $\mathsf{b}^-(r') \cap (D^{s-1} \setminus A_i^s) \neq \emptyset$, then clearly $\mathsf{b}^-(r') \cap (D^{s-1} \setminus A_{i-1}^s) \neq \emptyset$ holds as well; either way, we have $F \in \mathsf{inst}_r \Big[(I^s|_o \setminus D^s) \cup A_{i-1}^s \colon A_{i-1}^s \setminus D^{s-1}, D^{s-1} \setminus A_{i-1}^s \Big]$. Therefore, the $\supseteq$ direction of property (A.29) holds. But then, property (A.28) holds, which together with (A.27) and the induction assumption for (A.26) ensures that (A.26) holds for i as well. □

Claim 37. *Property* (A.12) *holds.*

Proof. $F \notin I^s|_o$ and the definition of compatibility jointly imply $C_{\mathsf{r}}[F]|_d = 0$, so we have $C_{\mathsf{r}}[F]|_a = \mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket (I^s|_o \setminus D^s) \cup A^s \colon A^s \setminus D^{s-1}, D^{s-1} \setminus A^{s-1} \rrbracket)$ in the same way as in the proof for the previous claim. Moreover, $\mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket I^s|_o \setminus (D^s \setminus A^{s-1}), I^s|_o \cup A^{s-1} \rrbracket) = 0$ holds since we have $F \notin I^s|_o$. Therefore, $C_{\mathsf{r}}[F]|_a = \mathsf{Occ}(F, \Pi_{\mathsf{r}}^s \llbracket (I^s|_o \setminus D^s) \cup A^s \rrbracket)$ holds due to property (A.21). □

A.2 Proof of Theorem 5

Theorem 5. *Let n, m, k, u, and w be defined as in Section 3.3. Then, Algorithm 4 updates the materialisation I in $O(m \cdot w^2 \cdot u^k \cdot n^k)$ time.*

Proof. First we revisit the definitions of n, m, k, u, and w. More specifically, let $n = |E| + |E^-| + |E^+|$ be the total number of facts in E, E^-, and E^+, and let $m = |\Pi|$ be the number of rules in Π; moreover, let k be the largest possible number of distinct variables in the body of one rule in Π, let u be the largest possible arity of a predicate that occurs in Π, and let w be the largest possible rule body size. We assume that all facts are indexed, and that it takes $O(1)$ time to check if a fact is in a particular set or to add a fact to a set. In addition, we assume that it takes $O(1)$ time to access (read or write) $C_{\mathsf{nr}}[F]$ or $C_{\mathsf{r}}[F]$ given a fact F. Consider the execution of Algorithm 4. It is straightforward to see that line 67 requires only $O(n)$ time. We next look into the cost of handling one stratum Π^s with $1 \leq s \leq S$. More specifically, we will examine the running time for each of the three phases, i.e., overdeletion, one-step rederivation, and insertion.

Overdeletion starts with line 74, which requires $O(1)$ time. Then, the algorithm identifies facts affected by changes in the previous stratum by evaluating $\Pi^s_{\mathsf{nr}}[\![I \dot{:} D \setminus A, A \setminus D]\!]$. We analyse the cost of handling one rule $r \in \Pi^s_{\mathsf{nr}}$ in line 75. The rule could have at most w positive and negative body atoms. Consider the ith body atom B of the rule, and let l be the number of distinct variables in B. Then, whether B is a positive or a negative body atom, during the execution of line 75, B could be used as the pivot atom in a maximum number of $u^l \cdot n^l$ queries; for each of these queries, there are at most $k - l$ variables left to be matched; hence, the running time for using B as the pivot atom is $O(w \cdot u^k \cdot n^k)$. There is a maximum number of w body atoms in r, and the number of rules considered in line 75 is $|\Pi^s_{\mathsf{nr}}|$, so the overall cost for rule evaluation in line 75 is $O(|\Pi^s_{\mathsf{nr}}| \cdot w^2 \cdot u^k \cdot n^k)$. The cost of iterating through the obtained multiset as well as $E \cap \mathsf{O}^s$ and updating N_A and C_{nr} require $O(n + |\Pi^s_{\mathsf{nr}}| \cdot u^k \cdot n^k)$ time, which can be safely discarded. Hence, the cost of lines 75–76 is $O(|\Pi^s_{\mathsf{nr}}| \cdot w^2 \cdot u^k \cdot n^k)$. Similarly, the cost of lines 77–78 is $O(|\Pi^s_{\mathsf{r}}| \cdot w^2 \cdot u^k \cdot n^k)$. The algorithm then enters the loop of lines 79–84. We focus on the cost of rule evaluation first, i.e., the cost of line 82, and we start with

examining the running time of handling one rule $r \in \Pi^s_{\mathsf{r}}$ throughout the loop. The rule could have at most w positive body atoms. Consider the ith body atom B of the rule, and let l be the number of distinct variables in B. Since Δ_D is distinct among iterations, atom B could be used as the pivot atom in at most $u^l \cdot n^l$ queries. Each of these queries require $O(w \cdot u^{k-l} \cdot n^{k-l})$ time to execute, so the cost of evaluating rule r in the loop of lines 79–84 using B as the pivot is $O(w \cdot u^k \cdot n^k)$. The number of positive body atoms in r is at most w, and the number of recursive rules is $|\Pi^s_{\mathsf{r}}|$. Therefore, the overall cost for the rule evaluation in line 82 throughout the loop is $O(|\Pi^s_{\mathsf{r}}| \cdot w^2 \cdot u^k \cdot n^k)$. Finally, during the execution of lines 79–84, the additional cost incurred due to iterating through the result multiset, manipulating sets, and accessing C_{nr} and C_{r} is linear in the total number of distinct rule instances considered in lines 75, 77, and 82. This is in the order of $O(|\Pi^s| \cdot u^k \cdot n^k)$, which is negligible. Hence, the cost of overdeletion for stratum Π^s is simply $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$.

We next examine the one-step rederivation step that takes place in line 70. For each fact $F \in D \cap \mathsf{O}^s$, the cost of verifying if $C_{\mathsf{r}}[F] > 0$ is $O(1)$. Hence, the cost of line 70 is simply $O(|\Pi^s| \cdot u^u \cdot n^u)$.

As for the insertion stage, the behaviour of the algorithm in terms of running time is similar to that of the overdeletion stage. In particular, lines 86–90 are analogous to lines 74–78, and thus require $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$ time. The loop of lines 91–96 is analogous to the loop of lines 79–84, and thus requires $O(|\Pi^s_{\mathsf{r}}| \cdot w^2 \cdot u^k \cdot n^k)$ time. Hence, the overall complexity for insertion is $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$.

The overall cost of handling one stratum Π^s is $O(|\Pi^s| \cdot w^2 \cdot u^k \cdot n^k)$. As a result, the loop of lines 68–71 requires $O(\Sigma_{1 \leq s \leq S} |\Pi^s| \cdot w^2 \cdot u^k \cdot n^k) = O(m \cdot w^2 \cdot u^k \cdot n^k)$. Finally, line 72 requires $O(m \cdot u^u \cdot n^u)$ time, which can be safely discarded. The algorithm thus runs in $O(m \cdot w^2 \cdot u^k \cdot n^k)$ time. □

B

Proof of Theorems in Chapter 5

B.1 Proof of Theorem 13

Theorem 13. *Algorithm 6 computes I as $\mathsf{mat}(\Pi, E)$, provided that each call of a module function made during the algorithm's execution is correct in the context of E and Π.*

Consider an execution of Algorithm 6 on program Π, stratification λ of Π with maximum stratum index S, and dataset E. Let $U^0 = I^0 = \emptyset$. Moreover, for each s with $1 \leq s \leq S$, let $U^s_0, U^s_1, \ldots$ be the sequence of sets where $U^s_0 = U^{s-1} \cup (E \cap \mathsf{O}^s)$, and $U^s_i = U^s_{i-1} \cup \Pi^s\left[U^s_{i-1}\right]$ for each $i > 0$; let $U^s = U^s_i$ where $U^s_i = U^s_{i+1}$ holds. It is straightforward to see that $U^s = \mathsf{O}^{\leq s} \cap \mathsf{mat}(\Pi, E)$ holds. Consider the execution of lines 145–155 for stratum s.

- Let $\ell(s)$ be the number of iterations for the loop of lines 148–155.
- For each iteration $1 \leq j \leq \ell(s)$ and module $1 \leq k \leq n(s)$, let $\Delta^{s,k}_j$ be the value of Δ^k after line 149 in the j-th iteration.
- For each module $1 \leq k \leq n(s)$,
  - let $C^{s,k}_1$ be the $\mathsf{Add}^{M^{s,k}}$ call made in line 147, and let $I|^{s,k}_1$, $\Delta^{\mathsf{p}}|^{s,k}_1$, $\Delta^{\mathsf{n}}|^{s,k}_1$, $\Delta^{\mathsf{m}}|^{s,k}_1$, and $J|^{s,k}_1$ be the arguments and result of the call;

  - for each iteration $2 \leq j \leq \ell(s)$, let $C_j^{s,k}$ be the $\mathsf{Add}^{M^{s,k}}$ call made in line 155 in the $(j-1)$-th iteration of lines 148–155, and let $I|_j^{s,k}$, $\Delta^{\mathsf{p}}|_j^{s,k}$, $\Delta^{\mathsf{n}}|_j^{s,k}$, $\Delta^{\mathsf{m}}|_j^{s,k}$, and $J|_j^{s,k}$ be the arguments and result of the call.

- Let I^s be the value of I after the loop of lines 145–155 finishes for stratum s.

We will now prove Theorem 38, which is more general then Theorem 13. The additional properties stated in Theorem 38 will be useful later when we establish the correctness of Theorem 18.

Theorem 38. *Each call of a module function made during the execution of Algorithm 6 on* Π, λ, *and* E *satisfy the conditions specified in the second column of Table 5.1. Moreover, for each* $1 \leq s \leq S$, *each* $1 \leq k \leq n(s)$, *and each* $2 \leq j \leq \ell(s)$, *calls* $C_{j-1}^{s,k}$ *and* $C_j^{s,k}$ *are both of type* Add, *with their arguments and results satisfying the respective conditions in Table 5.4. Furthermore, if each call of a module function made during the algorithm's execution is correct in the context of* E *and* Π, *then the algorithm computes* I *as* $\mathsf{mat}(\Pi, E)$, *and the following property holds for each stratum index* s *with* $1 \leq s \leq S$.

$$U^s = U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{\substack{1 \leq j \leq \ell(s) \\ 1 \leq k \leq n(s)}} \Delta_j^{s,k} \tag{B.1}$$

Consider arbitrary s, k, and j with $1 \leq s \leq S$, $1 \leq k \leq n(s)$, and $1 \leq j \leq \ell(s)$, we show that the arguments and result of $\mathsf{Add}^{M^{s,k}}$ call $C_j^{s,k}$ satisfies the conditions specified in the second column of Table 5.1—that is, the following properties hold. For simplicity, we make the assumption that (B.4) holds when (B.2) and (B.3) both hold. As we shall see later, this can be easily guaranteed by the implementation of module functions.

$$\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \subseteq I|_j^{s,k} \tag{B.2}$$

$$\Delta^{\mathsf{n}}|_j^{s,k} \cap I|_j^{s,k} = \emptyset \tag{B.3}$$

$$J|_j^{s,k} \cap I|_j^{s,k} = \emptyset \tag{B.4}$$

- For the case where $j = 1$ holds, by definition call $C_j^{s,k}$ is made in line 147, and so (B.2) and (B.3) both hold. Consequently, property (B.4) holds as well.

- For the case where $j > 1$ holds, by definition call $C_j^{s,k}$ is made in line 155, and so $\Delta^{\mathsf{n}}|_j^{s,k} = \emptyset$ holds, which implies property (B.3); moreover, lines 149, 151, 153, and 155 jointly imply that property (B.3) holds. Consequently, property (B.4) holds as well.

The choice of s, k, and j is arbitrary, and so each call of a module function made during the execution of Algorithm 6 on Π, λ, and E satisfies the conditions specified in the second column of Table 5.1.

Next we show that for each $1 \leq s \leq S$, each $1 \leq k \leq n(s)$, and each $2 \leq j \leq \ell(s)$, the arguments and results of Add calls $C_{j-1}^{s,k}$ and $C_j^{s,k}$ satisfy the respective conditions in Table 5.4—that is, the following properties hold.

$$\Delta^{\mathsf{m}}|_j^{s,k} = J|_{j-1}^{s,k} \tag{B.5}$$

$$I|_j^{s,k} = I|_{j-1}^{s,k} \cup \Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \tag{B.6}$$

To this end, consider arbitrary s, k, and j with $1 \leq s \leq S$, $1 \leq k \leq n(s)$, and $2 \leq j \leq \ell(s)$. We discuss the following two cases.

- For the case where $j = 2$ holds, line 147 ensures $\Delta_1^{s,k} = J|_1^{s,k}$; moreover, line 155 implies $\Delta^{\mathsf{m}}|_2^{s,k} = \Delta_1^{s,k}$. Consequently, we have $\Delta^{\mathsf{m}}|_j^{s,k} = J|_{j-1}^{s,k}$, as required. Furthermore, lines 147, 149, 151, and 155 jointly imply $I|_j^{s,k} = I|_{j-1}^{s,k} \cup \bigcup_{1 \leq k' \leq n(s)} \Delta_{j-1}^{s,k}$; lines 153 and 155 jointly imply $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} = \bigcup_{1 \leq k' \leq n(s)} \Delta_{j-1}^{s,k}$. Consequently, property (B.6) holds, as required.
- For the case where $2 < j \leq \ell(s)$ holds, line 155 implies $\Delta_{j-1}^{s,k} = J|_{j-1}^{s,k}$ and $\Delta^{\mathsf{m}}|_j^{s,k} = \Delta_{j-1}^{s,k}$. Consequently, property (B.5) holds. Furthermore, lines 149, 151, and 155 jointly imply $I|_j^{s,k} = I|_{j-1}^{s,k} \cup \bigcup_{1 \leq k' \leq n(s)} \Delta_{j-1}^{s,k}$. In addition, $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} = \bigcup_{1 \leq k' \leq n(s)} \Delta_{j-1}^{s,k}$ follows from lines 153 and 155. Consequently, property (B.6) holds, as required.

The choice of s, k, and j is arbitrary, and so properties (B.5) and (B.6) holds for each pair of $C_{j-1}^{s,k}$ and $C_j^{s,k}$ with $1 \leq s \leq S$, $1 \leq k \leq n(s)$, and $2 \leq j \leq \ell(s)$.

Next we show by induction on s with $0 \leq s \leq S$ that property (B.8) holds and that if $s > 0$ holds, then property (B.7) holds. For the base case where $s = 0$, property (B.8) trivially holds since both sides of the equation are empty. For the inductive step, consider

arbitrary s with $1 \leq s \leq S$ such that property (B.8) holds for $s-1$, and we show properties (B.7) and (B.8) for s. For clarity we break the proof into several claims.

$$U^s = U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{\substack{1 \leq j \leq \ell(s) \\ 1 \leq k \leq n(s)}} \Delta_j^{s,k} \tag{B.7}$$

$$I^s = U^s \tag{B.8}$$

Claim 39. *The $\subseteq$ direction of property* (B.7) *holds.*

Proof. We show by induction on i that $U_i^s \subseteq U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1 \leq j \leq \ell(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}$ holds for $i \geq 0$.

- For the induction base, by definition we have $U_0^s = U^{s-1} \cup (E \cap \mathsf{O}^s)$, and so $U_0^s \subseteq U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1 \leq j \leq \ell(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}$ clearly holds, as required.
- For the inductive step, consider arbitrary $i > 0$ such that U_{i-1}^s is contained in the right-hand side of (B.7): we would like to show that U_i^s is contained in the right-hand side of (B.7) as well. By definition $U_i^s = U_{i-1}^s \cup \Pi^s\left[U_{i-1}^s\right]$ holds, and so it is enough to prove $\Pi^s\left[U_{i-1}^s\right] \subseteq U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1 \leq j \leq \ell(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}$. To this end, consider arbitrary $F \in \Pi^s\left[U_{i-1}^s\right]$. There exists a module with index k such that $F \in M^{s,k}\left[U_{i-1}^s\right]$, and so there exist rule r and its instance r' such that $r \in M^{s,k}$, $\mathsf{b}^+(r') \subseteq U_{i-1}^s$, and $\mathsf{b}^-(r') \cap U_{i-1}^s = \emptyset$ all hold. By the induction assumption we have $\mathsf{b}^+(r') \subseteq U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1 \leq j \leq \ell(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}$. There are two possibilities.
  - $\mathsf{b}^+(r') \subseteq U^{s-1} \cup (E \cap \mathsf{O}^s)$ holds. $\mathsf{b}^-(r') \cap U_{i-1}^s = \emptyset$ implies $\mathsf{b}^-(r') \cap U_0^s = \emptyset$ and $\mathsf{b}^-(r') \cap (U^{s-1} \cup (E \cap \mathsf{O}^s)) = \emptyset$, and so $F \in M^{s,k}\left[U^{s-1} \cup (E \cap \mathsf{O}^s)\right]$ holds. Now if $F \notin U^{s-1} \cup (E \cap \mathsf{O}^s)$ holds, then $F \in M^{s,k\uparrow}[U^{s-1} \cup (E \cap \mathsf{O}^s) : U^{s-1} \cup (E \cap \mathsf{O}^s), \emptyset]$ follows from Definitions 9. Together with the induction assumption $I^{s-1} = U^{s-1}$ this implies $F \in M^{s,k\uparrow}[I^{s-1} \cup (E \cap \mathsf{O}^s) : I^{s-1} \cup (E \cap \mathsf{O}^s), \emptyset]$. But then, the fact that the $\mathsf{Add}^{M^{s,k}}$ call made in line 147 is correct and Definition 11 jointly imply $F \in J|_1^{s,k} = \Delta_1^{s,k}$. Thus, we have $F \in U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1 \leq j \leq \ell(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}$.
  - Otherwise, let m be the largest index such that $\mathsf{b}^+(r') \cap \bigcup_{1 \leq k \leq n(s)} \Delta_m^{s,k} \neq \emptyset$ holds. Then, $F \in M^{s,k}\left[U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1 \leq j \leq m, 1 \leq k \leq n(s)} \Delta_j^{s,k} : \bigcup_{1 \leq k \leq n(s)} \Delta_m^{s,k}\right]$ holds. Now if $F \notin U^{s-1} \cup (E \cap \mathsf{O}^s)$ and $F \notin \bigcup_{1 \leq j \leq m, 1 \leq k \leq n(s)} \Delta_j^{s,k}$ both hold,

then $F \in M^{s,k\uparrow}[U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1\leq j\leq m,1\leq k\leq n(s)} \Delta_j^{s,k} \mathbin{:} \bigcup_{1\leq k\leq n(s)} \Delta_m^{s,k}, \emptyset]$ follows from Definitions 9. Together with the induction assumption $I^{s-1} = U^{s-1}$ this implies $F \in M^{s,k\uparrow}[I^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1\leq j\leq m,1\leq k\leq n(s)} \Delta_j^{s,k} \mathbin{:} \bigcup_{1\leq k\leq n(s)} \Delta_m^{s,k}, \emptyset]$. But then, the fact that each $\mathsf{Add}^{M^{s,k}}$ call made in line 155 is correct and Definition 11 jointly imply $F \in J|_{m+1}^{s,k} = \Delta_{m+1}^{s,k}$, so F belongs to the right-hand side of (B.7).

Since the choice of F is arbitrary, $\Pi^s\left[U_{i-1}^s\right] \subseteq U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1\leq j\leq \ell(s),1\leq k\leq n(s)} \Delta_j^{s,k}$ holds, which completes our inductive proof. □

Claim 40. *The $\supseteq$ direction of property* (B.7) *holds.*

Proof. First, $U^{s-1} \cup (E \cap \mathsf{O}^s) \subseteq U^s$ holds by the definition of U^s. Next we prove by induction on m that $\bigcup_{1\leq j\leq m,1\leq k\leq n(s)} \Delta_j^{s,k} \subseteq U^s$ holds for $1 \leq m \leq \ell(s)$.

- For the base case $m = 1$, line 147 ensures $\bigcup_{1\leq j\leq m,1\leq k\leq n(s)} \Delta_j^{s,k} = \bigcup_{1\leq k\leq n(s)} J|_1^{s,k}$. Consider arbitrary k and F such that $F \in J|_1^{s,k}$ holds. Call $C_1^{s,k}$ is correct in the context of E and Π, and so by Definition 11 we have $F \in M^{s,k\uparrow}_{\infty}[I^{s-1} \cup (E \cap \mathsf{O}^s) \mathbin{:} I^{s-1} \cup (E \cap \mathsf{O}^s)]$. Let Δ_i and J_i be defined with regards to $M^{s,k\uparrow}_{\infty}[I^{s-1} \cup (E \cap \mathsf{O}^s) \mathbin{:} I^{s-1} \cup (E \cap \mathsf{O}^s)]$ in the same way as in Definition 9. We show by induction on i that $\Delta_i \subseteq U^s$ and $J_i \subseteq U^s$ hold for $i \geq 1$, which then implies $F \in U^s$.
  - For the induction base, we have $\Delta_1 \subseteq M^{s,k}\left[I^{s-1} \cup (E \cap \mathsf{O}^s) \mathbin{:} I^{s-1} \cup (E \cap \mathsf{O}^s)\right]$. Then, $M^{s,k} \subseteq \Pi^s$ implies $\Delta_1 \subseteq \Pi^s\left[I^{s-1} \cup (E \cap \mathsf{O}^s)\right]$, which together with the induction assumption $I^{s-1} = U^{s-1}$ and the definition of U_0^s implies $\Delta_1 \subseteq \Pi^s\left[U_0^s\right]$. Consequently, $\Delta_1 \subseteq U^s$ holds, as required. Moreover, $J_1 = \Delta_1 \subseteq U^s$ holds.
  - For the inductive step, consider arbitrary $i > 1$ such that $\Delta_{i-1} \subseteq U^s$ and $J_{i-1} \subseteq U^s$ hold. By Definition 9 we have $\Delta_i = M^{s,k}\left[U_0^s \cup J_{i-1} \mathbin{:} \Delta_{i-1}\right] \setminus (U_0^s \cup J_{i-1})$, and so $\Delta_i \subseteq M^{s,k}\left[U_0^s \cup J_{i-1}\right]$ holds. But then, the induction assumption $J_{i-1} \subseteq U^s$ implies $\Delta_i \subseteq M^{s,k}\left[U^s\right] \subseteq U^s$. Moreover, $J_i = J_{i-1} \cup \Delta_i \subseteq U^s$ holds, as required.

  The choice of k and F is arbitrary, so $\bigcup_{1\leq j\leq m,1\leq k\leq n(s)} \Delta_j^{s,k} \subseteq U^s$ holds for $m = 1$.
- For the inductive step, consider arbitrary $m > 1$ such that $\bigcup_{1\leq j\leq m-1,1\leq k\leq n(s)} \Delta_j^{s,k} \subseteq U^s$ holds. Then, line 155 ensures $\bigcup_{1\leq k\leq n(s)} \Delta_m^{s,k} = \bigcup_{1\leq k\leq n(s)} J|_m^{s,k}$. Now consider arbitrary k with $1 \leq k \leq n(s)$. Call $C_m^{s,k}$ is correct in the context of E and Π, and so we

have $\Delta_m^{s,k} \subseteq M^{s,k\uparrow}_{\infty}[U_0^s \cup \bigcup_{1\leq j\leq m-1,1\leq k'\leq n(s)} \Delta_j^{s,k'} : \bigcup_{1\leq k'\leq n(s)} \Delta_{m-1}^{s,k'}]$. Let Δ_i and J_i be defined with regards to $M^{s,k\uparrow}_{\infty}[U_0^s \cup \bigcup_{1\leq j\leq m-1,1\leq k'\leq n(s)} \Delta_j^{s,k'} : \bigcup_{1\leq k'\leq n(s)} \Delta_{m-1}^{s,k'}]$ in the same way as in Definition 9. We show by induction on i that $\Delta_i \subseteq U^s$ and $J^i \subseteq U^s$ hold for $i \geq 1$.

- For the induction base, $M^{s,k} \subseteq \Pi^s$ implies $\Delta_1 \subseteq \Pi^s\Big[U_0^s \cup \bigcup_{1\leq j\leq m-1,1\leq k\leq n(s)} \Delta_j^{s,k}\Big]$. Together with the induction assumption $\bigcup_{1\leq j\leq m-1,1\leq k\leq n(s)} \Delta_j^{s,k} \subseteq U^s$ this implies $\Delta_1 \subseteq \Pi^s\Big[U^s\Big] \subseteq U^s$. Moreover, $J_1 = \Delta_1 \subseteq U^s$ holds, as required.
- For the inductive step, consider arbitrary $i > 1$ such that $\Delta_{i-1} \subseteq U^s$ and $J_{i-1} \subseteq U^s$ hold. We have $\Delta_i \subseteq \Pi^s\Big[U_0^s \cup \bigcup_{1\leq j\leq m-1,1\leq k\leq n(s)} \Delta_j^{s,k} \cup J_{i-1}\Big]$ by Definition 9. But then, the induction assumptions for $m-1$ and $i-1$ imply $\Delta_i \subseteq \Pi^s\Big[U^s\Big] \subseteq U^s$. Moreover, $J_i = J_{i-1} \cup \Delta_i \subseteq U^s$ holds, as required.

The choice of k is arbitrary, so $\Delta_m^{s,k} \subseteq U^s$ holds for each $1 \leq k \leq n(s)$. But then, $\bigcup_{1\leq k\leq n(s)} \Delta_m^{s,k} \subseteq U^s$ holds. Together with the induction assumption for $m-1$ this implies $\bigcup_{1\leq j\leq m,1\leq k\leq n(s)} \Delta_j^{s,k} \subseteq U^s$. This completes our proof for Claim 40. □

Claim 41. *The $\subseteq$ direction of property* (B.8) *holds.*

Proof. Consider arbitrary $F \in I^s$. If $F \in I^{s-1} \cup (E \cap \mathsf{O}^s)$, then the induction assumption $I^{s-1} = U^{s-1}$ ensures $F \in U^{s-1} \cup (E \cap \mathsf{O}^s) \subseteq U^s$. Otherwise, F is added to I^s at some point in line 151 during the execution of lines 145–155 for stratum s. Consequently, due to line 149 there exists j and k with $1 \leq j \leq \ell(s)$ and $1 \leq k \leq n(s)$ such that $F \in \Delta_j^{s,k}$ holds. But then, Claim 40 ensures $F \in U^s$. The choice of F is arbitrary, so $I^s \subseteq U^s$ holds, as required. □

Claim 42. *The $\supseteq$ direction of property* (B.8) *holds.*

Proof. Consider arbitrary $F \in U^s$. Then, $F \in U^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1\leq j\leq \ell(s),1\leq k\leq n(s)} \Delta_j^{s,k}$ follows from Claim 39. Together with the induction assumption $I^{s-1} = U^{s-1}$ this implies $F \in I^{s-1} \cup (E \cap \mathsf{O}^s) \cup \bigcup_{1\leq j\leq \ell(s),1\leq k\leq n(s)} \Delta_j^{s,k}$. Now if $F \in I^{s-1} \cup (E \cap \mathsf{O}^s)$, then line 145 ensures $F \in I^s$. Otherwise, there exist j with $1 \leq j \leq \ell(s)$ and k with $1 \leq k \leq n(s)$ such that $F \in \Delta_j^{s,k}$ holds. But then, F is first added to Δ in line 149 and later added to I

in line 151 in the j-th iteration of line 148–155. Consequently, $F \in I^s$ holds. Since the choice of F is arbitrary, we have $U^s \subseteq I^s$, as required. □

B.2 Proof of Theorem 14

Theorem 14. *Algorithm 7 updates dataset I from $\mathsf{mat}(\Pi, E)$ to $\mathsf{mat}(\Pi, (E \setminus E^-) \cup E^+)$, provided that each module function call made during the execution of the algorithm is correct in the context of $(E \setminus E^-) \cup E^+$ and Π.*

Consider an execution of Algorithm 7 on program Π, stratification λ of Π with maximum stratum index S, and datasets E, I, E^- and E^+ where $I = \mathsf{mat}(\Pi, E)$ holds. Due to line 156, without loss of generality we assume that $E^- \subseteq E$ and $E^+ \cap E = \emptyset$ hold. Now let $E|_o = E$ and $U^0|_o = \emptyset$. Moreover, for each s with $1 \leq s \leq S$, let $U^s_0|_o, U^s_1|_o, \ldots$ be the sequence of sets where $U^s_0|_o = U^{s-1}|_o \cup (E|_o \cap \mathsf{O}^s)$, and for $i > 0$, $U^s_i|_o = U^s_{i-1}|_o \cup \Pi^s\left[U^s_{i-1}|_o\right]$. Index k clearly exists at which the sequence reach the fixpoint (i.e., $U^s_k|_o = U^s_{k+1}|_o$), so let $U^s|_o = U^s_k|_o$. It is straightforward to see that $U^s|_o = \mathsf{O}^{\leq s} \cap \mathsf{mat}(\Pi, E|_o)$ holds. Now let $E|_n = (E|_o \setminus E^-) \cup E^+$, and let $U^s_i|_n$ and $U^s|_n$ be defined analogously. Finally, let $D^0 = A^0 = \emptyset$; for each $1 \leq s \leq S$, consider the execution of lines 158–159 for stratum s.

- Let D^s and A^s be the values of D and A, respectively, after lines 158–159 finish for stratum s.
- Let $\ell(s)$ be the number of iterations for the loop of lines 167–173 in the call of line 158, and for each iteration $1 \leq j \leq \ell(s)$ and each module $1 \leq k \leq n(s)$, let D^s_j, $\Delta^s_j|_d$ and $\Delta^{s,k}_j|_d$ be the values of D, Δ and Δ^k, respectively, at the beginning of the j-th iteration of the loop.
- Let $I^s|_d = I \setminus (D^s \setminus A^{s-1})$.
- Let $m(s)$ be the number of iterations for the loop of lines 183–190 in the call of line 159, and for each iteration $1 \leq j \leq m(s)$, let A^s_j, $\Delta^s_j|_a$ and $\Delta^{s,k}_j|_a$, $1 \leq k \leq n(s)$, be the values of A, Δ and Δ^k, respectively, after line 184 in the j-th iteration of the loop.
- For each $1 \leq k \leq n(s)$,

- let $C_1^{s,k}$ be the $\mathsf{Del}^{M^{s,k}}$ call made in line 163, and let $I^{\mathsf{p}}|_1^{s,k}$, $I^{\mathsf{n}}|_1^{s,k}$, $\Delta^{\mathsf{p}}|_1^{s,k}$, $\Delta^{\mathsf{n}}|_1^{s,k}$, $\Delta^{\mathsf{m}}|_1^{s,k}$, and $J|_1^{s,k}$ be its arguments and result;
- for each $1 < j \leq \ell(s) + 1$, let $C_j^{s,k}$ be the $\mathsf{Del}^{M^{s,k}}$ call made in line 169 in the $(j-1)$-th iteration of lines 167–173, and let $I^{\mathsf{p}}|_j^{s,k}$, $I^{\mathsf{n}}|_j^{s,k}$, $\Delta^{\mathsf{p}}|_j^{s,k}$, $\Delta^{\mathsf{n}}|_j^{s,k}$, $\Delta^{\mathsf{m}}|_j^{s,k}$, and $J|_j^{s,k}$ be the arguments and result of the call;
- for $j = \ell(s)+2$, let $C_j^{s,k}$ be the $\mathsf{Red}^{M^{s,k}}$ call made in line 176, and let $I^{\mathsf{p}}|_j^{s,k}$, $I^{\mathsf{n}}|_j^{s,k}$, $\Delta|_j^{s,k}$, and $J|_j^{s,k}$ be the arguments and result;
- for $j = \ell(s)+3$, let $C_j^{s,k}$ be the $\mathsf{Add}^{M^{s,k}}$ call made in line 182, and let $I|_j^{s,k}$, $\Delta^{\mathsf{p}}|_j^{s,k}$, $\Delta^{\mathsf{n}}|_j^{s,k}$, $\Delta^{\mathsf{m}}|_j^{s,k}$, and $J|_j^{s,k}$ be the arguments and result;
- for each $\ell(s) + 3 < j < \ell(s) + m(s) + 3$, let $C_j^{s,k}$ be the $\mathsf{Add}^{M^{s,k}}$ call made in line 190 in the $(j - \ell_i(s) - 3)$th iteration of lines 183–190, and let $I|_j^{s,k}$, $\Delta^{\mathsf{p}}|_j^{s,k}$, $\Delta^{\mathsf{n}}|_j^{s,k}$, $\Delta^{\mathsf{m}}|_j^{s,k}$, and $J|_j^{s,k}$ be the arguments and result.

We will now prove Theorem 43, which is more general then Theorem 14. The additional properties stated in Theorem 43 will be useful later when we establish the correctness of Theorem 18.

Theorem 43. *Each call of a module function made during the execution of Algorithm 7 on* Π, λ, S, $E|_o$, E^-, *and* E^+ *satisfy the conditions specified in the second column of Table 5.1. Moreover, for each* $1 \leq s \leq S$, *each* $1 \leq k \leq n(s)$, *and each* $2 \leq j \leq \ell(s) + m(s) + 2$, *calls* $C_{j-1}^{s,k}$ *and* $C_j^{s,k}$ *must be of one of the forms from Table 5.4, with their arguments and results satisfying the respective conditions. Furthermore, for each stratum index* s *with* $1 \leq s \leq S$, *if each module function call made during the execution of Algorithm 7 before line 158 finishes for stratum* s *is correct in the context of* $E|_n$ *and* Π, *then the following property holds.*

$$U^s|_o \setminus U^s|_n \subseteq (D^{s-1} \setminus A^{s-1}) \cup (E^- \cap \mathsf{O}^s) \cup \bigcup_{\substack{1 \leq j \leq \ell(s) \\ 1 \leq k \leq n(s)}} \Delta_j^{s,k}|_d$$

If each module function call made before line 159 finishes for stratum s is correct in the context of $E|_n$ and Π, then the following property hold.

$$U^s|_n \setminus I^s|_d = (A_1^s \setminus D^{s-1}) \cup \bigcup_{\substack{1 \le j \le m(s) \\ 1 \le k \le n(s)}} \Delta_j^{s,k}|_a$$

Finally, if each module function call made during the execution of the algorithm is correct in the context of $E|_n$ and Π, then the algorithm updates dataset I from $\mathsf{mat}(\Pi, E|_o)$ to $\mathsf{mat}(\Pi, E|_n)$.

First we show by induction on s with $0 \le s \le S$ that the following properties hold. The induction base is trivial since $D^0 = A^0 = \emptyset$ holds. For the inductive step, consider arbitrary s with $1 \le s \le S$ such that properties (a)–(c) hold for $s-1$, and we show that these properties hold for s as well. The proof is lengthy, so for clarity we break it into several claims.

(a) If $s > 0$, then for each k with $1 \le k \le n(s)$ and each j with $1 \le j \le \ell(s) + m(s) + 2$, the arguments and result of call $C_j^{s,k}$ satisfy the conditions specified in the second column of Table 5.1.

(b) $D^s \subseteq I$.

(c) $(A^s \setminus D^s) \cap I = \emptyset$.

Claim 44. *For each k with $1 \le k \le n(s)$ and each j with $1 \le j \le \ell(s) + m(s) + 2$, the arguments and result of call $C_j^{s,k}$ satisfy the conditions specified in the second column of Table 5.1.*

Proof. As in the proof for Theorem 38 we make the assumption that whenever the arguments of a call satisfy the conditions specified in the second column of Table 5.1, the result of the call satisfies the corresponding conditions as well. We show by induction on j with $1 \le j \le \ell(s) + m(s) + 2$ that the arguments and result of each call $C_j^{s,k}$ with $1 \le k \le n(s)$ satisfy the conditions specified in the second column of Table 5.1.

- For the induction base where $j = 1$, consider arbitrary k with $1 \le k \le n(s)$. Call $C_j^{s,k}$ is of type Del and is made in line 163. Consequently, we have $I^{\mathsf{p}}|_j^{s,k} = I$ and $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} = D^{s-1} \setminus A^{s-1}$. But then, the induction assumption $D^{s-1} \subseteq I$ implies

$\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \subseteq I^{\mathsf{p}}|_j^{s,k}$, as required. Moreover, by the induction assumption for (c) we have $\Delta^{\mathsf{n}}|_j^{s,k} \cap I^{\mathsf{n}}|_j^{s,k} = \emptyset$. Therefore, the arguments and result of call $C_j^{s,k}$ satisfy the conditions specified in the second column of Table 5.1.

- For the inductive step, consider arbitrary $2 \leq j \leq \ell(s) + m(s) + 2$ such that the arguments and result of each call $C_j^{s,k}$ with $1 \leq k \leq n(s)$ satisfy the conditions specified in the second column of Table 5.1. We discuss the following possibilities.
  - $j = 2$ holds. Consider arbitrary $C_j^{s,k}$ with $1 \leq k \leq n(s)$. The call is of type Del and is made in line 169. Line 163 and the induction assumption for $j-1$ jointly imply $\Delta_{j-1}^{s,k}|_d = J_{j-1}^{s,k} \subseteq I \setminus (D^{s-1} \setminus A^{s-1})$. The choice of k is arbitrary, so we have $\bigcup_{1 \leq k' \leq n(s)} \Delta_{j-1}^{s,k'}|_d \subseteq I \setminus (D^{s-1} \setminus A^{s-1})$. Moreover, lines 166 and 169 jointly imply $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} = (E^- \cap \mathsf{O}^s) \cup \bigcup_{1 \leq k' \leq n(s)} \Delta_{j-1}^{s,k'}|_d$. Consequently, we have $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \subseteq I^{\mathsf{p}}|_j^{s,k}$, as required. Finally, $\Delta^{\mathsf{n}}|_j^{s,k} = \emptyset$, and so $\Delta^{\mathsf{n}}|_j^{s,k} \cap I^{\mathsf{n}}|_j^{s,k} = \emptyset$ holds. Therefore, the arguments and result of call $C_j^{s,k}$ satisfy the conditions specified in the second column of Table 5.1 in this case.
  - $3 \leq j \leq \ell(s) + 1$ holds. Consider arbitrary $C_j^{s,k}$ with $1 \leq k \leq n(s)$. The call is of type Del and is made in line 169, so the induction assumption for $j-1$ implies $\Delta_{j-1}^{s,k}|_d = J|_{j-1}^{s,k} \subseteq I \setminus ((D_{j-2}^s \setminus A^{s-1}) \cup \bigcup_{1 \leq k' \leq n(s)} \Delta_{j-2}^{s,k}|_d)$. Together with line 170 this implies $\Delta_{j-1}^{s,k}|_d \subseteq I \setminus (D_{j-1}^s \setminus A^{s-1})$. The choice of k is arbitrary, so we have $\bigcup_{1 \leq k' \leq n(s)} \Delta_{j-1}^{s,k'}|_d \subseteq I \setminus (D_{j-1}^s \setminus A^{s-1})$. Moreover, lines 173 and 169 jointly imply $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} = \bigcup_{1 \leq k' \leq n(s)} \Delta_{j-1}^{s,k'}|_d$. Consequently, we have $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \subseteq I^{\mathsf{p}}|_j^{s,k}$, as required. Finally, $\Delta^{\mathsf{n}}|_j^{s,k} = \emptyset$, and so $\Delta^{\mathsf{n}}|_j^{s,k} \cap I^{\mathsf{n}}|_j^{s,k} = \emptyset$ holds. Therefore, the arguments and result of call $C_j^{s,k}$ satisfy the conditions specified in the second column of Table 5.1 in this case.
  - $j = \ell(s) + 2$ holds. Consider arbitrary $C_j^{s,k}$ with $1 \leq k \leq n(s)$. The call is of type Red and is made in line 176, so $I^{\mathsf{p}}|_j^{s,k} = I \setminus (D^s \setminus A^{s-1})$ and $\Delta|_j^{s,k} = D^s \cap \mathsf{O}^s$ holds. Now consider arbitrary $F \in D^s \cap \mathsf{O}^s$. Then, $F \in \mathsf{O}^s$ implies $F \notin A^{s-1}$. Consequently, $F \in D^s \setminus A^{s-1}$ holds, and so we have $F \notin I^{\mathsf{p}}|_j^{s,k}$. The choice of F is arbitrary, so $\Delta|_j^{s,k} \cap I^{\mathsf{p}}|_j^{s,k} = \emptyset$ holds, as required. Therefore, the arguments and result of $C_j^{s,k}$ satisfy the conditions specified in the second column of Table 5.1.

– $j = \ell(s) + 3$ holds. Consider arbitrary $C_j^{s,k}$ with $1 \leq k \leq n(s)$. The call is of type Add and is made in line 182. Consequently, we have $I|_j^{s,k} = (I \setminus D^s) \cup A_1^s$. But then, lines 178, 180, and 182 jointly imply $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \subseteq A_1^s$, and so $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \subseteq I|_j^{s,k}$ holds, as required. To see that $\Delta^{\mathsf{n}}|_j^{s,k} \cap I|_j^{s,k} = \emptyset$ holds as well, consider arbitrary $F \in \Delta^{\mathsf{n}}|_j^{s,k} = D^s \setminus A_1^s$. Then, $F \in D^s$ implies $F \notin I \setminus D^s$, which together with $F \notin A_1^s$ implies $F \notin (I \setminus D^s) \cup A_1^s = I|_j^{s,k}$. The choice of F is arbitrary, so $\Delta^{\mathsf{n}}|_j^{s,k} \cap I|_j^{s,k} = \emptyset$ holds. Therefore, the arguments and result of $C_j^{s,k}$ satisfy the conditions specified in the second column of Table 5.1.

– $\ell(s) + 4 \leq j \leq \ell(s) + m(s) + 2$ holds. Consider arbitrary $C_j^{s,k}$ with $1 \leq k \leq n(s)$. The call is of type Add and is made in line 190. Consequently, we have $I|_j^{s,k} = (I \setminus D^s) \cup A_{j-\ell(s)-2}^s$. But then, lines 184, 186, 188, and 190 jointly imply $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \subseteq A_{j-\ell(s)-2}^s$, and so $\Delta^{\mathsf{p}}|_j^{s,k} \cup \Delta^{\mathsf{m}}|_j^{s,k} \subseteq I|_j^{s,k}$ holds, as required. Moreover, $\Delta^{\mathsf{n}}|_j^{s,k} = \emptyset$ implies $\Delta^{\mathsf{n}}|_j^{s,k} \cap I|_j^{s,k} = \emptyset$. Therefore, the arguments and result of $C_j^{s,k}$ satisfy the conditions specified in the second column of Table 5.1. □

Claim 45. *Property (b) holds.*

Proof. Consider arbitrary $F \in D^s$. If $F \in D^{s-1}$ holds, then the induction assumption for (b) ensures that $F \in I$ holds. Otherwise, F is added to D^s in line 170 at some point during the execution of line 158 for stratum s. Note that a fact can only be added to D^s via Δ. There are two possibilities.

- $F \in E^- \cap \mathsf{O}^s$ holds. Then, $F \in I$ clearly holds.
- F is added to Δ via some Δ^k. In this case, there exist k with $1 \leq k \leq n(s)$ and j with $1 \leq j \leq \ell(s) + 1$ such that $F \in J|_j^{s,k}$ holds. But then, Claim 44 implies $F \in I$, as required.

The choice of F is arbitrary, so $D^s \subseteq I$ holds. □

Claim 46. *Property (c) holds.*

Proof. Consider arbitrary $F \in A^s \setminus D^s$. If $F \in A^{s-1}$ holds, then $F \notin D^s$ implies $F \notin D^{s-1}$, and so the induction assumption for (c) ensures that $F \notin I$ holds. Otherwise, F is added

to A^s in line 178 or line 186 at some point during the execution of line 159 for stratum s. There are two possibilities.

- F is added to N in line 177. $F \notin D^s$ implies $F \in (E^+ \cap \mathsf{O}^s) \setminus I$, and so $F \notin I$ holds.
- F is added to A^s via some Δ^k. Then, there exist k with $1 \leq k \leq n(s)$ and j with $\ell(s) + 2 \leq j \leq \ell(s) + m(s) + 2$ such that $F \in J|_j^{s,k}$ holds. But then, Claim 44 and $F \notin D^s$ jointly imply $F \notin I$, as required.

The choice of F is arbitrary, so $(A^s \setminus D^s) \cap I = \emptyset$ holds. □

Next we show that for each $1 \leq s \leq S$, $1 \leq k \leq n(s)$, and $2 \leq j \leq \ell(s) + m(s) + 2$, the arguments and results of calls $C_{j-1}^{s,k}$ and $C_j^{s,k}$ must be of one of the forms from Table 5.4, with their arguments and results satisfying the respective conditions. To this end, consider arbitrary s, k, and j with $1 \leq s \leq S$, $1 \leq k \leq n(s)$, and $2 \leq j \leq \ell(s) + m(s) + 2$. We consider the following cases.

- $2 \leq j \leq \ell(s) + 1$ holds. Then, by definition $C_{j-1}^{s,k}$ and $C_j^{s,k}$ are both of type Del. There are two possibilities.
  - $j = 2$ holds. Then, call $C_{j-1}^{s,k}$ is made in line 163, and call $C_j^{s,k}$ is made in line 169. Consequently, we have $\Delta^{\mathsf{m}}|_j^{s,k} = J|_{j-1}^{s,k} = \Delta_{j-1}^{s,k}|_d$. Moreover, $I^{\mathsf{p}}|_{j-1}^{s,k} = I$, $\Delta^{\mathsf{p}}|_{j-1}^{s,k} \cup \Delta^{\mathsf{m}}|_{j-1}^{s,k} = D^{s-1} \setminus A^{s-1}$, and $I^{\mathsf{p}}|_j^{s,k} = I \setminus (D^{s-1} \setminus A^{s-1})$ jointly imply $I^{\mathsf{p}}|_j^{s,k} = I^{\mathsf{p}}|_{j-1}^{s,k} \setminus (\Delta^{\mathsf{p}}|_{j-1}^{s,k} \cup \Delta^{\mathsf{m}}|_{j-1}^{s,k})$, as required.
  - $3 \leq j \leq \ell(s) + 1$ holds. Then, both $C_{j-1}^{s,k}$ and $C_j^{s,k}$ are Del calls made in line 169. Consequently, we have $\Delta^{\mathsf{m}}|_j^{s,k} = J|_{j-1}^{s,k} = \Delta_{j-1}^{s,k}|_d$. Moreover, the way P^k and Δ are constructed implies $I^{\mathsf{p}}|_{j-1}^{s,k} \setminus (\Delta^{\mathsf{p}}|_{j-1}^{s,k} \cup \Delta^{\mathsf{m}}|_{j-1}^{s,k}) = I \setminus ((D_{j-2}^s \setminus A^{s-1}) \cup \Delta_{j-2}^s|_d)$, so $I^{\mathsf{p}}|_j^{s,k} = I \setminus (D_{j-1}^s \setminus A^{s-1})$ and line 170 imply $I^{\mathsf{p}}|_j^{s,k} = I^{\mathsf{p}}|_{j-1}^{s,k} \setminus (\Delta^{\mathsf{p}}|_{j-1}^{s,k} \cup \Delta^{\mathsf{m}}|_{j-1}^{s,k})$.
  - $j = \ell(s) + 2$ holds. Then, $C_{j-1}^{s,k}$ is a Del call made in line 169 and $C_j^{s,k}$ is a Red call made in line 176. The loop of lines 167–173 terminates after $\ell(s)$ iterations, so line 167 and the way Δ is constructed ensure $J|_{j-1}^{s,k} = \emptyset$. Moreover, in the same way as in the previous case we have $I^{\mathsf{p}}|_j^{s,k} = I^{\mathsf{p}}|_{j-1}^{s,k} \setminus (\Delta^{\mathsf{p}}|_{j-1}^{s,k} \cup \Delta^{\mathsf{m}}|_{j-1}^{s,k})$. Finally, line 176 and property (b) jointly imply $\Delta|_j^{s,k} = I^{\mathsf{p}}|_1^{s,k} \setminus I^{\mathsf{p}}|_j^{s,k}$, as required.

– $j = \ell(s) + 3$ holds. Then, $C^{s,k}_{j-1}$ is a Red call made in line 176, and $C^{s,k}_{j}$ is a Add call made in line 182. Consequently, we have $\Delta^{\mathsf{m}}|^{s,k}_{j} = J|^{s,k}_{j-1}$. Next we prove that $I|^{s,k}_{j} = I|^{s,k}_{j-1} \cup \Delta^{\mathsf{p}}|^{s,k}_{j} \cup \Delta^{\mathsf{m}}|^{s,k}_{j}$ holds.

  * For the $\subseteq$ direction, consider arbitrary $F \in I|^{s,k}_{j} = (I \setminus D^s) \cup A^s_1$. There are several cases.

    · $F \in I \setminus D^s$ holds. Then, we clearly have $F \in I \setminus (D^s \setminus A^{s-1}) = I|^{s,k}_{j-1}$.

    · $F \in A^{s-1} \cap D^s$ holds. Then, $F \notin D^s \setminus A^{s-1}$ holds, and so property (b) implies $F \in I \setminus (D^s \setminus A^{s-1}) = I^{s,k}_{j-1}$.

    · $F \in A^{s-1} \setminus D^s$ holds. Then, due to line 180 we have $F \in \Delta^{\mathsf{p}}|^{s,k}_{j}$.

    · $F \in A^s_1 \setminus A^{s-1}$ holds. Then, due to line 178 we have $F \in \Delta^{\mathsf{p}}|^{s,k}_{j} \cup \Delta^{\mathsf{m}}|^{s,k}_{j}$.

    The choice of F is arbitrary, so the $\subseteq$ direction holds.

  * For the $\supseteq$ direction, we have $I|^{s,k}_{j-1} = I \setminus (D^s \setminus A^{s-1}) \subseteq (I \setminus D^s) \cup A^{s-1}$, and so $I|^{s,k}_{j-1} \subseteq I|^{s,k}_{j}$ holds; moreover, we have already shown in the proof of Claim 44 that $\Delta^{\mathsf{p}}|^{s,k}_{j} \cup \Delta^{\mathsf{m}}|^{s,k}_{j} \subseteq I|^{s,k}_{j}$ holds. Consequently, the $\supseteq$ direction holds as well.

– $\ell(s) + 4 \leq j \leq \ell(s) + m(s) + 2$ holds. Then, $C^{s,k}_{j-1}$ and $C^{s,k}_{j}$ are both Add calls with $\Delta^{\mathsf{m}}|^{s,k}_{j} = J|^{s,k}_{j-1}$. Moreover, lines 184, 186, and 188 jointly imply $I|^{s,k}_{j} = I|^{s,k}_{j-1} \cup \Delta^{\mathsf{p}}|^{s,k}_{j} \cup \Delta^{\mathsf{m}}|^{s,k}_{j}$, as required.

The choice of s, k, and j is arbitrary, so the required properties hold for each $1 \leq s \leq S$, each $1 \leq k \leq n(s)$, and each $2 \leq j \leq \ell(s) + m(s) + 2$.

Due to line 156 of Algorithm 7, without loss of generality we assume that $E^- \subseteq E$ and $E^+ \cap E = \emptyset$. Now let $E|_o = E$ and let $I^0|_o = \emptyset$. Moreover, for each $1 \leq s \leq S$, let $I^s_0|_o, I^s_1|_o, \ldots$ be the sequence of sets where $I^s_0|_o = I^{s-1}|_o \cup (E|_o \cap \mathsf{O}^s)$, and for $i > 0$, $I^s_i|_o = I^s_{i-1}|_o \cup \Pi^s\big[I^s_{i-1}|_o\big]$. Index k clearly exists at which the sequence reaches the fixpoint (i.e., $I^s_k|_o = I^s_{k+1}|_o$), so let $I^s|_o = I^s_k|_o$. Finally, let $I|_o = I^S|_o$; we clearly have $I|_o = \mathsf{mat}(\Pi, E|_o)$—that is, $I|_o$ is the 'old' materialisation. Now let $E|_n = (E|_o \setminus E^-) \cup E^+$, and let $I^s_i|_n$, $I^s|_n$, and $I|_n$ be defined analogously, so $I|_n$ is the 'new' materialisation.

Finally we show by induction on s with $1 \leq s \leq S$ that the following properties hold. Then, property (c) for $s = S$ and the way I is updated in line 160 imply that the algorithm correctly updates I from $\mathsf{mat}(\Pi, E|_o)$ to $\mathsf{mat}(\Pi, E|_n)$ provided that each module function call is correct in the context of Π and $E|_n$. The proof is lengthy, so we break it into several claims.

(a) If each module function call made before line 158 finishes for stratum s is correct in the context of $E|_n$ and Π, then (B.9)–(B.10) hold.

$$D^s \setminus A^{s-1} = (D^{s-1} \setminus A^{s-1}) \cup (E^- \cap \mathsf{O}^s) \cup \bigcup_{\substack{1 \leq j \leq \ell(s) \\ 1 \leq k \leq n(s)}} \Delta_j^{s,k}|_d \tag{B.9}$$

$$U^s|_o \setminus U^s|_n \subseteq D^s \setminus A^{s-1} \tag{B.10}$$

(b) If each module function call made before line 159 finishes for stratum s is correct in the context of $E|_n$ and Π, then (B.11)–(B.12) hold.

$$A^s \setminus D^{s-1} = (A_1^s \setminus D^{s-1}) \cup \bigcup_{\substack{1 \leq j \leq m(s) \\ 1 \leq k \leq n(s)}} \Delta_j^{s,k}|_a \tag{B.11}$$

$$U^s|_n \setminus I^s|_d = A^s \setminus D^{s-1} \tag{B.12}$$

(c) If each module function call made before line 159 finishes for stratum s is correct in the context of $E|_n$ and Π, then (B.13) holds.

$$(U^s|_o \setminus D^s) \cup A^s = U^s|_n \tag{B.13}$$

Claim 47. *Property (a) holds for $s = 1$.*

Proof. Assume that each module function call made before line 158 finishes for stratum 1 is correct in the context of $E|_n$ and Π. We show properties (B.9)–(B.10) for $s = 1$.

For the $\subseteq$ direction of property (B.9), consider arbitrary $F \in D^1 \setminus A^0 = D^1$. Then, F is added to D via Δ in line 170. The way Δ is constructed in in line 164 and line 171 implies $F \in (E^- \cap \mathsf{O}^1) \cup \bigcup_{1 \leq j \leq \ell(1), 1 \leq k \leq n(1)} \Delta_j^{1,k}|_d$. The choice of F is arbitrary, so the $\subseteq$ direction holds.

For the $\supseteq$ direction, consider arbitrary $F \in (E^- \cap \mathsf{O}^1) \cup \bigcup_{1 \leq j \leq \ell(1), 1 \leq k \leq n(1)} \Delta_j^{1,k}|_d$. Now if $F \in (E^- \cap \mathsf{O}^1) \cup \bigcup_{1 \leq k \leq n(1)} \Delta_1^{1,k}|_d$ holds, then lines 164 and 170 ensure that $F \in D^1$ holds. Otherwise, lines 171 and 170 ensure that $F \in D^1$ holds. The choice of F is arbitrary, so the $\supseteq$ direction holds as well.

To see that (B.10) holds for $s = 1$, we prove by induction on i that (B.14) holds.

$$U_i^1|_o \setminus U^1|_n \subseteq D^1 \tag{B.14}$$

For the base case where $i = 0$, consider arbitrary $F \in U_0^1|_o \setminus U^1|_n$. By defnition we have $U_0^1|_o = U^0|_o \cup (E|_o \cap \mathsf{O}^1) = E|_o \cap \mathsf{O}^1$, and so $F \in E|_o \cap \mathsf{O}^1$ holds. Moreover, $F \notin U^1|_n$ implies $F \notin E|_n \cap \mathsf{O}^1$. Consequently, $F \in E^- \cap \mathsf{O}^1 \subseteq D^1$ holds.

For the inductive step, assume that $U_{i-1}^1|_o$ satisfies (B.14) for $i > 0$, and consider arbitrary $F \in U_i^1|_o \setminus U^s|_n$. If $F \in U_{i-1}^1|_o$, then by the induction assumption $F \in D^1$ holds. Otherwise, we have $F \in \Pi^1\big[U_{i-1}^1|_o\big]$, so there exist a rule $r \in \Pi^1$ with only positive body atoms and a substitution σ such that $\mathsf{b}^+(r)\sigma \subseteq U_{i-1}^1|_o$ holds. Moreover, $F \notin U^1|_n$ implies $F \notin \Pi^1\big[U^1|_n\big]$, and so $\mathsf{b}^+(r)\sigma \not\subseteq U^1|_n$ holds. Consequently, we have $\mathsf{b}^+(r)\sigma \cap (U_{i-1}^1|_o \setminus U^1|_n) \neq \emptyset$, and so the induction assumption for $i-1$ implies $\mathsf{b}^+(r)\sigma \cap D^1 \neq \emptyset$. Now let j be the smallest index such that $\mathsf{b}^+(r)\sigma \cap D_j^1 \neq \emptyset$ holds. By definition D_1^1 is empty, so we have $j \geq 2$. Since j is the smallest such index, we have $\mathsf{b}^+(r)\sigma \cap D_{j-1}^1 = \emptyset$, which together with $\mathsf{b}^+(r)\sigma \subseteq U_{i-1}^1|_o$ implies $\mathsf{b}^+(r)\sigma \subseteq U^1|_o \setminus (D_{j-1}^1 \setminus A^0)$. Moreover, $\mathsf{b}^+(r)\sigma \cap D_{j-1}^1 = \emptyset$ and $\mathsf{b}^+(r)\sigma \cap D_j^1 \neq \emptyset$ jointly imply $\mathsf{b}^+(r)\sigma \cap \Delta_{j-1}^1|_d \neq \emptyset$. Let k be the index such that $r \in M^{1,k}$ holds. Then we have $F \in M^{1,k}\big[U^1|_o \setminus (D_{j-1}^1 \setminus A^0), U^1|_o \cup A^0 \,\vdots\, \Delta_{j-1}^1|_d\big]$. Assume for the sake of a contradiction that $F \notin D^1$ holds, then Definitions 10 and 11 ensure that F is derived in line 169, which is a contradiction. Consequently, $F \in D^1$ holds, as required. □

Claim 48. *Property (b) holds for $s = 1$.*

Proof. Assume that each module function call made before line 159 finishes for stratum 1 is correct in the context of $E|_n$ and Π. We show properties (B.11)–(B.12) for $s = 1$.

For the $\subseteq$ direction of (B.11), consider arbitrary $F \in A^1 \setminus D^0 = A^1$. Now if $F \in A_1^1$ holds, then clearly F belongs to the right-hand side of (B.11). Otherwise, F is added to A via Δ in line 186. But then, the way Δ is constructed in line 184 implies $F \in \bigcup_{1 \leq j \leq m(1), 1 \leq k \leq n(1)} \Delta_j^{1,k}|_a$. The choice of F is arbitrary, so the $\subseteq$ direction holds.

For the $\supseteq$ direction of (B.11), consider arbitrary $F \in A_1^1 \cup \bigcup_{1 \leq j \leq m(1), 1 \leq k \leq n(1)} \Delta_j^{1,k}|_a$. If $F \in A_1^1$ holds, then the definition of A^1 ensures $F \in A^1$. Otherwise, lines 184 and 186 ensure that $F \in A^1$ holds. The choice of F is arbitrary, so the $\supseteq$ direction holds as well.

To see that the $\subseteq$ direction of (B.12) holds for $s = 1$, we prove by induction on i that (B.15) holds.

$$U_i^1|_n \setminus I^1|_d \subseteq A^1 \tag{B.15}$$

For the base case where $i = 0$, consider arbitrary $F \in U_0^1|_n \setminus I^1|_d$. By definition we have $U_0^1|_n = U^0|_n \cup (E|_n \cap \mathsf{O}^1) = E|_n \cap \mathsf{O}^1$, and so $F \in E|_n \cap \mathsf{O}^s$ holds. There are two possibilities, which we discuss below.

- $F \in (E \setminus E^-) \cap \mathsf{O}^1$ holds. Then, we clearly have $F \in I$, which together with $F \notin I^1|_d$ implies $F \in D^1$. Consequently, lines 177 and 178 ensure $F \in A^1$, as required.
- $F \in E^+ \cap \mathsf{O}^1$ holds. Then, $F \notin I^1|_d$, line 177, and line 178 jointly imply $F \in A^1$.

The choice of F is arbitrary, so the base case holds.

For the inductive step, assume that $U_{i-1}^1|_n$ with $i > 0$ satisfies (B.15), and consider arbitrary $F \in U_i^1|_n \setminus I^1|_d$. If $F \in U_{i-1}^1|_n$ holds, then by the induction assumption we have $F \in A^1$. Otherwise, $F \in \Pi^1\big[U_{i-1}^1|_n\big]$ holds, so there exist a rule $r \in \Pi^1$ with only positive body atoms and a substitution σ such that $\mathsf{b}^+(r)\sigma \subseteq U_{i-1}^1|_n$ holds. We discuss the following possibilities.

- $\mathsf{b}^+(r)\sigma \subseteq I^1|_d = I \setminus D^1$ holds. Then, $F = \mathsf{h}(r\sigma)$ implies $F \in I$, which together with $F \notin I^1|_d$ implies $F \in D^1$. Let k be the index such that $r \in M^{1,k}$ holds. We have $F \in M^{1,k}\big[I \setminus (D^1 \setminus A^0), I \cup A^0\big] \cap (D^1 \setminus A^0)$. Consequently, the fact that Red call $C_{\ell(1)+2}^{1,k}$ is correct in the context of $E|_n$ and Π and Definition 11 jointly imply that F is derived in line 176. Then, line 178 ensures that $F \in A^1$ holds, as required.

- $\mathsf{b}^+(r)\sigma \not\subseteq I^1|_d$ holds. $\mathsf{b}^+(r)\sigma \subseteq U^1_{i-1}|_n \subseteq (I \setminus D^1) \cup A^1$ follows from the induction assumption for $i-1$. Let j be the smallest index such that $\mathsf{b}^+(r)\sigma \subseteq (I \setminus D^1) \cup A^1_j$ holds. There are two cases.
  - $j = 1$ holds. Then, $\mathsf{b}^+(r)\sigma \subseteq (I \setminus D^1) \cup A^1_1$ and $\mathsf{b}^+(r)\sigma \not\subseteq I \setminus D^1$ jointly imply $\mathsf{b}^+(r)\sigma \cap A^1_1 \neq \emptyset$. Let k be the index such that $r \in M^{1,k}$ holds. Lines 178, 180, and 182 jointly imply $F \in M^{1,k}\Big[I|^{1,k}_{\ell(1)+3} \colon \Delta^{\mathsf{p}}|^{1,k}_{\ell(1)+3} \cup \Delta^{\mathsf{m}}|^{1,k}_{\ell(1)+3}, \Delta^{\mathsf{n}}|^{1,k}_{\ell(1)+3}\Big]$. Assume for the sake of a contradiction that $F \notin A^1$ holds. Then, $F \notin I \setminus D^1$ implies $F \notin I|^{1,k}_{\ell(1)+3}$. Consequently, Definitions 9 and 11, and the fact that call $C^{1,k}_{\ell(1)+3}$ is correct in the context of $E|_n$ and Π jointly imply that F is derived in line 182, and so $F \in A^1$ holds due to lines 184 and 186, which is a contradiction. Hence, $F \in A^1$ holds, as required.
  - $j >= 2$ holds. j is the smallest such index, so $\mathsf{b}^+(r)\sigma \not\subseteq (I \setminus D^1) \cup A^1_{j-1}$ holds. Then, $\mathsf{b}^+(r)\sigma \cap \Delta^1_{j-1}|_a = \mathsf{b}^+(r)\sigma \cap (A^1_j \setminus A^1_{j-1}) \neq \emptyset$ follows from the way A is updated in line 186. Consequently, lines 184, 188, and 190 jointly imply $F \in M^{1,k}\Big[I|^{1,k}_{j+\ell(1)+2} \colon \Delta^{\mathsf{p}}|^{1,k}_{j+\ell(1)+2} \cup \Delta^{\mathsf{m}}|^{1,k}_{j+\ell(1)+2}, \Delta^{\mathsf{n}}|^{1,k}_{j+\ell(1)+2}\Big]$. In a similar way as in the previous paragraph, we have $F \in A^1$, as required.

The choice of F is arbitrary, and so this completes the inductive step.

To see that the $\supseteq$ direction of (B.12) holds for $s = 1$, we prove by induction on j that (B.16) holds.

$$U^1|_n \setminus I^1|_d \supseteq A^1_j \tag{B.16}$$

For the base case where $j = 1$, consider arbitrary $F \in A^1_1$. Then, F is added to A in line 178. We discuss the following possibilities.

- $F \in (E \setminus E^-) \cap (D^1 \cap \mathsf{O}^1)$ holds. Then, $F \in E|_n \cap \mathsf{O}^1 = U^1_0|_n \subseteq U^1|_n$ holds. Moreover, $F \in D^1$ ensures $F \notin I^1|_d$. Consequently, we have $F \in U^1|_n \setminus I^1|_d$, as required.
- $F \in (E^+ \cap \mathsf{O}^1) \setminus (I \setminus D^1)$ holds. Then, $F \in E^+ \cap \mathsf{O}^1$ implies $F \in U^1_0|_n \subseteq U^1|_n$. Moreover, $F \notin I \setminus D^1$ ensures $F \notin I^1|_d$. Consequently, $F \in U^1|_n \setminus I^1|_d$ holds.

- There exists k with $1 \leq k \leq n(1)$ such that $F \in J|^{1,k}_{\ell(1)+2}$ holds. Red call $C^{1,k}_{\ell(1)+2}$ is correct in the context of $E|_n$ and Π, so by Definition 11 we have $F \in \mathsf{mat}(\Pi, E|_n) \cap D^1$. But then, $D^1 \subseteq \mathsf{O}^1$ implies $F \in \mathsf{mat}(\Pi, E|_n) \cap \mathsf{O}^1 = U^1|_n$. Moreover, $F \in D^1$ implies $F \notin I^1|_d$. Consequently, we have $F \in U^1|_n \setminus I^1|_d$, as required.

For the inductive step, assume that A^1_{j-1} with $2 \leq j \leq m(1)$ satisfies (B.16), and consider arbitrary $F \in A^1_j$. If $F \in A^1_{j-1}$, then by the induction assumption $F \in U^1|_n \setminus I^1|_d$ clearly holds. Otherwise we have $F \in A^1_j \setminus A^1_{j-1}$. There are two cases.

- $j = 2$ holds. Then, F is derived in line 182 and added to A via Δ in line 186. Consequently, there exists k with $1 \leq k \leq n(1)$ such that $F \in J|^{1,k}_{\ell(1)+3}$ holds. Add call $C^{1,k}_{\ell(1)+3}$ is correct in the context of $E|_n$ and Π, so $F \in M^{1,k\uparrow}_{\infty}[(I \setminus D^1) \cup A^1_1 \colon A^1_1, D^1]$ holds. Due to stratification we have $F \in M^{1,k\uparrow}_{\infty}[(I \setminus D^1) \cup A^1_1 \colon A^1_1]$. Let Δ_i and J_i be defined with regards to $M^{1,k\uparrow}_{\infty}[(I \setminus D^1) \cup A^1_1 \colon A^1_1]$ in the same way as in Definition 9. We show by induction on i that $\Delta_i \subseteq U^1|_n \setminus I^1|_d$ and $J_i \subseteq U^1|_n \setminus I^1|_d$ hold for $i \geq 1$, which then implies $F \in U^1|_n \setminus I^1|_d$.
  - For the induction base, we have $\Delta_1 = M^{1,k}\Big[(I \setminus D^1) \cup A^1_1 \colon A^1_1\Big] \setminus ((I \setminus D^1) \cup A^1_1)$. Then, we clearly have $\Delta_1 \cap I^1|_d = \Delta_1 \cap (I \setminus D^1) = \emptyset$. Moreover, $M^{1,k} \subseteq \Pi^1$ implies $\Delta_1 \subseteq \Pi^1\Big[(U^1|_o \setminus D^1) \cup A^1_1\Big]$, which together with property (B.10) for $s = 1$ and $A^1_1 \subseteq U^1|_n$ implies $\Delta_1 \subseteq \Pi^1\Big[U^1|_n\Big] \subseteq U^1|_n$. Consequently, we have $\Delta_1 \subseteq U^1|_n \setminus I^1|_d$. Furthermore, $J_1 = \Delta_1 \subseteq U^1|_n \setminus I^1|_d$ holds.
  - For the inductive step, consider arbitrary $i > 1$ such that $\Delta_{i-1} \subseteq U^1|_n \setminus I^1|_d$ and $J_{i-1} \subseteq U^1|_n \setminus I^1|_d$ hold. $\Delta_i \subseteq M^{1,k}\Big[U^1|_n \cup J_{i-1}\Big] \setminus ((I \setminus D^1) \cup A^1_1 \cup J_{i-1})$ follows from Definition 9. But then, the induction assumption $J_{i-1} \subseteq U^1|_n \setminus I^1|_d$ implies $\Delta_i \subseteq U^1|_n \setminus I^1|_d$. Moreover, $J_i = J_{i-1} \cup \Delta_i \subseteq U^1|_n \setminus I^1|_d$ holds, as required.
- $3 \leq j \leq m(1)$ holds. Then, F is derived in line 190 and added to A via Δ in line 186. Consequently, there exists k with $1 \leq k \leq n(1)$ such that $F \in J|^{1,k}_{j+\ell(1)+1}$ holds. Add call $C^{1,k}_{j+\ell(1)+1}$ is correct, so we have $F \in M^{1,k\uparrow}_{\infty}[(I \setminus D^1) \cup A^1_{j-1} \colon \Delta^1_{j-2}|_a]$. Let Δ_i and J_i be defined with regards to $M^{1,k\uparrow}_{\infty}[(I \setminus D^1) \cup A^1_{j-1} \colon \Delta^1_{j-2}|_a]$ in the same way as in Definition 9. We show by induction on i that $\Delta_i \subseteq U^1|_n \setminus I^1|_d$ and $J_i \subseteq U^1|_n \setminus I^1|_d$ hold for $i \geq 1$, which then implies $F \in U^1|_n \setminus I^1|_d$.

– For the base case, $\Delta_1 = M^{1,k}\Big[(I \setminus D^1) \cup A^1_{j-1} \vdots \Delta^1_{j-2}|_a\Big] \setminus ((I \setminus D^1) \cup A^1_{j-1})$ holds. Then, we clearly have $\Delta_1 \cap I^1|_d = \Delta_1 \cap (I \setminus D^1) = \emptyset$. Moreover, $M^{1,k} \subseteq \Pi^1$ implies $\Delta_1 \subseteq \Pi^1\Big[(U^1|_o \setminus D^1) \cup A^1_{j-1}\Big]$, which together with property (B.10) for $s = 1$ and the induction assumption $A^1_{j-1} \subseteq U^1|_n$ implies $\Delta_1 \subseteq \Pi^1\Big[U^1|_n\Big] \subseteq U^1|_n$. Consequently, we have $\Delta_1 \subseteq U^1|_n \setminus I^1|_d$ and $J_1 = \Delta_1 \subseteq U^1|_n \setminus I^1|_d$.

– For the inductive step, consider arbitrary $i > 1$ such that $\Delta_{i-1} \subseteq U^1|_n \setminus I^1|_d$ and $J_{i-1} \subseteq U^1|_n \setminus I^1|_d$ hold. $\Delta_i \subseteq M^{1,k}\Big[U^1|_n \cup J_{i-1}\Big] \setminus ((I \setminus D^1) \cup A^1_{j-1} \cup J_{i-1})$ follows from Definition 9. But then, the induction assumption $J_{i-1} \subseteq U^1|_n \setminus I^1|_d$ implies $\Delta_i \subseteq U^1|_n \setminus I^1|_d$. Moreover, $J_i = J_{i-1} \cup \Delta_i \subseteq U^1|_n \setminus I^1|_d$ holds, as required.

The choice of F is arbitrary, so $A^1_j \subseteq U^1|_n \setminus I^1|_d$ holds, and this completes the inductive proof for (B.16). □

Claim 49. *Property (c) holds for $s = 1$.*

Proof. Assume that each module function call made before line 159 finishes for stratum 1 is correct in the context of $E|_n$ and Π. We show that property (B.13) holds for $s = 1$.

For the $\subseteq$ direction, Claim 47 and $A^0 = \emptyset$ jointly imply $U^1|_o \setminus D^1 \subseteq U^1|_n$; moreover, Claim 48 and $D^0 = \emptyset$ jointly imply $A^1 \subseteq U^1|_n$. Hence, $(U^1|_o \setminus D^1) \cup A^1 \subseteq U^1|_n$ holds.

For the $\supseteq$ direction, consider arbitrary $F \in U^1|_n$. There are two possibilities.

- $F \notin I^1|_d$ holds. Then, Claim 48 implies $F \in A^1 \subseteq (U^1|_o \setminus D^1) \cup A^1$.
- $F \in I^1|_d = I \setminus (D^1 \setminus A^0) = I \setminus D^1$ holds. Then, $F \in U^1|_n$ implies $F \in \mathsf{O}^1$, so $F \in U^1|_o \setminus D^1 \subseteq (U^1|_o \setminus D^1) \cup A^1$ holds.

The choice of F is arbitrary, so the $\supseteq$ direction holds as well. □

Claim 50. *For each s with $2 \leq s \leq S$, if properties (a)–(c) hold for $s - 1$, then property (a) holds for s as well.*

Proof. Consider arbitrary s with $2 \leq s \leq S$ such that properties (a)–(c) hold for $s - 1$, and assume that each module function call made before line 159 finishes for stratum s is correct in the context of $E|_n$ and Π. We show that properties (B.9)–(B.10) hold for s.

For the $\subseteq$ direction of property (B.9), consider arbitrary $F \in D^s \setminus A^{s-1}$. If $F \in D^{s-1}$ holds, then clearly F belongs to the right-hand side of (B.9). Otherwise, F is added to D via Δ in line 170. The way Δ is constructed in in line 164 and line 171 implies $F \in (E^- \cap \mathsf{O}^s) \cup \bigcup_{1 \leq j \leq \ell(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}|_d$.

For the $\supseteq$ direction of property (B.9), consider arbitrary F that belongs to the right-hand side of (B.9). There are several cases.

- $F \in D^{s-1} \setminus A^{s-1}$ holds. Then, $D^{s-1} \subseteq D^s$ implies $F \in D^s \setminus A^{s-1}$.
- $F \in (E^- \cap \mathsf{O}^s) \cup \bigcup_{1 \leq k \leq n(s)} \Delta_1^{s,k}|_d$ holds. Then, lines 164 and 170 ensures that $F \in D^s$ holds; moreover, each call for a module that belongs to stratum s only produces facts in O^s, so $F \notin A^{s-1}$ holds. Hence, we have $F \in D^s \setminus A^{s-1}$.
- $F \in \bigcup_{2 \leq j \leq \ell(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}|_d$ holds. Then, lines 171 and 170 ensure $F \in D^s \setminus A^{s-1}$.

The choice of F is arbitrary, so the $\supseteq$ direction holds as well.

To see that (B.10) holds for s, we prove by induction on i that (B.17) holds.

$$U_i^s|_o \setminus U^s|_n \subseteq D^s \setminus A^{s-1} \tag{B.17}$$

For the base case where $i = 0$, consider arbitrary $F \in U_0^s|_o \setminus U^s|_n$. By defnition we have $U_0^s|_o = U^{s-1}|_o \cup (E|_o \cap \mathsf{O}^s)$. There are two cases.

- $F \in U^{s-1}|_o$ holds. Then, the induction assumption for $s-1$ implies $F \in D^{s-1} \setminus A^{s-2}$, so $F \in D^s$ holds. Moreover, $F \notin U^s|_n$ ensures $F \notin U^{s-1}|_n$, which together with property (B.13) for $s-1$ implies $F \notin A^{s-1}$. Consequently, we have $F \in D^s \setminus A^{s-1}$.
- $F \in E|_o \cap \mathsf{O}^s$ holds. Then, $F \notin U^s|_n$ implies $F \notin E|_n \cap \mathsf{O}^s$. Consequently, we have $F \in E^- \cap \mathsf{O}^s \subseteq D^s$. Moreover, $F \notin U^s|_n$ implies $F \notin A^{s-1}$ in the same way as in the previous case. Hence, $F \in D^s \setminus A^{s-1}$ holds.

The choice of F is arbitrary, so the base case holds.

For the inductive step, assume that $U_{i-1}^s|_o$ satisfies (B.14) for $i > 0$, and consider arbitrary $F \in U_i^s|_o \setminus U^s|_n$. If $F \in U_{i-1}^s|_o$, then by the induction assumption $F \in D^s \setminus A^{s-1}$ holds. Otherwise, we have $F \in \Pi^s\left[U_{i-1}^s|_o\right]$, so there exist a rule $r \in \Pi^s$ and a substitution σ such that $\mathsf{b}^+(r)\sigma \subseteq U_{i-1}^s|_o$ and $\mathsf{b}^-(r)\sigma \cap U_{i-1}^s|_o = \emptyset$ hold. Moreover, $F \notin U^s|_n$ implies

$F \notin \Pi^s\big[U^s|_n\big]$, and so $\mathsf{b}^+(r)\sigma \not\subseteq U^s|_n$ or $\mathsf{b}^-(r)\sigma \cap U^s|_n \neq \emptyset$ holds. There are two possibilities, which we discuss below.

- $\mathsf{b}^-(r)\sigma \cap U^s|_n \neq \emptyset$ holds. Then, $\mathsf{b}^-(r)\sigma \cap U^{s-1}|_n \neq \emptyset$ holds due to stratification. Similarly, $\mathsf{b}^-(r)\sigma \cap U^s_{i-1}|_o = \emptyset$ implies $\mathsf{b}^-(r)\sigma \cap U^{s-1}|_o = \emptyset$. Hence, there exists $G \in \mathsf{b}^-(r)\sigma$ such that $G \in U^{s-1}|_n \setminus U^{s-1}|_o$ holds. $G \in U^{s-1}|_n$ implies $G \in \mathsf{O}^{\leq s-1}$, which together with $G \notin U^{s-1}|_o$ ensures $G \notin I$. Hence, $G \notin D^{s-1}$ holds. Moreover, property (B.13) for $s-1$ implies $G \in A^{s-1}$. Therefore, we have $G \in A^{s-1} \setminus D^{s-1}$. Let k be the index such that $r \in M^{s,k}$ holds. $F \in M^{s,k}\big[U^s|_o, U^s|_o \colon D^{s-1} \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}\big]$ clearly holds; moreover, $F \in U^s|_o \setminus U^s_{i-1}|_o$ implies $F \in \mathsf{O}^s$, which in turn implies $F \notin D^{s-1}$. Consequently, $F \in M^{s,k}{\downarrow}[U^s|_o, U^s|_o \colon D^{s-1} \setminus A^{s-1}, A^{s-1} \setminus D^{s-1}]$ follows from Definition 10. Furthermore, $F \notin U^s|_n$ implies $F \notin \mathsf{mat}(\Pi, E|_n)$. Therefore, Definition 11 and the fact that each call is correct in the context of $E|_n$ and Π ensure that F is derived in line 163, and so $F \in D^s$ holds.
- $\mathsf{b}^-(r)\sigma \cap U^s|_n = \emptyset$ and $\mathsf{b}^+(r)\sigma \not\subseteq U^s|_n$ both hold. Then, $\mathsf{b}^+(r)\sigma \subseteq U^s_{i-1}|_o$ implies $\mathsf{b}^+(r)\sigma \cap (U^s_{i-1}|_o \setminus U^s|_n) \neq \emptyset$, and so the induction assumption for $i-1$ implies $\mathsf{b}^+(r)\sigma \cap (D^s \setminus A^{s-1}) \neq \emptyset$. Now let j be the smallest index such that $\mathsf{b}^+(r)\sigma \cap (D^s_j \setminus A^{s-1}) \neq \emptyset$ holds, and we discuss the following two cases.
  - $j = 1$ holds. Then, we have $\mathsf{b}^+(r)\sigma \cap (D^{s-1} \setminus A^{s-1}) \neq \emptyset$. In the same way as in the previous paragraph, F will be derived in line 163, and so $F \in D^s$ holds.
  - $j \geq 2$ holds. Since j is the smallest such index, we have $\mathsf{b}^+(r)\sigma \cap (D^s_{j-1} \setminus A^{s-1}) = \emptyset$, which together with $\mathsf{b}^+(r)\sigma \subseteq U^s_{i-1}|_o$ implies $\mathsf{b}^+(r)\sigma \subseteq U^s|_o \setminus (D^s_{j-1} \setminus A^{s-1})$. Moreover, $\mathsf{b}^+(r)\sigma \cap (D^s_{j-1} \setminus A^{s-1}) = \emptyset$ and $\mathsf{b}^+(r)\sigma \cap (D^s_j \setminus A^{s-1}) \neq \emptyset$ jointly imply $\mathsf{b}^+(r)\sigma \cap \Delta^s_{j-1}|_d \neq \emptyset$. Furthermore, $\mathsf{b}^-(r)\sigma \cap U^s_{i-1}|_o = \emptyset$, $\mathsf{b}^-(r)\sigma \cap U^s|_n = \emptyset$, and property (B.13) for $s-1$ jointly imply $\mathsf{b}^-(r)\sigma \cap (U^{s-1}|_o \cup A^{s-1}) = \emptyset$, which in turn implies $\mathsf{b}^-(r)\sigma \cap (U^s|_o \cup A^s) = \emptyset$. Let k be the index such that $r \in M^{s,k}$ holds. Then we have $F \in M^{s,k}\big[U^s|_o \setminus (D^s_{j-1} \setminus A^{s-1}), U^s|_o \cup A^s \colon \Delta^s_{j-1}|_d\big]$. Assume for the sake of a contradiction that $F \notin D^s$ holds, then Definitions 10 and 11 ensure that F is derived in line 169, which is a contradiction. Consequently, $F \in D^s$ holds, as required.

Finally, $F \notin U^s|_n$ ensures $F \notin U^{s-1}|_n$, which together with property (B.13) for $s-1$ implies $F \notin A^{s-1}$. Consequently, we have $F \in D^s \setminus A^{s-1}$, as required. The choice of F is arbitrary, so (B.17) holds for s as well, and this completes the inductive step. □

Claim 51. *For each s with $2 \leq s \leq S$, if properties (a)–(c) hold for $s-1$, then property (b) holds for s as well.*

Proof. Consider arbitrary s with $2 \leq s \leq S$ such that properties (a)–(c) hold for $s-1$, and assume that each module function call made before line 159 finishes for stratum s is correct in the context of $E|_n$ and Π. We show that properties (B.11)–(B.12) hold for s.

For the $\subseteq$ direction of (B.11), consider arbitrary $F \in A^s \setminus D^{s-1}$. If $F \in A_1^s$ holds, then clearly F belongs to the right-hand side of (B.11). Otherwise, F is added to A via Δ in line 186. But then, the way Δ is constructed in line 184 implies $F \in \bigcup_{1 \leq j \leq m(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}|_a$.

For the $\supseteq$ direction, consider arbitrary $F \in (A_1^s \setminus D^{s-1}) \cup \bigcup_{1 \leq j \leq m(s), 1 \leq k \leq n(s)} \Delta_j^{s,k}|_a$. If $F \in A_1^s \setminus D^{s-1}$ holds, then the definition of A^s ensures $F \in A^s \setminus D^{s-1}$. Otherwise, lines 184 and 186 ensure that $F \in A^s$ holds; moreover, we clearly have $F \in \mathsf{O}^s$, which implies $F \notin D^{s-1}$. The choice of F is arbitrary, so the $\supseteq$ direction holds as well.

To see that the $\subseteq$ direction of (B.12) holds for s, we prove (B.18) by induction on i.

$$U_i^s|_n \setminus I^s|_d \subseteq A^s \setminus D^{s-1} \tag{B.18}$$

For the base case where $i = 0$, consider arbitrary $F \in U_0^s|_n \setminus I^s|_d$. By definition we have $U_0^s|_n = U^{s-1}|_n \cup (E|_n \cap \mathsf{O}^s)$. There are several possibilities.

- $F \in U^{s-1}|_n$ holds. Then, property (B.13) for $s-1$ implies $F \in (U^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1}$. Now $D^s \setminus D^{s-1} \subseteq \mathsf{O}^s$ holds, so we have $F \in (U^{s-1}|_o \setminus D^s) \cup A^{s-1} \subseteq (I \setminus D^s) \cup A^{s-1}$. Together with $F \notin I \setminus (D^s \setminus A^{s-1})$ this implies $F \in A^{s-1}$. Assume for the sake of a contradiction that $F \in D^s$ holds, then $D^s \subseteq I$ and $F \in A^{s-1}$ jointly imply $F \in I \setminus (D^s \setminus A^{s-1})$, which is a contradiction. Hence, $F \in A^{s-1} \setminus D^s \subseteq A^s \setminus D^{s-1}$ holds, as required.

- $F \in (E \setminus E^-) \cap \mathsf{O}^s$ holds. Then, we clearly have $F \in I$, which together with $F \notin I^s|_d$ implies $F \in D^s \setminus A^{s-1} \subseteq D^s$. Consequently, lines 177 and 178 ensure $F \in A^s$. Moreover, $F \in \mathsf{O}^s$ implies $F \notin D^{s-1}$, so we have $F \in A^s \setminus D^{s-1}$, as required.
- $F \in E^+ \cap \mathsf{O}^s$ holds. Then, $F \notin I^s|_d$, line 177, and line 178 jointly imply $F \in A^s$. Moreover, $F \in \mathsf{O}^s$ implies $F \notin D^{s-1}$. Hence, $F \in A^s \setminus D^{s-1}$ holds, as required.

The choice of F is arbitrary, so the base case holds.

For the inductive step, assume that $U^s_{i-1}|_n$ with $i > 0$ satisfies (B.18), and consider arbitrary $F \in U^s_i|_n \setminus I^s|_d$. If $F \in U^s_{i-1}|_n$ holds, then by the induction assumption we have $F \in A^s \setminus D^{s-1}$. Otherwise, $F \in \Pi^s\left[U^s_{i-1}|_n\right]$ holds, so there exist a rule $r \in \Pi^s$ and a substitution σ such that $\mathsf{b}^+(r)\sigma \subseteq U^s_{i-1}|_n$ and $\mathsf{b}^-(r)\sigma \cap U^s_{i-1}|_n = \emptyset$ hold. By the induction assumption for (B.18) we have $\mathsf{b}^+(r)\sigma \subseteq (I \setminus (D^s \setminus A^{s-1})) \cup (A^s \setminus D^{s-1}) \subseteq (I \setminus D^s) \cup A^s$. Moreover, by property (B.13) for $s-1$ we have $\mathsf{b}^-(r)\sigma \cap ((I \setminus D^{s-1}) \cup A^{s-1}) = \emptyset$. We discuss the following possibilities.

- $\mathsf{b}^+(r)\sigma \subseteq I^s|_d = I \setminus (D^s \setminus A^{s-1})$ and $\mathsf{b}^-(r)\sigma \cap (I \cup A^{s-1}) = \emptyset$ hold. Then, $F = \mathsf{h}(r\sigma)$ implies $F \in I$, which together with $F \notin I^s|_d$ implies $F \in D^s \setminus A^{s-1}$. Let k be the index such that $r \in M^{s,k}$ holds. We have $F \in M^{s,k}\left[I \setminus (D^s \setminus A^{s-1}), I \cup A^{s-1}\right] \cap (D^s \setminus A^{s-1})$. Consequently, the fact that Red call $C^{s,k}_{\ell(s)+2}$ is correct in the context of $E|_n$ and Π and Definition 11 jointly imply that F is derived in line 176. Then, line 178 ensures that $F \in A^s$ holds.
- $\mathsf{b}^+(r)\sigma \subseteq I^s|_d$ and $\mathsf{b}^-(r)\sigma \cap (I \cup A^{s-1}) \neq \emptyset$ hold. Then, $\mathsf{b}^-(r)\sigma \cap ((I \setminus D^{s-1}) \cup A^{s-1}) = \emptyset$ implies $\mathsf{b}^-(r)\sigma \cap (D^{s-1} \setminus A^{s-1}) \neq \emptyset$, so $F \in M^{s,k}\left[I|^{s,k}_{\ell(s)+3} \vdots \Delta^{\mathsf{p}}|^{s,k}_{\ell(s)+3} \cup \Delta^{\mathsf{m}}|^{s,k}_{\ell(s)+3}, \Delta^{\mathsf{n}}|^{s,k}_{\ell(s)+3}\right]$ holds. Assume for the sake of a contradiction that $F \notin A^s$ holds. Then, $F \notin I \setminus D^s$ implies $F \notin I|^{s,k}_{\ell(s)+3}$. Consequently, Definitions 9 and 11, and the fact that call $C^{s,k}_{\ell(s)+3}$ is correct in the context of $E|_n$ and Π jointly imply that F is derived in line 182, and so $F \in A^s$ holds due to lines 184 and 186, which is a contradiction. Hence, $F \in A^s$ holds, as required.
- $\mathsf{b}^+(r)\sigma \not\subseteq I^1|_d$ holds. Let j be the smallest index such that $\mathsf{b}^+(r)\sigma \subseteq (I \setminus D^s) \cup A^s_j$ holds. There are two cases.

– $j = 1$ holds. Then, $\mathsf{b}^+(r)\sigma \subseteq (I \setminus D^s) \cup A_1^s$ and $\mathsf{b}^+(r)\sigma \not\subseteq (I \setminus (D^s \setminus A^{s-1}))$ jointly imply $\mathsf{b}^+(r)\sigma \cap A_1^s \neq \emptyset$. Assume for the sake of a contradiction that $\mathsf{b}^+(r)\sigma \cap A_1^s \subseteq D^{s-1}$ holds, which implies $\mathsf{b}^+(r)\sigma \cap A_1^s = \mathsf{b}^+(r)\sigma \cap A^{s-1} \subseteq \mathsf{O}^{\leq s-1}$. Then, together with $D^{s-1} \subseteq D^s \subseteq I$ we have $\mathsf{b}^+(r)\sigma \cap A_1^s \subseteq I \setminus (D^s \setminus A^{s-1})$, which is a contradiction. Hence, we have $\mathsf{b}^+(r)\sigma \cap (A_1^s \setminus D^{s-1}) \neq \emptyset$. Let k be the index such that $r \in M^{s,k}$ holds. Lines 178, 180, and 182 jointly imply $F \in M^{s,k}\Big[I|^{s,k}_{\ell(s)+3} \colon \Delta^{\mathsf{p}}|^{s,k}_{\ell(s)+3} \cup \Delta^{\mathsf{m}}|^{s,k}_{\ell(s)+3}, \Delta^{\mathsf{n}}|^{s,k}_{\ell(s)+3}\Big]$. Assume for the sake of a contradiction that $F \notin A^s$ holds. Then, $F \notin I \setminus D^s$ implies $F \notin I|^{s,k}_{\ell(s)+3}$. Consequently, Definitions 9 and 11, and the fact that call $C^{s,k}_{\ell(s)+3}$ is correct in the context of $E|_n$ and Π jointly imply that F is derived in line 182, and so $F \in A^s$ holds due to lines 184 and 186, which is a contradiction. Hence, $F \in A^s$ holds, as required.

– $j >= 2$ holds. Since j is the smallest such index, $\mathsf{b}^+(r)\sigma \not\subseteq (I \setminus D^s) \cup A^s_{j-1}$ holds. Then, $\mathsf{b}^+(r)\sigma \cap \Delta^s_{j-1}|_a = \mathsf{b}^+(r)\sigma \cap (A^s_j \setminus A^s_{j-1}) \neq \emptyset$ follows from the way A is updated in line 186. Consequently, lines 184, 188, and 190 jointly imply $F \in M^{s,k}\Big[I|^{s,k}_{j+\ell(s)+2} \colon \Delta^{\mathsf{p}}|^{s,k}_{j+\ell(s)+2} \cup \Delta^{\mathsf{m}}|^{s,k}_{j+\ell(s)+2}, \Delta^{\mathsf{n}}|^{s,k}_{j+\ell(s)+2}\Big]$. In a similar way as in the previous paragraph, we have $F \in A^s$, as required.

Finally, $F \in \Pi^s\Big[U^s_{i-1}|_n\Big]$ implies $F \notin D^{s-1}$. Hence, $F \in A^s \setminus D^{s-1}$ holds. The choice of F is arbitrary, and so this completes the inductive step.

To see that the $\supseteq$ direction of (B.12) holds for s, we prove by induction on j that (B.19) holds.

$$U^s|_n \setminus I^s|_d \supseteq A^s_j \setminus D^{s-1} \tag{B.19}$$

For the base case where $j = 1$, consider arbitrary $F \in A^s_1 \setminus D^{s-1}$. Then, F either belongs to $A^{s-1} \setminus D^{s-1}$ or is added to A in line 178. We discuss the following possibilities.

- $F \in A^{s-1} \setminus D^{s-1}$ holds. Then, $D^{s-2} \subseteq D^{s-1}$ implies $F \in A^{s-1} \setminus D^{s-2}$. By property (B.12) for $s-1$ we have $F \in U^{s-1}|_n \setminus I^{s-1}|_d$. Now $U^{s-1}|_n \subseteq U^s|_n$ implies $F \in U^s|_n$. Assume for the sake of a contradiction that $F \in I^s|_d$ holds. Then, we have

$F \in I$, which together with $F \notin I^{s-1}|_d$ implies $F \in D^{s-1} \setminus A^{s-2} \subseteq D^{s-1}$. This is a contradiction. Hence, $F \in U^s|_n \setminus I^s|_d$ holds, as required.

- $F \in (E \setminus E^-) \cap (D^s \cap \mathsf{O}^s)$ holds. Then, $F \in E|_n \cap \mathsf{O}^s \subseteq U^s|_n$ holds. Moreover, $F \in D^s \cap \mathsf{O}^s$ implies $F \in D^s \setminus A^{s-1}$, which in turn implies $F \notin I^s|_d$. Consequently, we have $F \in U^s|_n \setminus I^s|_d$, as required.
- $F \in (E^+ \cap \mathsf{O}^s) \setminus I^s|_d$ holds. Then, $F \in E^+ \cap \mathsf{O}^s$ implies $F \in U^s_0|_n \subseteq U^s|_n$. Hence, $F \in U^s|_n \setminus I^s|_d$ holds.
- There exists k with $1 \leq k \leq n(s)$ such that $F \in J|^{s,k}_{\ell(s)+2}$ holds. Red call $C^{s,k}_{\ell(s)+2}$ is correct in the context of $E|_n$ and Π, so by Definition 11 we have $F \in \mathsf{mat}(\Pi, E|_n) \cap (D^s \setminus A^{s-1})$. But then, $D^s \subseteq \mathsf{O}^{\leq s}$ implies $F \in \mathsf{mat}(\Pi, E|_n) \cap \mathsf{O}^{\leq s} = U^s|_n$. Moreover, $F \in D^s \setminus A^{s-1}$ implies $F \notin I^s|_d$. Consequently, we have $F \in U^s|_n \setminus I^s|_d$, as required.

For the inductive step, assume that A^s_{j-1} with $2 \leq j \leq m(s)$ satisfies (B.19), and consider arbitrary $F \in A^s_j$. If $F \in A^s_{j-1}$, then by the induction assumption $F \in U^s|_n \setminus I^s|_d$ clearly holds. Otherwise we have $F \in A^s_j \setminus A^s_{j-1}$. There are two cases.

- $j = 2$ holds. Then, F is derived in line 182 and added to A via Δ in line 186. Consequently, there exists k with $1 \leq k \leq n(s)$ such that $F \in J|^{s,k}_{\ell(s)+3}$ holds. Add call $C^{s,k}_{\ell(s)+3}$ is correct, so we have $F \in M^{s,k\uparrow}_{\infty}[(I \setminus D^s) \cup A^s_1 \colon A^s_1 \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}]$. Let Δ_i and J_i be defined with regards to $M^{s,k\uparrow}_{\infty}[(I \setminus D^s) \cup A^s_1 \colon A^s_1 \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}]$ in the same way as in Definition 9. We show by induction on i that $\Delta_i \subseteq U^s|_n \setminus I^s|_d$ and $J_i \subseteq U^s|_n \setminus I^s|_d$ hold for $i \geq 1$, which then implies $F \in U^s|_n \setminus I^s|_d$.
  - For the induction base, we have $\Delta_1 \subseteq M^{s,k}\Big[(I \setminus D^s) \cup A^s_1 \colon A^s_1 \setminus D^{s-1}, D^{s-1} \setminus A^{s-1}\Big]$ and $\Delta_1 \cap ((I \setminus D^s) \cup A^s_1) = \emptyset$, so $\Delta_1 \cap I^s|_d = \Delta_1 \cap (I \setminus (D^s \setminus A^{s-1})) = \emptyset$ holds. Moreover, $M^{s,k} \subseteq \Pi^s$ implies $\Delta_1 \subseteq \Pi^s\Big[(U^s|_o \setminus D^s) \cup A^s_1\Big]$; due to stratification we have $\Delta_1 \subseteq \Pi^s\Big[(U^s|_o \setminus D^s) \cup A^s_1, (U^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1}\Big]$, which together with property (B.13) for $s-1$ implies $\Delta_1 \subseteq \Pi^s\Big[(U^s|_o \setminus D^s) \cup A^s_1, U^{s-1}|_n\Big]$; then, property (B.10) for s and the induction assumption $A^s_1 \subseteq U^s|_n$ jointly imply $\Delta_1 \subseteq \Pi^s\Big[U^s|_n, U^{s-1}|_n\Big] = \Pi^s\Big[U^s|_n\Big] \subseteq U^s|_n$. Consequently, $\Delta_1 \subseteq U^s|_n \setminus I^s|_d$ holds. Furthermore, we have $J_1 = \Delta_1 \subseteq U^s|_n \setminus I^s|_d$.

  - For the inductive step, consider arbitrary $i > 1$ such that $\Delta_{i-1} \subseteq U^s|_n \setminus I^s|_d$ and $J_{i-1} \subseteq U^s|_n \setminus I^s|_d$ hold. By Definition 9 we have $\Delta_i \subseteq M^{s,k}\Big[U^s|_n \cup J_{i-1}, U^{s-1}|_n\Big]$ and $\Delta_i \cap ((I \setminus D^s) \cup A^s_1 \cup J_{i-1}) = \emptyset$. But then, $\Delta_i \subseteq U^s|_n \setminus I^s|_d$ follows from the induction assumption $J_{i-1} \subseteq U^s|_n \setminus I^s|_d$. Moreover, $J_i = J_{i-1} \cup \Delta_i \subseteq U^s|_n \setminus I^s|_d$ holds, as required.

- $3 \leq j \leq m(s)$ holds. Then, F is derived in line 190 and added to A via Δ in line 186. Consequently, there exists k with $1 \leq k \leq n(s)$ such that $F \in J|^{s,k}_{j+\ell(s)+1}$ holds. Add call $C^{s,k}_{j+\ell(s)+1}$ is correct, so we have $F \in M^{s,k\uparrow}_{\infty}[(I \setminus D^s) \cup A^s_{j-1} \vdots \Delta^s_{j-2}|_a]$. Let Δ_i and J_i be defined with regards to $M^{s,k\uparrow}_{\infty}[(I \setminus D^s) \cup A^s_{j-1} \vdots \Delta^s_{j-2}|_a]$ in the same way as in Definition 9. We show by induction on i that $\Delta_i \subseteq U^s|_n \setminus I^s|_d$ and $J_i \subseteq U^s|_n \setminus I^s|_d$ hold for $i \geq 1$, which then implies $F \in U^s|_n \setminus I^s|_d$.

  - For the induction base, we have $\Delta_1 \subseteq M^{s,k}\Big[(I \setminus D^s) \cup A^s_{j-1} \vdots \Delta^s_{j-2}|_a\Big]$ and $\Delta_1 \cap ((I \setminus D^s) \cup A^s_{j-1}) = \emptyset$, so $\Delta_1 \cap I^s|_d = \Delta_1 \cap (I \setminus (D^s \setminus A^{s-1})) = \emptyset$ holds. Moreover, $M^{s,k} \subseteq \Pi^s$ implies $\Delta_1 \subseteq \Pi^s\Big[(U^s|_o \setminus D^s) \cup A^s_{j-1}\Big]$; due to stratification we have $\Delta_1 \subseteq \Pi^s\Big[(U^s|_o \setminus D^s) \cup A^s_{j-1}, (U^{s-1}|_o \setminus D^{s-1}) \cup A^{s-1}\Big]$, which together with property (B.13) for $s-1$ implies $\Delta_1 \subseteq \Pi^s\Big[(U^s|_o \setminus D^s) \cup A^s_{j-1}, U^{s-1}|_n\Big]$; then, property (B.10) for s and the induction assumption $A^s_{j-1} \subseteq U^s|_n$ jointly imply $\Delta_1 \subseteq \Pi^s\Big[U^s|_n, U^{s-1}|_n\Big] = \Pi^s\Big[U^s|_n\Big] \subseteq U^s|_n$. Consequently, we have $\Delta_1 \subseteq U^s|_n \setminus I^s|_d$. Furthermore, $J_1 = \Delta_1 \subseteq U^s|_n \setminus I^s|_d$ holds.

  - For the inductive step, consider arbitrary $i > 1$ such that $\Delta_{i-1} \subseteq U^s|_n \setminus I^s|_d$ and $J_{i-1} \subseteq U^s|_n \setminus I^s|_d$ hold. By Definition 9 we have $\Delta_i \subseteq M^{s,k}\Big[U^s|_n \cup J_{i-1}, U^{s-1}|_n\Big]$ and $\Delta_i \cap ((I \setminus D^s) \cup A^s_{j-1} \cup J_{i-1}) = \emptyset$. But then, the induction assumption $J_{i-1} \subseteq U^s|_n \setminus I^s|_d$ implies $\Delta_i \subseteq U^s|_n \setminus I^s|_d$. Moreover, $J_i = J_{i-1} \cup \Delta_i \subseteq U^s|_n \setminus I^s|_d$ holds, as required.

The choice of F is arbitrary, so $A^s_j \subseteq U^s|_n \setminus I^s|_d$ holds, and this completes the inductive proof for (B.19). □

Claim 52. *For each s with $2 \leq s \leq S$, if properties (a)–(c) hold for $s-1$, then property (c) holds for s as well.*

Proof. Consider arbitrary s with $2 \leq s \leq S$ such that properties (a)–(c) hold for $s-1$, and assume that each module function call made before line 159 finishes for stratum s is correct in the context of $E|_n$ and Π. We show that property (B.13) holds for s.

For the $\subseteq$ direction, consider arbitrary $F \in (U^s|_o \setminus D^s) \cup A^s$. There are two possibilities, which we discuss below.

- $F \in U^s|_o \setminus D^s$ holds. Then, $F \notin D^s$ implies $F \notin D^s \setminus A^{s-1}$, which together with $F \in U^s|_o$ and Claim 50 implies $F \in U^s|_n$.
- $F \in A^s$ holds. There are two cases. If $F \notin D^{s-1}$ holds, then by Claim 51 we have $F \in U^s|_n$. Otherwise, $F \in A^s \cap D^{s-1} \subseteq A^s \cap D^s$ holds. But then, $F \in D^{s-1}$ implies $F \in \mathsf{O}^{\leq s-1}$, which together with $F \in A^s$ implies $F \in A^{s-1}$. Hence, $F \notin D^s \setminus A^{s-1}$ holds. Moreover, $D^{s-1} \subseteq I$ and $F \in \mathsf{O}^{\leq s-1}$ jointly imply $F \in U^{s-1}|_o \subseteq U^s|_o$. Consequently, we have $F \in U^s|_o \setminus (D^s \setminus A^{s-1})$, and so by Claim 50 we have $F \in U^s|_n$, as required.

The choice of F is arbitrary, so the $\subseteq$ direction holds.

For the $\supseteq$ direction, consider arbitrary $F \in U^s|_n$. There are two possibilities.

- $F \notin I^s|_d$ holds. Then, Claim 51 implies $F \in A^s \subseteq (U^s|_o \setminus D^s) \cup A^s$.
- $F \in I^s|_d = I \setminus (D^s \setminus A^{s-1})$ holds. Then, $F \in U^s|_n$ implies $F \in \mathsf{O}^s$, so we have $F \in U^s|_o \setminus (D^s \setminus A^{s-1})$. Now if $F \in A^{s-1}$ holds, then $F \in (U^s|_o \setminus D^s) \cup A^s$ trivially holds. Otherwise, $F \notin D^s \setminus A^{s-1}$ and $F \notin A^{s-1}$ jointly imply $F \notin D^s$. Consequently, we have $F \in U^s|_o \setminus D^s \subseteq (U^s|_o \setminus D^s) \cup A^s$.

The choice of F is arbitrary, so the $\supseteq$ direction holds. □

B.3 Proof of Theorem 18

Theorem 18. *Consider an arbitrary program* Π*, stratification* λ *of* Π*, dataset* E_0*, and datasets* E_i^- *and* E_i^+ *with* $1 \leq i \leq m$*. Let* $E_i = E_{i-1} \setminus E_i^- \cup E_i^+$ *for* $1 \leq i \leq m$*. Moreover, let* $M^{s,k}$ *be the modules corresponding to* Π *and* λ *as described in Section 5.2.2, and assume that the module functions for each* $M^{s,k}$ *are correct. Let* I_0 *be the result of applying Algorithm 6 to* Π*,* λ*, and* E_0*, and let* I_i *be the result of successively applying Algorithm 7 to* Π*,* λ*,* E_i^-*, and* E_i^+ *for* $1 \leq i \leq m$*. Then,* $I_i = \mathsf{mat}(\Pi, E_i)$ *holds for each* $0 \leq i \leq m$*.*

We first define some additional notations which will be used later in the proof.

- Consider the execution of Algorithm 6 on Π, λ, and E_0. For each $1 \leq s \leq S$ where S is the maximum stratum index of λ, let $U_0^s = \mathsf{O}^{\leq s} \cap \mathsf{mat}(\Pi, E_0)$; moreover, consider the execution of lines 145–155 for stratum s.
  - Let $\ell_0(s)$ be the number of iterations for the loop of lines 148–155.
  - For each $1 \leq j \leq \ell_0(s)$ and each $1 \leq k \leq n(s)$, let $\Delta_{0,j}^{s,k}$ be the value of Δ^k right after line 149 in the jth iteration of lines 148–155.
  - For each $1 \leq k \leq n(s)$,
    - let $C_{0,1}^{s,k}$ be the $\mathsf{Add}^{M^{s,k}}$ call made in line 147, and let $I|_{0,1}^{s,k}$, $\Delta^{\mathsf{p}}|_{0,1}^{s,k}$, $\Delta^{\mathsf{n}}|_{0,1}^{s,k}$, and $\Delta^{\mathsf{m}}|_{0,1}^{s,k}$ be the arguments of the call;
    - for each $1 < j \leq \ell_0(s)$, let $C_{0,j}^{s,k}$ be the $\mathsf{Add}^{M^{s,k}}$ call made in line 155 in the $(j-1)$th iteration of lines 148–155, and let $I|_{0,j}^{s,k}$, $\Delta^{\mathsf{p}}|_{0,j}^{s,k}$, $\Delta^{\mathsf{n}}|_{0,j}^{s,k}$, and $\Delta^{\mathsf{m}}|_{0,j}^{s,k}$ be the arguments of the call;
    - let $H_0^{s,k} = Q_0^{s,k} = C_{0,1}^{s,k}, \ldots, C_{0,\ell_0(s)}^{s,k}$.
- For each $1 \leq i \leq m$, consider the execution of Algorithm 7 on Π, λ, E_i^- and E_i^+. For each $1 \leq s \leq S$, let $U_i^s = \mathsf{O}^{\leq s} \cap \mathsf{mat}(\Pi, E_i)$; moreover, consider the execution of lines 158–159 for stratum s.
  - Let $\ell_i(s)$ be the number of iterations for the loop of lines 167–173.
  - For each $1 \leq j \leq \ell_i(s)$ and each $1 \leq k \leq n(s)$, let $\Delta_{i,j}^{s,k}|_d$ be the value of Δ^k at the beginning of the jth iteration of lines 167–173.
  - Let $m_i(s)$ be the number of iterations for the loop of lines 183–190.
  - For each $1 \leq j \leq m_i(s)$, let $A_{i,j}^s$ and $\Delta_{i,j}^{s,k}|_a$, $1 \leq k \leq n(s)$ be the values of A and Δ^k, respectively, right after line 184 in the jth iteration of lines 183–190.
  - For each $1 \leq k \leq n(s)$,
    - let $C_{i,1}^{s,k}$ be the $\mathsf{Del}^{M^{s,k}}$ call made in line 163, and let $I^{\mathsf{p}}|_{i,1}^{s,k}$, $I^{\mathsf{n}}|_{i,1}^{s,k}$, $\Delta^{\mathsf{p}}|_{i,1}^{s,k}$, $\Delta^{\mathsf{n}}|_{i,1}^{s,k}$, and $\Delta^{\mathsf{m}}|_{i,1}^{s,k}$ be its arguments;

- for each $1 < j \leq \ell_i(s) + 1$, let $C_{i,j}^{s,k}$ be the $\mathsf{Del}^{M^{s,k}}$ call made in line 169 in the $(j-1)$th iteration of lines 167–173, and let $I^{\mathsf{p}}|_{i,j}^{s,k}$, $I^{\mathsf{n}}|_{i,j}^{s,k}$, $\Delta^{\mathsf{p}}|_{i,j}^{s,k}$, $\Delta^{\mathsf{n}}|_{i,j}^{s,k}$, and $\Delta^{\mathsf{m}}|_{i,j}^{s,k}$ be the arguments of the call;
- for $j = \ell_i(s) + 2$, let $C_{i,j}^{s,k}$ be the $\mathsf{Red}^{M^{s,k}}$ call made in line 176, and let $I^{\mathsf{p}}|_{i,j}^{s,k}$, $I^{\mathsf{n}}|_{i,j}^{s,k}$, and $\Delta|_{i,j}^{s,k}$ be the arguments;
- for $j = \ell_i(s) + 3$, let $C_{i,j}^{s,k}$ be the $\mathsf{Add}^{M^{s,k}}$ call made in line 182, and let $I|_{i,j}^{s,k}$, $\Delta^{\mathsf{p}}|_{i,j}^{s,k}$, $\Delta^{\mathsf{n}}|_{i,j}^{s,k}$, and $\Delta^{\mathsf{m}}|_{i,j}^{s,k}$ be the arguments;
- for each $\ell_i(s) + 3 < j < \ell_i(s) + m_i(s) + 3$, let $C_{i,j}^{s,k}$ be the $\mathsf{Add}^{M^{s,k}}$ call made in line 190 in the $(j - \ell_i(s) - 3)$th iteration of lines 183–190, and let $I|_{i,j}^{s,k}$, $\Delta^{\mathsf{p}}|_{i,j}^{s,k}$, $\Delta^{\mathsf{n}}|_{i,j}^{s,k}$, and $\Delta^{\mathsf{m}}|_{i,j}^{s,k}$ be the arguments;
- let $Q_i^{s,k}|_d = C_{i,1}^{s,k}, \ldots, C_{i,\ell_i(s)+1}^{s,k}$, and let $H_i^{s,k}|_d = Q_0^{s,k}, \ldots, Q_{i-1}^{s,k}, Q_i^{s,k}|_d$;
- let $Q_i^{s,k} = C_{i,1}^{s,k}, \ldots, C_{i,\ell_i(s)+m_i(s)+2}^{s,k}$, and let $H_i^{s,k} = Q_0^{s,k}, \ldots, Q_i^{s,k}$.

We next prove by induction on i with $0 \leq i \leq m$ that $I_i = \mathsf{mat}(\Pi, E_i)$ holds. The proof is lengthy, so we break it into several claims.

Claim 53. *Π, λ, $M^{s,k}$, $H_0^{s,k}$, and $\vec{E}_0$ are compatible for each s, $1 \leq s \leq S$ and each k, $1 \leq k \leq n(s)$.*

Proof. Consider arbitrary s with $1 \leq s \leq S$ and arbitrary k with $1 \leq k \leq n(s)$. Note that by definition $H_0^{s,k}$ is a sequence of Add calls. In addition, theorem 38 ensures that each call $C_{0,j}^{s,k}$ in $H_0^{s,k}$ with $1 \leq j \leq \ell_0(s)$ satisfy the conditions specified in the second column of Table 5.1; moreover, for each j with $2 \leq j \leq \ell_0(s)$, calls $C_{0,j-1}^{s,k}$ and $C_{0,j}^{s,k}$ are both of type Add, and their arguments and results satisfy the respective conditions in Table 5.4. Consequently, $H_0^{s,k}$ is a call history by Definition 15. Now $M^{s,k} \subseteq \Pi^s$ holds, and $H_0^{s,k}$ clearly satisfies condition (5.8) in Definition 16. Consequently, Π, λ, $M^{s,k}$, $H_0^{s,k}$, and $\vec{E}_0$ are compatible, as required □

Claim 54. *$I_0 = \mathsf{mat}(\Pi, E_0)$ holds.*

Proof. Consider arbitrary s, $1 \leq s \leq S$ and arbitrary k, $1 \leq k \leq n(s)$. Then, Π, λ, $M^{s,k}$, $H_0^{s,k}$, and $\vec{E}_0$ are compatible by Claim 53. The module functions for $M^{s,k}$ are correct, and so by Definition 17 each call $C_{0,j}$ with $1 \leq j \leq \ell_0(s)$ is correct in the context of E_0 and Π. Since s and k are chosen arbitrarily, each call made during the execution of Algorithm 6 on Π, λ, and E_0 is correct in the context of E_0 and Π. Consequently, $I_0 = \mathsf{mat}(\Pi, E_0)$ holds by Theorem 38. □

Claim 55. *For each i, $1 \leq i \leq m$, if Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible for each s, $1 \leq s \leq S$ and each k, $1 \leq k \leq n(s)$, and $I_{i-1} = \mathsf{mat}(\Pi, E_{i-1})$ holds, then Π, λ, $M^{s,k}$, $H_i^{s,k}|_d$, and $\vec{E}_i$ are compatible for each s, $1 \leq s \leq S$ and each k, $1 \leq k \leq n(s)$.*

Proof. Consider arbitrary i, $1 \leq i \leq m$ such that Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible for each $1 \leq s \leq S$ and each $1 \leq k \leq n(s)$, and $I_{i-1} = \mathsf{mat}(\Pi, E_{i-1})$ holds. Now consider arbitrary $1 \leq s \leq S$ and arbitrary $1 \leq k \leq n(s)$. Then, by the induction assumption Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible, and so $H_{i-1}^{s,k}$ is a history. Note that Theorem 43 implies that calls in $Q_i^{s,k}|_d$ satisfy the relevant conditions specified in the second column of Table 5.1 and those in Table 5.4, and so $H_i^{s,k}|_d$ is a history as well. To show that Π, λ, $M^{s,k}$, $H_i^{s,k}|_d$, and $\vec{E}_i$ are compatible. We discuss the following two cases.

- $i = 1$ holds. Then, the fact that Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible implies that condition (5.8) in Definition 16 holds. Moreover, the definition of $H_0^{s,k}$ implies that condition (5.10) holds as well. Furthermore, the induction assumption and the fact that the module functions for each $M^{s,k}$ are correct imply that each call made in the execution of Algorithm 6 on Π, λ, and E_0 is correct in the context of Π and E_0, and so Theorem 38 implies that condition (5.9) holds as well. Finally, condition (5.11) trivially holds for i. Consequently, Π, λ, $M^{s,k}$, $H_i^{s,k}|_d$, and $\vec{E}_i$ are compatible by Definition 16.
- $i > 1$ holds. Then, the fact that Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible implies that conditions (5.11) and (5.12) in Definition 16 holds for $i - 1$. Moreover, the definition of $H_{i-1}^{s,k}$ implies that condition (5.14) holds for $i - 1$ as well. Furthermore, the induction assumption and the fact that the module functions for each $M^{s,k}$ are

correct imply that each call made in the execution of Algorithm 7 on Π, λ, E_{i-1}^{-}, and E_{i-1}^{+} is correct in the context of Π and E_{i-1}, and so Theorem 43 implies that condition (5.13) holds for $i-1$ as well. Finally, condition (5.11) trivially holds for i. Consequently, Π, λ, $M^{s,k}$, $H_i^{s,k}|_d$, and $\vec{E}_i$ are compatible by Definition 16. □

Claim 56. *For each i, $1 \leq i \leq m$, if Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible for each s, $1 \leq s \leq S$ and each k, $1 \leq k \leq n(s)$, and $I_{i-1} = \mathsf{mat}(\Pi, E_{i-1})$ holds, then Π, λ, $M^{s,k}$, $H_i^{s,k}$, and $\vec{E}_i$ are compatible for each s, $1 \leq s \leq S$ and each k, $1 \leq k \leq n(s)$.*

Proof. Consider arbitrary i, $1 \leq i \leq m$ such that Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible for each $1 \leq s \leq S$ and each $1 \leq k \leq n(s)$, and $I_{i-1} = \mathsf{mat}(\Pi, E_{i-1})$ holds. Now consider arbitrary $1 \leq s \leq S$ and arbitrary $1 \leq k \leq n(s)$. Then, Π, λ, $M^{s,k}$, $H_i^{s,k}|_d$, and $\vec{E}_i$ are compatible by Claim 55. Note that Theorem 43 implies that calls in $Q_i^{s,k}$ satisfy the relevant conditions specified in the second column of Table 5.1 and those in Table 5.4, and so $H_i^{s,k}$ is a history. The fact that Π, λ, $M^{s,k}$, $H_i^{s,k}|_d$, and $\vec{E}_i$ are compatible jointly implies that the relevant conditions in Definition 16 hold. Moreover, Theorem 43 implies that condition (5.12) holds for i. Consequently, Π, λ, $M^{s,k}$, $H_i^{s,k}$, and $\vec{E}_i$ are compatible by Definition 16, as required. □

Claim 57. *For each i, $1 \leq i \leq m$, if Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible for each s, $1 \leq s \leq S$ and each k, $1 \leq k \leq n(s)$, and $I_{i-1} = \mathsf{mat}(\Pi, E_{i-1})$ holds, then $I_i = \mathsf{mat}(\Pi, E_i)$ holds as well.*

Proof. Consider arbitrary $1 \leq i \leq m$ such that Π, λ, $M^{s,k}$, $H_{i-1}^{s,k}$, and $\vec{E}_{i-1}$ are compatible for each $1 \leq s \leq S$ and each $1 \leq k \leq n(s)$, and $I_{i-1} = \mathsf{mat}(\Pi, E_{i-1})$ holds. Now consider arbitrary $1 \leq s \leq S$ and arbitrary $1 \leq k \leq n(s)$. Then, Π, λ, $M^{s,k}$, $H_i^{s,k}$, and $\vec{E}_i$ are compatible by Claim 56. The module functions for $M^{s,k}$ are correct, and so by Definition 17 each call $C_{i,j}$ with $1 \leq j \leq \ell_i(s) + m_i(s) + 2$ is correct in the context of E_i and Π. Since s and k are chosen arbitrarily, each call made during the execution of Algorithm 7 on Π, λ, E_i^{-} and E_i^{+} is correct in the context of E_i and Π. Consequently, $I_i = \mathsf{mat}(\Pi, E_i)$ holds by Theorem 43. □

B.4 Proof of Theorem 20

Theorem 20. *Functions* $\mathsf{Add}^{\mathsf{tc}(R)}$, $\mathsf{Del}^{\mathsf{tc}(R)}$, *and* $\mathsf{Red}^{\mathsf{tc}(R)}$ *are correct for the module that axiomatises relation* R *as transitive, provided that the oracle function* T *is correct.*

Let R be an arbitrary binary relation. $\mathsf{tc}(R)$ is the module that axiomatises R as transitive. For each dataset I, let $R[I]$ denote the set of all R facts in I. Now consider arbitrary program Π, stratification λ of Π, call history H for $\mathsf{tc}(R)$ of the form (5.6), and vector $\vec{E} = E_0, \ldots, E_m$ of datasets such that Π, λ, $\mathsf{tc}(R)$, H, and $\vec{E}$ are compatible; let s be the stratum index such that $\mathsf{tc}(R) \subseteq \Pi^s$ holds. Moreover, for each $0 \leq i \leq m$ and each $1 \leq j \leq n_i$, let $X_{i,j}$ and $Y_{i,j}$ be the values of X_R and Y_R, respectively, after call $C_{i,j}$. We show by induction on i with $0 \leq i \leq m$ that the following properties hold. Then, Definition 17 implies the correctness of the theorem.

(a) For each $1 \leq j \leq n_i$, if call $C_{i,j}$ is of type Del, then the following properties hold.

$$R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup J_{i,j})] \subseteq \mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j} \tag{B.20}$$

$$X_{i,j} \subseteq I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}) \tag{B.21}$$

$$\begin{aligned} \mathsf{tc}(R)_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \setminus \mathsf{mat}(\Pi, E_i) &\subseteq J_{i,j} \\ \subseteq \mathsf{tc}(R)^{\infty}_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] & \end{aligned} \tag{B.22}$$

(b) For each $1 \leq j \leq n_i$, if call $C_{i,j}$ is of type Red, then the following properties hold.

$$\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) = R[I^{\mathsf{p}}_{i,j} \cup J_{i,j}] \tag{B.23}$$

$$\mathsf{tc}(R)\left[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j}\right] \cap \Delta_{i,j} \subseteq J_{i,j} \subseteq \mathsf{mat}(\Pi, E_i) \cap \Delta_{i,j} \tag{B.24}$$

(c) For each $1 \leq j \leq n_i$, if call $C_{i,j}$ is of type Add, then the following properties hold.

$$\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) = R[I_{i,j} \cup J_{i,j}] \tag{B.25}$$

$$X_{i,j} \subseteq I_{i,j} \tag{B.26}$$

$$\mathsf{tc}(R)^{\uparrow}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \subseteq J_{i,j} \subseteq \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \tag{B.27}$$

(d) If $i \neq m$, then $X_{i,n_i} \subseteq U_i$ holds, where U_i is defined as in Definition 16.

The proof is lengthy, so we break it into several claims.

Claim 58. *Properties (a)–(d) hold for $i = 0$.*

Proof. H is a call history, so by Definition 15 each call $C_{0,j}$ with $1 \leq j \leq n_0$ is of type Add. Therefore, properties (a) and (b) trivially hold. We next show by induction on j with $1 \leq j \leq n_0$ that $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j}) = R[I_{0,j} \cup J_{0,j}]$ holds.

- For the base case where $j = 1$, we show that $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1}) = R[I_{0,1} \cup J_{0,1}]$ holds.
  - For the $\subseteq$ direction, first note that for each $R(u, v) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1})$, u and v can be connected using a chain of R facts in $X_{0,1}$. We show by induction on the length of such chains that $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1}) \subseteq R[I_{0,1} \cup J_{0,1}]$ holds, and that each fact in $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1})$ is added to Q at some point in the call of $C_{0,1}$.
    - For the base case, consider arbitrary $R(u, v) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1})$ where u and v can be connected using just one R fact in $X_{0,1}$—that is, $R(u, v) \in X_{0,1}$. But then, line 191 ensure that $R(u, v) \in R[\Delta^{\mathsf{p}}_{0,1}]$ holds, and so $R(u, v)$ is added to Q in line 191. Moreover, Definition 16 ensures $\Delta^{\mathsf{p}}_{0,1} = I_{0,1}$, and so $R(u, v) \in R[I_{0,1}] \subseteq R[I_{0,1} \cup J_{0,1}]$ holds, as required.
    - For the inductive step, consider arbitrary l that satisfies the following condition: for each $R(u, v) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1})$ where u and v can be connected using $(l-1)$ or less facts in $X_{0,1}$, $R(u, v) \in R[I_{0,1} \cup J_{0,1}]$ holds, and $R(u, v)$ is added to Q at some point in the call of $C_{0,1}$. Let $R(a_0, a_l) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1})$ be an arbitrary fact for which a chain, $R(a_0, a_1), \ldots, R(a_{l-1}, a_l) \in X_{0,1}$, exist. Then, by the induction assumption $R(a_1, a_l) \in R[I_{0,1} \cup J_{0,1}]$ holds, and $R(a_1, a_l)$ is added to Q at some point in the execution of Algorithm 8. Consequently, when $R(a_1, a_l)$ is considered in the loop of lines 194–197, line 196 and 197 ensure that $R(a_0, a_l) \in R[I_{0,1} \cup J_{0,1}]$ holds, and that $R(a_0, a_l)$ is added to Q.
  - For the $\supseteq$ direction, $R[I_{0,1}] = R[\Delta^{\mathsf{p}}_{0,1}] \subseteq X_{0,1} \subseteq \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1})$ follows from line 191 and $I_{0,1} = \Delta^{\mathsf{p}}_{0,1}$; moreover, it can be shown by a straightforward induction on the construction of $J_{0,1}$ that $R[J_{0,1}] \subseteq \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1})$ holds. Therefore, $R[I_{0,1} \cup J_{0,1}] \subseteq \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,1})$ holds, as required.

- For the inductive step, consider arbitrary $j > 0$ such that $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1})$ and $R[I_{0,j-1} \cup J_{0,j-1}]$ are equal; we would like to show that $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j}) = R[I_{0,j} \cup J_{0,j}]$ holds as well.
  - For the $\subseteq$ direction, first note that for each $R(u, v) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$, u and v can be connected using a chain of R facts in $X_{0,j}$. We show by induction on the length of such chains that $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j}) \subseteq R[I_{0,j} \cup J_{0,j}]$ holds, and that each fact in $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$ is either in $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1})$ or added to Q at some point in the call of $C_{0,j}$.
    - For the base case, consider arbitrary $R(u, v) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$ where a chain of length one exists in $X_{0,j}$. This implies $R(u, v) \in X_{0,j}$. Then, there are two possibilities. If $R(u, v) \in X_{0,j-1}$, then the induction assumption on $j-1$ and Definition 15 imply $R(u, v) \in R[I_{0,j-1} \cup J_{0,j-1}] \subseteq R[I_{0,j}]$. Otherwise, line 191 ensures that $R(u, v) \in R[\Delta^{\mathsf{p}}_{0,j}] \subseteq R[I_{0,j}]$ holds, and that $R(u, v)$ is added to Q in line 191.
    - For the inductive step, consider arbitrary l that satisfies the following condition: for each $R(u, v) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$ where u and v can be connected using $(l-1)$ or less facts in $X_{0,j}$, $R(u, v) \in R[I_{0,j} \cup J_{0,j}]$ holds, and if $R(u, v) \notin \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1})$, then it is added to Q at some point in the call of $C_{0,j}$. Let $R(a_0, a_l) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$ be an arbitrary fact for which a chain of l facts, $R(a_0, a_1), \ldots, R(a_{l-1}, a_l) \in X_{0,j}$, exists. There are three possibilities, which we discuss below.
      - If $R(a_0, a_1) \in X_{0,j-1}$ and $R(a_1, a_l) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1})$, then by the induction assumption on $j-1$ we have $R(a_0, a_l) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1})$, and so $R(a_0, a_l) \in R[I_{0,j-1} \cup J_{0,j-1}] \subseteq R[I_{0,j}]$ holds.
      - If $R(a_0, a_1) \in X_{0,j} \setminus X_{0,j-1}$ and $R(a_1, a_l) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1})$, then $R(a_1, a_l) \in R[I_{0,j-1} \cup J_{0,j-1}]$ holds. Definition 15 implies $R(a_1, a_l) \in I_{0,j}$, and so there are two cases. If $R(a_1, a_l) \in I_{0,j} \setminus \Delta^{\mathsf{p}}_{0,j}$, then, lines 192 and 193 ensure that $R(a_0, a_l) \in R[I_{0,j} \cup J_{0,j}]$ holds; otherwise, $R(a_1, a_l) \in \Delta^{\mathsf{p}}_{0,j}$ holds, and so we have $R(a_0, a_l) \in R[I_{0,j} \cup J_{0,j}]$ due to lines 194–197.

· If $R(a_0, a_1) \in X_{0,j}$ and $R(a_1, a_l) \in \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j}) \setminus \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1})$ hold, then by the induction assumption for $l-1$ we know that $R(a_1, a_l)$ is added to Q at some point. Consequently, when $R(a_1, a_l)$ is considered in the loop of lines 194–197, line 196 and 197 ensure that $R(a_0, a_l) \in R[I_{0,j} \cup J_{0,j}]$ holds.

Finally, if $R(a_0, a_l) \notin \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1})$, then $R(a_0, a_l) \in \Delta^{\mathsf{p}}_{0,j} \cup J_{0,j}$ follows from $\mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$ and $R(a_0, a_l) \in R[I_{0,j} \cup J_{0,j}]$. Line 191, and the fact that Q and J are always updated together imply that $R(a_0, a_l)$ is added to Q at some point as well.

– For the $\supseteq$ direction, $R[I_{0,j-1} \cup J_{0,j-1}] = \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j-1}) \subseteq \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$ holds by the induction assumption on $j-1$; moreover, line 191 ensures that $R[\Delta^{\mathsf{p}}_{0,j}] \subseteq X_{0,j} \subseteq \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$ holds; finally, it can be shown by a straightforward induction on the construction of $J_{0,j}$ that $R[J_{0,j}] \subseteq \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$ holds. Consequently, $R[I_{0,j} \cup J_{0,j}] \subseteq \mathsf{mat}(\Pi^{\mathsf{tc}(R)}, X_{0,j})$ holds, as required.

Now we show that property (B.26) holds for $i = 0$ and each $1 \leq j \leq n_0$. This is achieved by an induction on j.

- For the induction base where $j = 1$, consider Add call $C_{0,1}$. Definition 11 ensures $\Delta^{\mathsf{p}}_{0,1} \subseteq I_{0,1}$, and so line 191 ensures $X_{0,1} \subseteq I_{0,1}$.
- For the inductive step, consider arbitrary $j > 1$ and the corresponding Add call $C_{0,j}$ such that $X_{0,j-1} \subseteq I_{0,j-1}$ holds. Then, consider arbitrary fact $F \in X_{0,j}$. There are two cases. If $F \in X_{0,j-1}$, then the induction assumption for $j-1$ and Definition 15 jointly imply $F \in I_{0,j}$. Otherwise, F is added to $X_{0,j}$ in line 191, and so we have $F \in \Delta^{\mathsf{p}}_{0,j} \subseteq I_{0,j}$. Since F is chosen arbitrarily, we have $X_{0,j} \subseteq I_{0,j}$, as required.

We next prove that the left-hand side $\subseteq$ of property (B.27) holds for $i = 0$ and each $1 \leq j \leq n_0$. To this end, consider arbitrary j with $1 \leq j \leq n_0$ and arbitrary fact $F \in \mathsf{tc}(R)^{\uparrow}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$. By the definition of $\mathsf{tc}(R)^{\uparrow}$ we have $F \in \mathsf{tc}(R)\left[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}\right] \setminus I_{0,j}$. Note that $\mathsf{tc}(R)$ involves only R facts; moreover, $\Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j} \subseteq I_{0,j}$ holds by Definition 11. Consequently, $F \in \mathsf{mat}(\mathsf{tc}(R), R[I_{0,j}]) \setminus I_{0,j}$ holds. Now $\mathsf{mat}(\mathsf{tc}(R), R[I_{0,j}]) \subseteq \mathsf{mat}(\mathsf{tc}(R), R[I_{0,j} \cup J_{0,j}]) = R[I_{0,j} \cup J_{0,j}]$ follows from

$\mathsf{mat}(\mathsf{tc}(R), X_{0,j}) = R[I_{0,j} \cup J_{0,j}]$. Therefore, $F \in R[I_{0,j} \cup J_{0,j}] \setminus I_{0,j} \subseteq J_{0,j}$ holds. F and j are chosen arbitrarily, so the left-hand side $\subseteq$ of property (B.27) holds for $i = 0$ and each $1 \leq j \leq n_0$, as required.

Next, we show that the right-hand side $\subseteq$ of property (B.27) holds for $i = 0$ and each $1 \leq j \leq n_0$. To this end, consider arbitrary j with $1 \leq j \leq n_0$ and the corresponding Add call $C_{0,j}$. Let K be the total number of iterations for the loop of lines 194–197; moreover, let J^0 and Q^0 be the value of J and Q, respectively, right before line 194, and for each $1 \leq k \leq K$, let J^k and Q^k be the value of J and Q, respectively, right after the kth iteration of lines 194–197; finally, let Δ_i be defined with respect to $\mathsf{tc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$ as in Definition 9. We prove $J^k \subseteq \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$ by induction on k.

- For the induction base where $k = 0$, consider arbitrary $R(u, w) \in J^0$. Lines 192 and 193 ensure that there exist $R(u, v) \in \Delta^{\mathsf{p}}_{0,j}$ and $R(v, w) \in I_{0,j}$ such that $R(u, w) \notin I_{0,j}$ holds. Consequently, $R(u, w) \in \mathsf{tc}(R)^{\uparrow}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}] \subseteq \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$ holds, as required.
- For the inductive step, consider arbitrary $k > 0$ such that J^{k-1} is contained in $\mathsf{tc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$. Let $R(u, w)$ be an arbitrary fact in J^k. If $R(u, w) \in J^{k-1}$ holds, then by the induction assumption $R(u, w) \in \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$ holds. Otherwise, $R(u, w)$ is added to J^k in the kth iteration of lines 194–197. Consequently, there exist $R(u, v) \in X_{0,j} \subseteq I_{0,j}$ and $R(v, w) \in Q^{k-1}$ such that $R(u, w) \notin I_{0,j} \cup J^{k-1}$ holds. But then, lines 191, 193, and 197 imply $R(v, w) \in R[\Delta^{\mathsf{p}}_{0,j}] \cup J^{k-1}$. We discuss the following possibilities.
  - If $R(v, w) \in R[\Delta^{\mathsf{p}}_{0,j}]$ holds, then $R(u, v) \in I_{0,j}$ and $R(u, w) \notin I_{0,j}$ jointly imply $R(u, w) \in \mathsf{tc}(R)^{\uparrow}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}] \subseteq \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$.
  - If $R(v, w) \in J^{k-1}$, then by the induction assumption for $k - 1$ and the definition of $\mathsf{tc}(R)^{\uparrow}_{\infty}$ there exists index k' such that $R(v, w) \in \Delta_{k'}$ holds. Consequently, $R(u, v) \in I_{0,j}$, $R(u, w) \notin I_{0,j}$, and the definition of $\mathsf{tc}(R)^{\uparrow}_{\infty}$ jointly imply $R(u, w) \in \bigcup_{1 \leq i \leq k'+1} \Delta_i$, which ensures $R(u, w) \in \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$, as required. □

Finally, property ((d)) is a direct consequence of condition (5.9) in Definition 16 and the way X_R is updated in line 191.

Claim 59. *For each $1 \leq i \leq m$, if properties (a)–(d) holds for $i-1$, then property (a) holds for i as well.*

Proof. Consider arbitrary $1 \leq i \leq m$ such that properties (a)–(d) hold for $i-1$, we show that property (a) holds for i as well. To this end, we prove by induction on $1 \leq j \leq n_i$ that if call $C_{i,j}$ is of type Del, then the following property holds.

$$R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup J_{i,j})] \subseteq \mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j} \tag{B.28}$$

For $j = 1$, we show that $R[I^{\mathsf{p}}_{i,1}] \setminus \mathsf{mat}(\mathsf{tc}(R), X_{i,1}) \subseteq R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$ holds. Definition 16 ensures that $C_{i-1,n_{i-1}}$ is of type Add with $J_{i-1,n_{i-1}} = \emptyset$, and so the induction assumption for $i-1$ implies $\mathsf{mat}(\mathsf{tc}(R), X_{i-1,n_{i-1}}) = R[I_{i-1,n_{i-1}}]$. Moreover, there exists exactly one stratum s such that $\mathsf{tc}(R) \subseteq \Pi^s$ holds, which implies $\lambda(R) = s$, and so by condition (5.11) we have $R[I^{\mathsf{p}}_{i,1}] = R[I_{i-1,n_{i-1}}]$. Therefore, $R[I^{\mathsf{p}}_{i,1}] = \mathsf{mat}(\mathsf{tc}(R), X_{i-1,n_{i-1}})$ holds, and so for each $R(u,v) \in R[I^{\mathsf{p}}_{i,1}]$, u and v can be connected using a (finite) chain of R facts in $X_{i-1,n_{i-1}}$. We show by induction on the length of such chains that $R[I^{\mathsf{p}}_{i,1}] \setminus \mathsf{mat}(\mathsf{tc}(R), X_{i,1}) \subseteq R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$ holds, and that each fact in $R[I^{\mathsf{p}}_{i,1}]$ but not in $\mathsf{mat}(\mathsf{tc}(R), X_{i,1})$ is added to Q at some point during the execution of call $C_{i,1}$.

- For the base case, consider arbitrary $R(u,v) \in R[I^{\mathsf{p}}_{i,1}] \setminus \mathsf{mat}(\mathsf{tc}(R), X_{i,1})$ where u and v can be connected using just one R fact in $X_{i-1,n_{i-1}}$—that is, $R(u,v) \in X_{i-1,n_{i-1}}$. But then, $R(u,v) \notin \mathsf{mat}(\mathsf{tc}(R), X_{i,1})$ implies $R(u,v) \in X_{i-1,n_{i-1}} \setminus X_{i,1}$, and so line 210 ensures $R(u,v) \in R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1}] \subseteq R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$. Together with lines 199, 203–204, and 207–208 this ensures that $R(u,v)$ is added to Q during the execution of $C_{i,1}$.
- For the inductive step, consider arbitrary l that satisfies the following condition: for each $R(u,v) \in R[I^{\mathsf{p}}_{i,1}] \setminus \mathsf{mat}(\mathsf{tc}(R), X_{i,1})$ where u and v can be connected using $(l-1)$ or less facts in $X_{i-1,n_{i-1}}$, $R(u,v) \in R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$ holds, and $R(u,v)$ is added to Q at some point during the execution of $C_{i,1}$. Let $R(a_0, a_l) \in R[I^{\mathsf{p}}_{i,1}] \setminus \mathsf{mat}(\mathsf{tc}(R), X_{i,1})$ be an arbitrary fact for which a chain of l facts, $R(a_0, a_1), \ldots, R(a_{l-1}, a_l) \in X_{i-1,n_{i-1}}$ exists.

Then, $R(a_1, a_l) \in \mathsf{mat}(\mathsf{tc}(R), X_{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$ holds. There are two possibilities, which we discuss below.

- $R(a_1, a_l) \notin \mathsf{mat}(\mathsf{tc}(R), X_{i,1})$. In this case, by the induction assumption for $l-1$ we have $R(a_1, a_l) \in R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$, and that $R(a_1, a_l)$ is added to Q at some point during the execution of $C_{i,1}$. Consequently, $R(a_0, a_l)$ is considered in line 206; since $R(a_0, a_l) \in R[I^{\mathsf{p}}_{i,1}]$, it is added to Q at some point in line 199, line 203, or line 207 during the execution of $C_{i,1}$; moreover, $R(a_0, a_l)$ is guaranteed to be in $R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$.
- $R(a_0, a_1) \notin X_{i,1}$. In this case, we show by induction on k with $1 \leq k \leq l$ that $R(a_0, a_k) \in R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$, and that $R(a_0, a_k)$ is added to Q at some point during the execution of call $C_{i,1}$.
  - For the base case where $k = 1$, we have $R(a_0, a_1) \in X_{i-1,n_{i-1}} \setminus X_{i,1}$. Line 210 ensures that $R(a_0, a_1) \in R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1}]$ holds. Together with lines 199, 203–204, and 207–208 this ensures that $R(a_0, a_1)$ is added to Q at some point during the execution of $C_{i,1}$.
  - For the inductive step, consider arbitrary $k > 1$ such that $R(a_0, a_{k-1})$ belongs to $R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$, and that $R(a_0, a_{k-1})$ is added to Q at some point during the execution of $C_{i,1}$. Then, consider the loop of lines 201–209 where $R(a_0, a_{k-1})$ is removed from Q. Line 202 and $R(a_{k-1}, a_k) \in X_{i-1,n_{i-1}}$ jointly imply that $R(a_0, a_k)$ will be examined in line 202. We clearly have $R(a_0, a_k) \in \mathsf{mat}(\mathsf{tc}(R), X_{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$, and so $R(a_0, a_k)$ is added to Q at some point in line 199, line 203, or line 207 during the execution of $C_{i,1}$; moreover, $R(a_0, a_k)$ is guaranteed to be in $R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup \cup Y_{i,1}]$.

This completes our proof for $R[I^{\mathsf{p}}_{i,1}] \setminus \mathsf{mat}(\mathsf{tc}(R), X_{i,1}) \subseteq R[\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1} \cup Y_{i,1}]$, which implies the correctness of property (B.28) for $j = 1$.

For the inductive step, consider arbitrary $1 < j \leq n_i$ such that property (B.28) holds for $j-1$ if call $C_{i,j-1}$ is of type Del, and we show that the property holds for j as well if call $C_{i,j}$ is of type Del. Assume that call $C_{i,j}$ is indeed of type Del. Then, Definition 15 ensures that $C_{i,j-1}$ is of type Del as well. Consequently, property (B.28) holds for $j-1$,

and we would like to show that the property holds for j as well. To this end, we first show that $R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \setminus (\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j-1}) \subseteq R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$ holds. Definition 15 implies $R[I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1} \cup J_{i,j-1})] = R[I^{\mathsf{p}}_{i,j} \setminus J_{i,j-1}] = R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}]$. But then, the induction assumption for $j-1$ implies $R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \subseteq \mathsf{mat}(\mathsf{tc}(R), X_{i,j-1}) \cup Y_{i,j-1}$, and so for each $R(u,v) \in R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \setminus (\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j-1})$, u and v can be connected using a (finite) chain of R facts in $X_{i,j-1}$. We show by induction on the length of such chains that $R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \setminus (\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j-1}) \subseteq R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}]$ holds, and that each fact in $R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \setminus (\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j-1})$ is added to Q at some point during the execution of $C_{i,j}$.

- For the base case, consider arbitrary $R(u,v) \in R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \setminus \mathsf{mat}(\mathsf{tc}(R), X_{i,j})$ where u and v can be connected using just one R fact in $X_{i,j-1}$—that is, $R(u,v) \in X_{i,j-1}$. But then, $R(u,v) \notin \mathsf{mat}(\mathsf{tc}(R), X_{i,j})$ implies $R(u,v) \in X_{i,j-1} \setminus X_{i,j}$, and so line 210 ensures $R(u,v) \in R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}]$. Together with lines 199, 203–204, and 207–208 this ensures that $R(u,v)$ is added to Q at some point during the execution of $C_{i,j}$.
- For the inductive step, consider arbitrary l that satisfies the following condition: for each $R(u,v) \in R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \setminus (\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j-1})$ where u and v can be connected using $(l-1)$ or less facts in $X_{i,j-1}$, $R(u,v) \in R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$ holds, and $R(u,v)$ is added to Q at some point during the execution of $C_{i,j}$. Let $R(a_0, a_l)$ be an arbitrary fact in $R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \setminus (\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j-1})$ for which a chain of l facts, $R(a_0, a_1), \ldots, R(a_{l-1}, a_l) \in X_{i,j-1}$ exists. Then, $R(a_1, a_l) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,j-1})$ holds, which implies $R(a_1, a_l) \in \mathsf{mat}(\mathsf{tc}(R), X_{i-1,n_{i-1}})$. Together with Definition 16 and the induction assumption for $i-1$ this implies $R(a_1, a_l) \in R[I^{\mathsf{p}}_{i,1} \setminus \Delta^{\mathsf{m}}_{i,1}]$. We would like to show that $R(a_0, a_l) \in R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$ holds, and that $R(a_0, a_l)$ is added to Q at some point during the execution of $C_{i,j}$. Since $R(a_0, a_l) \notin \mathsf{mat}(\mathsf{tc}(R), X_{i,j})$ holds, we have the following possibilities.
  - $R(a_1, a_l) \notin \mathsf{mat}(\mathsf{tc}(R), X_{i,j})$. There are two cases.
    - If $R(a_1, a_l) \in R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{m}}_{i,j} \cup Y_{i,j-1})]$, then the induction assumption for $(l-1)$ implies $R(a_1, a_l) \in R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$; moreover, $R(a_1, a_l)$ is added to Q at some point during the execution of $C_{i,j}$. Consequently, $R(a_0, a_l)$ is considered

in line 206; since $R(a_0, a_l) \in R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \subseteq R[I^{\mathsf{p}}_{i,j}]$, it is added to Q at some point in line 199, line 203, or line 207 during the execution of $C_{i,j}$; moreover, $R(a_0, a_l)$ is guaranteed to be in $R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$.

- If $R(a_1, a_l) \notin R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{m}}_{i,j} \cup Y_{i,j-1})]$, then we have $R(a_1, a_l) \in R[I^{\mathsf{p}}_{i,1} \setminus \Delta^{\mathsf{m}}_{i,1}]$ and $R(a_1, a_l) \notin R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{m}}_{i,j} \cup Y_{i,j-1})]$, and so Definitions 15 and 16 ensure that there exists k with $1 \leq k < j$ such that $R(a_1, a_l) \in R[\Delta^{\mathsf{p}}_{i,k} \cup J_{i,k} \cup Y_{i,k}]$ holds, and that $R(a_1, a_l)$ is added to Q at some point during the execution of $C_{i,k}$. Consequently, $R(a_0, a_l)$ is considered in line 206; since $R(a_0, a_l) \in R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \subseteq R[I^{\mathsf{p}}_{i,j}] \subseteq R[I^{\mathsf{p}}_{i,k}]$, it is added to Q at some point in line 199, line 203, or line 207 during the execution of $C_{i,k}$; moreover, $R(a_0, a_l)$ is guaranteed to be in $R[\Delta^{\mathsf{p}}_{i,k} \cup J_{i,k} \cup Y_{i,k}]$, which is disjoint from $R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{m}}_{i,j} \cup Y_{i,j-1})]$. This is clearly a contradiction.

– $R(a_0, a_1) \notin X_{i,j}$. First, it can be shown straightforwardly that for each $1 \leq k < l$, we have $R(a_0, a_k) \in R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{m}}_{i,j} \cup Y_{i,j-1})]$ (otherwise, $R(a_0, a_l)$ will be derived before the call $C_{i,j}$ and $R(a_0, a_l) \notin R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{m}}_{i,j} \cup Y_{i,j-1})]$ holds, which is a contradiction). Then, we show by induction on k with $1 \leq k \leq l$ that $R(a_0, a_k) \in R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$ holds, and that $R(a_0, a_k)$ is added to Q at some point during the execution of $C_{i,j}$.

  - For the base case where $k = 1$, we have $R(a_0, a_1) \in X_{i,j-1} \setminus X_{i,j}$. Line 210 ensures that $R(a_0, a_1) \in R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}]$ holds. Together with lines 199, 203–204, and 207–208 this ensures that $R(a_0, a_1)$ is added to Q at some point during the execution of $C_{i,j}$.

  - For the inductive step, consider arbitrary $k > 1$ such that $R(a_0, a_{k-1})$ belongs to $R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$, and that $R(a_0, a_{k-1})$ is added to Q at some point during the execution of $C_{i,j}$. Then, consider the loop of lines 201–209 where $R(a_0, a_{k-1})$ is removed from Q. Line 202 and $R(a_{k-1}, a_k) \in X_{i,j-1}$ jointly imply that $R(a_0, a_k)$ will be examined in line 202. We clearly have $R(a_0, a_k) \in R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{m}}_{i,j} \cup Y_{i,j-1})] \subseteq R[I^{\mathsf{p}}_{i,j}]$, and so $R(a_0, a_k)$ is added to Q

at some point in line 199, line 203, or line 207 during the execution of $C_{i,j}$; moreover, $R(a_0, a_k)$ is guaranteed to be in $R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$.

This completes our proof for $R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] \setminus (\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) \cup Y_{i,j-1}) \subseteq R[\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j} \cup Y_{i,j}]$, which implies property (B.28) for j.

Now we prove by induction on $1 \leq j \leq n_i$ that the following statement is true: if call $C_{i,j}$ is of type Del, then property (B.29) holds.

$$X_{i,j} \subseteq I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}) \tag{B.29}$$

- For the induction base where $j = 1$, by Definition 16 call $C_{i-1,n_{i-1}}$ is of type Add, and so by the induction assumption for $i-1$ we have $X_{i-1,n_{i-1}} \subseteq I_{i-1,n_{i-1}}$; moreover, there exists stratum s such that $\mathsf{tc}(R) \subseteq \Pi^s$ and $\mathsf{O}^{\leq s} \cap I_{i-1,n_{i-1}} = \mathsf{O}^{\leq s} \cap I^{\mathsf{p}}_{i,1}$ hold. Thus, $X_{i-1,n_{i-1}} \subseteq I^{\mathsf{p}}_{i,1}$ holds. $X_{i,1} = X_{i-1,n_{i-1}} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1}) \subseteq I^{\mathsf{p}}_{i,1} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup J_{i,1})$ follows from line 210. Definition 16 ensures $\Delta^{\mathsf{m}}_{i,1} = \emptyset$, and so $X_{i,1} \subseteq I^{\mathsf{p}}_{i,1} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1})$, as required.
- For the inductive step, consider arbitrary $j > 1$ such that the statement is true for $j-1$. If call $C_{i,j}$ is not of type Del, the statement is clearly true for j. Otherwise, by Definition 15 call $C_{i,j-1}$ is of type Del as well, and so the induction assumption ensures $X_{i,j-1} \subseteq I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1})$. By Definition 15 we have $X_{i,j-1} \subseteq I^{\mathsf{p}}_{i,j}$. Now consider arbitrary $F \in X_{i,j}$. Line 210 ensures $X_{i,j} = X_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j})$, and so $F \in I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$ holds. Moreover, $F \in X_{i,j-1}$ implies $F \notin J_{i,j-1}$ due to line 210, which together with Definition 15 implies $F \notin \Delta^{\mathsf{m}}_{i,j}$. Consequently, $F \in I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j})$ holds. Since the choice of F is arbitrary, we have $X_{i,j} \subseteq I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j})$, as required.

We next show by induction on $1 \leq j \leq n_i$ that the following statement is true: if call $C_{i,j}$ is of type Del, then property (B.30) holds.

$$\mathsf{tc}(R)_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \setminus \mathsf{mat}(\Pi, E_i) \subseteq J_{i,j} \tag{B.30}$$

- For the induction base where $j = 1$, consider the corresponding Del call $C_{i,1}$. Let F be an arbitrary fact in $\mathsf{tc}(R)_{\downarrow}[I^{\mathsf{p}}_{i,1}, I^{\mathsf{n}}_{i,1} \colon \Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1}, \Delta^{\mathsf{n}}_{i,1}] \setminus \mathsf{mat}(\Pi, E_i)$. Then $F \in \mathsf{tc}(R)\Big[I^{\mathsf{p}}_{i,1}, I^{\mathsf{n}}_{i,1} \colon \Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1}, \Delta^{\mathsf{n}}_{i,1}\Big] \cap I^{\mathsf{p}}_{i,1}$ and $F \notin \Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1} \cup \mathsf{mat}(\Pi, E_i)$ hold by the

definition of $\mathsf{tc}(R)_{\downarrow}$. Module $\mathsf{tc}(R)$ involves no negation in the body of its rules; moreover, $\Delta^{\mathsf{m}}_{i,1} = \emptyset$ holds by Definition 16. Consequently, we have $F \in \mathsf{tc}(R)\left[I^{\mathsf{p}}_{i,1} \vdots \Delta^{\mathsf{p}}_{i,1}\right] \cap I^{\mathsf{p}}_{i,1}$, and we discuss the following possibilities.

- There exist $R(u,v) \in \Delta^{\mathsf{p}}_{i,1}$ and $R(v,w) \in I^{\mathsf{p}}_{i,1}$ such that $F = R(u,w)$ holds. Definition 16 and property (c) jointly imply $\mathsf{mat}(\mathsf{tc}(R), X_{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$, and so there exist a (finite) chain of R facts in $X_{i-1,n_{i-1}}$ that connect v and w. Let $R(a_0,a_1),\ldots,R(a_{l-1},a_l)$ be such a chain where $a_0 = v$ and $a_l = w$ hold, and we show by induction on $0 \leq k \leq l$ that $R(u,a_l)$ is added to Q at some point during the execution of call $C_{i,1}$.
  - For the base case where $k = 0$, we have $R(u,a_0) = R(u,v) \in \Delta^{\mathsf{p}}_{i,1}$, and so line 199 ensures that $R(u,a_0)$ is added to Q during the execution of $C_{i,1}$.
  - For the inductive step, consider arbitrary $0 < k \leq l$ such that $R(u,a_{k-1})$ is added to Q at some point during the execution of $C_{i,1}$. Then, $R(u,a_{k-1})$ will be removed from Q in line 201 at some point during the execution of $C_{i,1}$, and so $R(a_{k-1},a_k) \in X_{i-1,n_{i-1}}$ implies that $R(u,a_k)$ will be considered in line 202. Now property (c) for $i-1$ and $R(u,v) \in R[\Delta^{\mathsf{p}}_{i,1}] \subseteq R[I^{\mathsf{p}}_{i,1}] = R[I_{i-1,n_{i-1}}]$ jointly imply $R(u,a_k) \in I^{\mathsf{p}}_{i,1}$. Consequently, whether $R(u,a_k) \in I^{\mathsf{p}} \setminus S$ holds or not in line 202, $R(u,a_k)$ is guaranteed to be added to Q in line 199, line 203, or line 207 at some point during the execution of $C_{i,1}$.

  Now since $a_l = w$ holds, $R(u,w)$ is added to Q at some point during the execution of $C_{i,1}$. Then, $R(u,w) \notin \Delta^{\mathsf{p}}_{i,1}$ implies that $R(u,w)$ is added to Q in either line 203 or line 207. Either way, $T(R(u,w))$ will be examined. Assume for the sake of a contradiction that $T(R(u,w)) = \mathsf{t}$ holds. Then, Definition 19 ensures that $R(u,w) \in \mathsf{mat}(\Pi, E_i)$ holds, which is a contradiction. Consequently, we have $T(R(u,w) = \mathsf{f}$, and so $R(u,w)$ is added to J either in line 204 or line 208. Therefore, $R(u,w) \in J_{i,1}$ holds, as required.
- There exist $R(t,u) \in I^{\mathsf{p}}_{i,1}$ and $R(u,v) \in \Delta^{\mathsf{p}}_{i,1}$ such that $F = R(t,v)$ holds. The proof is analogous to the first case, so we omit the details.

- For the inductive step, consider arbitrary $j > 1$ such that the statement is true for $j-1$. Now if call $C_{i,j}$ is not of type Del, then the statement is clearly true for j as well. Assume that call $C_{i,j}$ is indeed of type Del, and we show that property (B.30) holds. To this end, consider arbitrary fact $F \in \mathsf{tc}(R)_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \setminus \mathsf{mat}(\Pi, E_i)$. Similarly to the base case, we have $F \in \mathsf{tc}(R)\left[I^{\mathsf{p}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}\right] \cap I^{\mathsf{p}}_{i,j}$ and $F \notin \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup \mathsf{mat}(\Pi, E_i)$. We discuss the following possibilities.
  - There exist $R(u,v) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$ and $R(v,w) \in I^{\mathsf{p}}_{i,j}$ such that $F = R(u,w)$ holds. There are two cases.
    - $R(u,v) \in \Delta^{\mathsf{m}}_{i,j}$ holds. Since call $C_{i,j}$ is of type Del, by Definition 15 call $C_{i,j-1}$ is of type Del as well; moreover, $R(u,v) \in J_{i,j-1}$ and $R(v,w) \in I^{\mathsf{p}}_{i,j-1}$ hold. $R(u,v) \in J_{i,j-1}$ implies that $R(u,v)$ is added to Q at some point during the execution of $C_{i,j-1}$; $R(u,v)$ could also be added to Q in a Del call before $C_{i,j-1}$, and let $C_{i,m}$ be such a call with the smallest m, where $m \leq j-1$ holds. We discuss the following cases.
      - $m = 1$ holds. $R(v,w) \in I^{\mathsf{p}}_{i,j-1}$ and Definition 15 imply $R(v,w) \in I^{\mathsf{p}}_{i,1}$. But then, as argued in the case for $j = 1$, we have $F = R(u,w) \in \Delta^{\mathsf{p}}_{i,1} \cup J_{i,1}$, which is a contradiction to $F \in I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$.
      - $m > 1$ and $R(v,w)$ is not added to Q in any Del call $C_{i,n}$ with $1 \leq n < m$. Each fact added to J or Y_R is guaranteed to be added to Q, and so we have $R(v,w) \notin J_{i,m-1} \cup Y_{i,m-1}$. Now Definition 15 implies $R(v,w) \in I^{\mathsf{p}}_{i,m} = I^{\mathsf{p}}_{i,m-1} \setminus (\Delta^{\mathsf{p}}_{i,m-1} \cup \Delta^{\mathsf{m}}_{i,m-1})$. By property (B.28) we have $R(v,w) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,m-1}) \cup J_{i,m-1} \cup Y_{i,m-1}$. Together with $R(v,w) \notin J_{i,m-1} \cup Y_{i,m-1}$ this implies $R(v,w) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,m-1})$, and so there exist a (finite) chain of R facts in $X_{i,m-1}$ that connect v and w. Let $R(a_0,a_1), \ldots, R(a_{l-1},a_l)$ be such a chain where $v = a_0$ and $w = a_l$ hold. It can then be shown by induction on $0 \leq k \leq l$ that $R(u,a_k)$ is added to Q at some point during the execution of some $C_{i,m'}$ with $1 \leq m' \leq m$. Assume without loss of generality that $R(u,w)$ is added to Q at some point during the execution of $C_{i,n}$ with $1 \leq n \leq m$. Then,

Definition 19 and $F = R(u, w) \notin \mathsf{mat}(\Pi, E_i)$ jointly imply $F \in \Delta^{\mathsf{p}}_{i,n} \cup J_{i,n}$. But then, we have $F \notin I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$, which is a contradiction.

- $m > 1$ and there exists n with $1 \leq n < m$ such that $R(v, w)$ is added to Q at some point during the execution of $C_{i,n}$. If $n = 1$ holds, then similarly to the case of $m = 1$ we have $F \in \Delta^{\mathsf{p}}_{i,1} \cup J_{i,1}$, which is a contradiction. Otherwise, $R(u, v) \in \Delta^{\mathsf{m}}_{i,j} \subseteq I^{\mathsf{p}}_{i,j} \subseteq I^{\mathsf{p}}_{i,m} \subseteq I^{\mathsf{p}}_{i,n} = I^{\mathsf{p}}_{i,n-1} \setminus (\Delta^{\mathsf{p}}_{i,n-1} \cup \Delta^{\mathsf{m}}_{i,n-1})$ follows from Definition 15. But then, property (B.28) clearly ensures $R(u, v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,n-1}) \cup Y_{i,n-1} \cup J_{i,n-1}$. Note that m is the smallest index such that $R(u, v)$ is added to Q at some point during the execution of $C_{i,m}$, and so $n < m$ implies $R(u, v) \notin Y_{i,n-1} \cup J_{i,n-1}$. Consequently, there exist a finite chain of R facts in $X_{i,n-1}$ that connect u and v. $R(v, w)$ is added to Q during the execution of $C_{i,n}$, and so in a similar way as in the previous case there exists index $1 \leq n' \leq n$ such that $R(u, w) \in \Delta^{\mathsf{p}}_{i,n'} \cup J_{i,n'}$ holds, which is a contradiction to $R(u, w) \in I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$.

∗ $R(u, v) \in \Delta^{\mathsf{p}}_{i,j}$ holds. Line 199 ensures that $R(u, v)$ is added to Q during the execution of $C_{i,j}$. Let $C_{i,m}$ be the Del call with the smallest m such that $R(u, v)$ is added to Q during the execution of $C_{i,m}$, then $m \leq j$ holds. We discuss the following cases.

- $m = 1$ holds. $R(v, w) \in I^{\mathsf{p}}_{i,j-1}$ and Definition 15 imply $R(v, w) \in I^{\mathsf{p}}_{i,1}$. But then, as argued for $j = 1$, we have $F = R(u, w) \in \Delta^{\mathsf{p}}_{i,1} \cup J_{i,1}$, which is a contradiction to $F \in I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$.

- $m > 1$ and $R(v, w)$ is not added to Q in any Del call $C_{i,n}$ with $1 \leq n < m$. Each fact added to J or Y_R is guaranteed to be added to Q, and so we have $R(v, w) \notin J_{i,m-1} \cup Y_{i,m-1}$. Now Definition 15 implies $R(v, w) \in I^{\mathsf{p}}_{i,m} = I^{\mathsf{p}}_{i,m-1} \setminus (\Delta^{\mathsf{p}}_{i,m-1} \cup \Delta^{\mathsf{m}}_{i,m-1})$. By property (B.28) we have $R(v, w) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,m-1}) \cup J_{i,m-1} \cup Y_{i,m-1}$. Together with $R(v, w) \notin J_{i,m-1} \cup Y_{i,m-1}$ this implies $R(v, w) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,m-1})$, and so there exist a (finite) chain of R facts in $X_{i,m-1}$ that connect v and w.

Let $R(a_0, a_1), \ldots, R(a_{l-1}, a_l)$ be such a chain where $v = a_0$ and $w = a_l$ hold. It can then be shown by induction on $0 \leq k \leq l$ that $R(u, a_k)$ is added to Q at some point during the execution of some $C_{i,m'}$ with $1 \leq m' \leq m$. Assume without loss of generality that $R(u, w)$ is added to Q at some point during the execution of $C_{i,n}$ with $1 \leq n \leq m$. Then, Definition 19 and $F = R(u, w) \notin \mathsf{mat}(\Pi, E_i)$ jointly imply $F \in \Delta^{\mathsf{p}}_{i,n} \cup J_{i,n}$. Since $F \in I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$ holds, we have $n = m = j$. Then, $F \notin \Delta^{\mathsf{p}}_{i,j}$ implies $F \in J_{i,j}$, as required.

- $m > 1$ and there exists n with $1 \leq n < m$ such that $R(v, w)$ is added to Q at some point during the execution of $C_{i,n}$. If $n = 1$ holds, then similarly to the case of $m = 1$ we have $F \in \Delta^{\mathsf{p}}_{i,1} \cup J_{i,1}$, which is a contradiction. Otherwise, $R(u, v) \in \Delta^{\mathsf{m}}_{i,j} \subseteq I^{\mathsf{p}}_{i,j} \subseteq I^{\mathsf{p}}_{i,m} \subseteq I^{\mathsf{p}}_{i,n} = I^{\mathsf{p}}_{i,n-1} \setminus (\Delta^{\mathsf{p}}_{i,n-1} \cup \Delta^{\mathsf{m}}_{i,n-1})$ follows from Definition 15. But then, property (B.28) clearly ensures $R(u, v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,n-1}) \cup Y_{i,n-1} \cup J_{i,n-1}$. Note that m is the smallest index such that $R(u, v)$ is added to Q at some point during the execution of $C_{i,m}$, and so $n < m$ implies $R(u, v) \notin Y_{i,n-1} \cup J_{i,n-1}$. Consequently, there exist a finite chain of R facts in $X_{i,n-1}$ that connect u and v. $R(v, w)$ is added to Q during the execution of $C_{i,n}$, and so in a similar way as in the previous case there exists index n', $1 \leq n' \leq n$ such that $R(u, w) \in \Delta^{\mathsf{p}}_{i,n'} \cup J_{i,n'}$ holds, which is a contradiction to $R(u, w) \in I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$.

– There exist $R(t, u) \in I^{\mathsf{p}}_{i,j}$ and $R(u, v) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$ such that $F = R(t, v)$ holds. The proof is analogous to the first case, so we omit the details.

Finally, we show that for each $1 \leq j \leq n_i$, the following statement holds: if call $C_{i,j}$ is of type Del, then property (B.31) holds.

$$J_{i,j} \subseteq \mathsf{tc}(R)^{\infty}_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \tag{B.31}$$

To this end, consider arbitrary j with $1 \leq j \leq n_i$ such that call $C_{i,j}$ is of type Del. Let L be the total number of iterations for the loop of lines 200–209; moreover, let J^0 and

Q^0 be the value of J and Q, respectively, right before line 200, and for each $1 \leq l \leq L$, let J^l and Q^l be the value of J and Q, respectively, right after the lth iteration of lines 200–209; finally, let Δ_i be defined with respect to $\mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$ as in Definition 10. We show by induction on l that $J^l \subseteq \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$ and $Q^l \subseteq \Delta^\mathsf{p}_{i,j} \cup \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$ holds.

- For the induction base where $l = 0$, we have $J^0 = \emptyset$ and $Q^0 = R[\Delta^\mathsf{p}_{i,j}]$, and so the required conditions clearly hold.
- For the inductive step, consider arbitrary $l > 0$ such that J^{l-1} and Q^{l-1} satisfy the required conditions. Let $R(u, w)$ be an arbitrary fact in Q^l. If $R(u, w) \in Q^{l-1}$ holds, then by the induction assumption $R(u, w) \in \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$ clearly holds. Otherwise, $R(u, w)$ is added to Q^l in the lth iteration of lines 200–209. There are two possibilities.
  - There exist $R(u, v) \in Q^{l-1}$ and $R(v, w) \in X_{i,j} \cup (\Delta^\mathsf{p}_{i,j} \cup J_{i,j}) \subseteq I^\mathsf{p}_{i,j}$ such that $R(u, w) \in I^\mathsf{p}_{i,j} \setminus \Delta^\mathsf{p}_{i,j}$ holds. Then, by the induction assumption for $l - 1$ and the definition of $\mathsf{tc}(R)^\infty_\downarrow$ we have $R(u, w) \in \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$.
  - There exist $R(u, v) \in I^\mathsf{p}_{i,j}$ and $R(v, w) \in Q^{l-1}$ such that $R(u, w) \in I^\mathsf{p}_{i,j} \setminus \Delta^\mathsf{p}_{i,j}$ holds. Similarly to the first case, we have $R(u, w) \in \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$.

  The choice of $R(u, w)$ is arbitrary, and so $Q^l \subseteq \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$ holds, as required. Now consider arbitrary $R(u, w) \in J^l$. If $R(u, w) \in J^{l-1}$ holds, then by the induction assumption for $l - 1$ we have $R(u, w) \in \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$; otherwise, $R(u, w)$ is added to J^l in the lth iteration of lines 200–209, and so $R(u, w) \in Q^l \subseteq \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$ holds.

This completes our inductive proof. $J^l \subseteq \mathsf{tc}(R)^\infty_\downarrow[I^\mathsf{p}_{i,j}, I^\mathsf{n}_{i,j} \colon \Delta^\mathsf{p}_{i,j} \cup \Delta^\mathsf{m}_{i,j}, \Delta^\mathsf{n}_{i,j}]$ holds for $l = L$, and so property (B.31) holds. □

Claim 60. *For each $1 \leq i \leq m$, if properties (a)–(d) holds for $i - 1$, then property (b) holds for i as well.*

Proof. Consider arbitrary i with $1 \leq i \leq m$ such that properties (a)–(d) hold for $i - 1$, we show that property (b) holds for i as well. In particular, we prove that the following

statement is true for each $1 \leq j \leq n_i$: if call $C_{i,j}$ is of type Red, then property (B.32) holds.

$$\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) = R[I^{\mathsf{p}}_{i,j} \cup J_{i,j}] \tag{B.32}$$

Note that Definition 16 ensures that there exists at most one index k with $1 \leq k \leq n_i$ such that $C_{i,k}$ is of type Red. Therefore, if no such k exists, then the statement mentioned above clearly holds for each $1 \leq j \leq n_i$. Otherwise, the statement holds for each j with $1 \leq j \leq n_i$ and $j \neq k$, and we show that $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) = R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$ holds.

For the $\subseteq$ direction, Definition 16 ensures that for each $1 \leq j < k$, call $C_{i,j}$ is of type Del, and so by property (B.21) we have $X_{i,k-1} \subseteq I^{\mathsf{p}}_{i,k-1} \setminus (\Delta^{\mathsf{p}}_{i,k-1} \cup \Delta^{\mathsf{m}}_{i,k-1})$. Then, Definitions 15 and 16 jointly imply $X_{i,k-1} \subseteq I_{i-1,n_{i-1}}$. Moreover, the way Y_R is updated in lines 205 and 209, and Definition 16 jointly imply $Y_{i,k-1} \subseteq I_{i-1,n_{i-1}}$. Consequently, line 212 ensures $X_{i,k} = X_{i,k-1} \cup Y_{i,k-1} \subseteq I_{i-1,n_{i-1}}$. Call $C_{i-1,n_{i-1}}$ is of type Add, and so we have $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) \subseteq \mathsf{mat}(\mathsf{tc}(R), I_{i-1,n_{i-1}} \cup J_{i-1,n_{i-1}}) = I_{i-1,n_{i-1}} \cup J_{i-1,n_{i-1}}$. Definition 16 then ensures $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) \subseteq I^{\mathsf{p}}_{i,1}$. We next show that $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) \setminus R[I^{\mathsf{p}}_{i,k}] \subseteq R[J_{i,k}]$ holds. To this end, consider arbitrary $R(u,v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k}) \setminus R[I^{\mathsf{p}}_{i,k}]$. Then, $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) \subseteq I^{\mathsf{p}}_{i,1}$ and Definition 15 ensure $R(u,v) \in \Delta_{i,k}$. Consequently, $R(u,v)$ is considered in line 213 during the execution of $C_{i,k}$. Since $R(u,v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$ holds, v is clearly reachable from u via R facts in $X_{i,k}$, and so $R(u,v)$ is added to J in line 215. But then, $R(u,v) \in \Delta_{i,k}$ and line 216 jointly imply $R(u,v) \in R[J_{i,k}]$. The choice of $R(u,v)$ is arbitrary, so $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) \setminus R[I^{\mathsf{p}}_{i,k}] \subseteq R[J_{i,k}]$ holds, which implies $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) \subseteq R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$, as required.

For the $\supseteq$ direction, Definition 15 ensures that call $C_{i,k-1}$ is of type Del, and so $R[I^{\mathsf{p}}_{i,k-1} \setminus (\Delta^{\mathsf{p}}_{i,k-1} \cup \Delta^{\mathsf{m}}_{i,k-1} \cup J_{i,k-1})] \subseteq \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$ follows from Claim 59. By Definition 15 we have $R[I^{\mathsf{p}}_{i,k}] \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$. Moreover, due to line 214 each fact $R(u,v)$ in $J_{i,k}$ must satisfy $R(u,v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$. Thus, $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) \supseteq R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$ holds.

Next we show that $\mathsf{tc}(R)\left[I^{\mathsf{p}}_{i,k}, I^{\mathsf{n}}_{i,k}\right] \cap \Delta_{i,k} \subseteq J_{i,k}$ holds. To this end, consider arbitrary $R(u,w) \in \mathsf{tc}(R)\left[I^{\mathsf{p}}_{i,k}, I^{\mathsf{n}}_{i,k}\right] \cap \Delta_{i,k}$. $R(u,w) \in \Delta_{i,k}$ implies that u will be considered in line 213; moreover, we have already shown in the previous paragraph

that $R[I^{\mathsf{p}}_{i,k}] \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$ holds, and so there exist a finite chain of R facts in $X_{i,k}$ that connect u and w. Consequently, $R(u, w)$ will be added to J in line 215, and so $R(u, w) \in \Delta_{i,k}$ implies $R(u, w) \in J_{i,k}$, as required.

Finally, we show that $J_{i,j} \subseteq \mathsf{mat}(\Pi, E_i) \cap \Delta_{i,j}$ holds. Consider arbitrary $R(u, w) \in J_{i,j}$, line 216 ensures $R(u, w) \in \Delta_{i,j}$. Next we prove that $R(u, w) \in \mathsf{mat}(\Pi, E_i)$ holds. Due to lines 214 and 215 there exist a chain of R facts in $X_{i,k}$ that connect u and w. By property (d) for $i - 1$ and condition (5.12) we have $X_{i,k-1} \subseteq U_i$; Definition (19) implies $Y_{i,k-1} \subseteq \mathsf{mat}(\Pi, E_i)$. Consequently, we have $X_{i,k} = X_{i,k-1} \cup Y_{i,k-1} \subseteq \mathsf{mat}(\Pi, E_i)$, and so we have $R(u, w) \in \mathsf{mat}(\Pi, E_i)$, as required. □

Claim 61. *For each $1 \leq i \leq m$, if properties (a)–(d) holds for $i - 1$, then property (c) holds for i as well.*

Proof. Consider arbitrary i with $1 \leq i \leq m$ such that properties (a)–(d) hold for $i - 1$, we show that property (c) holds for i as well. In particular, we prove that the following statement is true for each $1 \leq j \leq n_i$: if call $C_{i,j}$ is of type Add, then properties (B.33)–(B.35) holds.

$$\mathsf{mat}(\mathsf{tc}(R), X_{i,j}) = R[I_{i,j} \cup J_{i,j}] \tag{B.33}$$

$$X_{i,j} \subseteq I_{i,j} \tag{B.34}$$

$$\mathsf{tc}(R)^{\uparrow}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \subseteq J_{i,j} \subseteq \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \tag{B.35}$$

Note that Definition 16 ensures that there exists at most one index k with $1 \leq k \leq n_i$ such that $C_{i,k}$ is of type Red. If no such k exists, then by Definition 15 each call $C_{i,j}$ with $1 \leq j \leq n_i$ is of type Del, and so the statement trivially holds for each j. If such a k does exist, then the statement is clearly true for each j with $1 \leq j \leq k$ since call $C_{i,j}$ is not of type Add; in constrast, each call $C_{i,j}$ with $k < j \leq n_i$ is of type Add by Definitions 15 and 16, and we first show by induction on j that property (B.33) holds.

- For the induction base where $j = k + 1$, note that call $C_{i,k}$ is of type Red, and so by Claim 60 we have $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) = R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$. We would like to show that $\mathsf{mat}(\mathsf{tc}(R), X_{i,k+1}) = R[I^{\mathsf{p}}_{i,k+1} \cup J_{i,k+1}]$.

– For the $\subseteq$ direction, first note that for each $R(u, v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$, u and v can be connected using a chain of R facts in $X_{i,k+1}$. We show by induction on the length of such chains that $\mathsf{mat}(\mathsf{tc}(R), X_{i,k+1}) \subseteq R[I_{i,k+1} \cup J_{i,k+1}]$ holds, and that each fact in $\mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$ is either in $\mathsf{mat}(\mathsf{tc}(R), X_{i,k})$ or added to Q at some point in the call of $C_{i,k+1}$.

  * For the base case, consider arbitrary $R(u, v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$ where a chain of length one exists in $X_{i,k+1}$. This implies $R(u, v) \in X_{i,k+1}$. If $R(u, v) \in X_{i,k}$, then $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) = R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$ and Definition 15 imply $R(u, v) \in R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}] \subseteq R[I_{i,k+1}]$. Otherwise, line 191 ensures that $R(u, v) \in R[\Delta^{\mathsf{p}}_{i,k+1}] \subseteq R[I_{i,k+1}]$ holds, and that $R(u, v)$ is added to Q.

  * For the inductive step, consider arbitrary l that satisfies the following condition: for each $R(u, v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$ where u and v can be connected using $(l - 1)$ or less facts in $X_{i,k+1}$, $R(u, v) \in R[I_{i,k+1} \cup J_{i,k+1}]$ holds, and if $R(u, v) \notin \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$, then it is added to Q at some point in the call of $C_{i,k+1}$. Let $R(a_0, a_l) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$ be an arbitrary fact for which a chain of l facts, $R(a_0, a_1), \ldots, R(a_{l-1}, a_l) \in X_{i,k+1}$, exists. There are three possibilities.

    · If $R(a_0, a_1) \in X_{i,k}$ and $R(a_1, a_l) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$, then Definition 15 and $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) = R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$ imply $R(a_0, a_l) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$, which in turn implies $R(a_0, a_l) \in R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}] \subseteq R[I_{i,k+1}]$.

    · If $R(a_0, a_1) \in X_{i,k+1} \setminus X_{i,k}$ and $R(a_1, a_l) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$, then we have $R(a_1, a_l) \in R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$. By Definition 15 we have $R(a_1, a_l) \in I_{i,k+1}$, and so there are two cases. If $R(a_1, a_l) \in I_{i,k+1} \setminus \Delta^{\mathsf{p}}_{i,k+1}$, then, lines 192 and 193 ensure $R(a_0, a_l) \in R[I_{i,k+1} \cup J_{i,k+1}]$; otherwise, $R(a_1, a_l) \in \Delta^{\mathsf{p}}_{i,k+1}$ holds, and so we have $R(a_0, a_l) \in R[I_{i,k+1} \cup J_{i,k+1}]$ due to lines 194–197.

    · If $R(a_0, a_1) \in X_{i,k+1}$ and $R(a_1, a_l) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1}) \setminus \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$. But then, by the induction assumption for $l - 1$ fact $R(a_1, a_l)$ is added to Q at some point. Consequently, when $R(a_1, a_l)$ is considered in the loop of lines 194–197, line 196 and 197 ensure $R(a_0, a_l) \in R[I_{i,k+1} \cup J_{i,k+1}]$.

Finally, if $R(a_0, a_l) \notin \mathsf{mat}(\mathsf{tc}(R), X_{i,k})$, then $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) = R[I_{i,k} \cup J_{i,k}]$ and $R(a_0, a_l) \in R[I_{i,k+1} \cup J_{i,k+1}]$ jointly imply $R(a_0, a_l) \in \Delta^{\mathsf{p}}_{i,k+1} \cup J_{i,k+1}$. Line 191, and the fact that Q and J are always updated together imply that $R(a_0, a_l)$ is added to Q at some point as well.

– For the $\supseteq$ direction, $\mathsf{mat}(\mathsf{tc}(R), X_{i,k}) = R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$ and the way X_R is updated imply $R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}] \subseteq \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$; moreover, line 191 ensures $R[\Delta^{\mathsf{p}}_{i,k+1}] \subseteq X_{i,k+1} \subseteq \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$; finally, it can be shown by a straightforward induction on the construction of $J_{i,k+1}$ that $R[J_{i,k+1}] \subseteq \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$ holds. Consequently, $R[I_{i,k+1} \cup J_{i,k+1}] \subseteq \mathsf{mat}(\mathsf{tc}(R), X_{i,k+1})$ holds, as required.

- For the inductive step, consider arbitrary $j > k + 1$ such that property (B.33) holds for $j - 1$, and we show that the property holds for j as well. The proof is analogous to the above, so we omit the details.

Now we show by induction on j with $k < j \leq n_i$ that property (B.34) holds.

- For the induction base where $j = k + 1$, note that call $C_{i,j-1}$ is of type Red, and call $C_{i,j-2}$ is of type Del. Thus, by Claim 59 we have $X_{i,j-2} \subseteq I^{\mathsf{p}}_{i,j-2} \setminus (\Delta^{\mathsf{p}}_{i,j-2} \cup \Delta^{\mathsf{m}}_{i,j-2})$. Then, by Definition 15 we have $X_{i,j-2} \subseteq I^{\mathsf{p}}_{i,j-1} \subseteq I_{i,j}$. Since X_R is not modified in Algorithm 10, we have $X_{i,j-1} \subseteq I_{i,j}$. Now consider Add call $C_{i,j}$ and arbitrary $F \in X_{i,j}$. If $F \in X_{i,j-1}$ holds, then $X_{i,j-1} \subseteq I_{i,j}$ implies $F \in I_{i,j}$. Otherwise, F is added to $X_{i,j}$ in line 191, which implies $F \in \Delta^{\mathsf{p}}_{i,j}$. But then, Definition 11 ensures $\Delta^{\mathsf{p}}_{i,j} \subseteq I_{i,j}$, and so $F \in I_{i,j}$ holds as well. The choice of F is arbitrary, so $X_{i,j} \subseteq I_{i,j}$ holds for $i = k + 1$.

- For the inductive step, consider arbitrary $j > k + 1$ and the corresponding Add call $C_{i,j}$ such that $X_{i,j-1} \subseteq I_{i,j-1}$ holds. Then, consider arbitrary fact $F \in X_{i,j}$. There are two cases. If $F \in X_{i,j-1}$, then the induction assumption for $j - 1$ and Definition 15 jointly imply $F \in I_{i,j}$. Otherwise, F is added to $X_{i,j}$ in line 191, and so we have $F \in \Delta^{\mathsf{p}}_{i,j} \subseteq I_{i,j}$. Since F is chosen arbitrarily, we have $X_{i,j} \subseteq I_{i,j}$, as required.

We next prove that the left-hand side $\subseteq$ of property (B.35) holds for each $k < j \leq n_i$. To this end, we first consider $j = k + 1$ and the corresponding Add call $C_{i,j}$. Let F be an arbitrary fact in $\mathsf{tc}(R)^{\uparrow}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$. By the definition of $\mathsf{tc}(R)^{\uparrow}$ we have

$F \in \mathsf{tc}(R)\left[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}\right] \setminus I_{i,j}$. Since $\mathsf{tc}(R)$ involves no negation in the body of its rules, $F \in \mathsf{tc}(R)\left[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}\right] \setminus I_{i,j}$ holds. We discuss the following possibilities.

- There exist $R(u,v) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$ and $R(v,w) \in I_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j})$ such that $F = R(u,w)$ holds. There are two cases.
  - If $R(u,v) \in \Delta^{\mathsf{p}}_{i,j}$, then lines 192 and 193 ensure that $F \in J_{i,j}$ holds.
  - If $R(u,v) \in \Delta^{\mathsf{m}}_{i,j}$, then by Definition 15 we have $R(u,v) \in J_{i,j-1}$. Moreover, Definition 15 implies $R(v,w) \in I_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}) = I^{\mathsf{p}}_{i,j-1}$. By Claim 60 we have $R(u,v) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,j-1})$ and $R(v,w) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,j-1})$; consequently, $R(u,w) \in \mathsf{mat}(\mathsf{tc}(R), X_{i,j-1}) = R[I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}]$, which is a contradiction to $R(u,w) \notin I_{i,j}$.
- There exist $R(u,v) \in I_{i,j}$ and $R(v,w) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$ such that $F = R(u,w) \notin I_{i,j}$ holds. There are several cases.
  - If $R(v,w) \in \Delta^{\mathsf{m}}_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$ and $R(u,v) \in \Delta^{\mathsf{p}}_{i,j}$ hold, then $R(v,w) \in I_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$ follows from Definition 11. But then, lines 192 and 193 ensure that $F \in J_{i,j}$ holds.
  - If $R(v,w) \in \Delta^{\mathsf{m}}_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$ and $R(u,v) \in I_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$ hold, then Definition 15 ensures $R(v,w) \in J_{i,j-1}$ and $R(u,v) \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}$. Consequently, by Claim 60 we have $F = R(u,w) \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}$, which is a contradiction to $R(u,w) \notin I_{i,j}$.
  - If $R(v,w) \in \Delta^{\mathsf{p}}_{i,j}$ and $R(u,v) \in \Delta^{\mathsf{p}}_{i,j}$ hold, then line 191 ensures that $R(u,v) \in X_{i,j}$ holds, and that $R(v,w)$ is added to Q. Consequently, $R(v,w)$ will be examined in line 195, and so lines 196 and 197 ensure $F = R(u,w) \in J_{i,j}$.
  - If $R(v,w) \in \Delta^{\mathsf{p}}_{i,j}$ and $R(u,v) \in I_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$, then $R(u,v) \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}$ follows from Definition 15. By Claim 60 there exist a chain of R facts in $X_{i,j-1}$ that connects u and v. Let $R(u,a_n), R(a_1,a_{n-1}), \ldots, R(a_1,v)$ be such a chain. We show by induction on $1 \leq m \leq n$ that $R(a_m,w) \in \Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}$ holds.
    - For the induction base where $m = 1$, line 191 ensures that $R(v,w) \in \Delta^{\mathsf{p}}_{i,j}$ is added Q; moreover, $R(a_1,v) \in X_{i,j-1} \subseteq X_{i,j}$ holds. Consequently, $R(a_1,w)$ will be examined in line 196. If $R(a_1,w) \notin I_{i,j}$, we have $R(a_1,w) \in J_{i,j}$. Otherwise, assume for the sake of a contradiction that $R(a_1,w) \in I_{i,j} \setminus \Delta^{\mathsf{p}}_{i,j}$

holds, then by Definition 15 we have $R(a_1, w) \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}$. Consequently, $R(u, w) \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}$ holds, which is a contradiction. Therefore, we have $R(a_1, w) \in \Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}$.

- For the inductive step, consider arbitrary $m > 1$ such that $R(a_{m-1}, w)$ belongs to $\Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}$. Then, $R(a_{m-1}, w)$ is added to Q at some point during the execution of the algorithm. In the same way as in the base case we have $R(a_m, w) \in \Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}$, as required.

$R(a_n, w) \in \Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}$ and $R(u, a_n) \in X_{i,j-1}$ jointly imply $R(u, w) \in \Delta^{\mathsf{p}}_{i,j} \cup J_{i,j}$. But then, $R(u, w) \notin I_{i,j}$ and $\Delta^{\mathsf{p}}_{i,j} \subseteq I_{i,j}$ ensure $R(u, w) \in J_{i,j}$.

The proof for $k + 1 < j \leq n_i$ is analogous to the above, so we omit the details.

Finally, we show that the right-hand side $\subseteq$ of property (B.35) holds for each $k < j \leq n_i$. To this end, consider arbitrary j with $k < j \leq n_i$ and the corresponding Add call $C_{i,j}$. Let L be the total number of iterations for the loop of lines 194–197; moreover, let J^0 and Q^0 be the value of J and Q, respectively, right before line 194, and for each $1 \leq l \leq L$, let J^l and Q^l be the value of J and Q, respectively, right after the lth iteration of lines 194–197; finally, let Δ_i be defined with respect to $\mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$ as in Definition 9. We show by induction on l that $J^l \subseteq \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$ holds.

- For the induction base where $l = 0$, consider arbitrary $R(u, w) \in J^0$. Lines 192 and 193 ensure that there exist $R(u, v) \in \Delta^{\mathsf{p}}_{i,j}$ and $R(v, w) \in I_{i,j}$ such that $R(u, w) \notin I_{i,j}$ holds. Consequently, $R(u, w) \in \mathsf{tc}(R)^{\uparrow}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \subseteq \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$ holds, as required.

- For the inductive step, consider arbitrary $l > 0$ such that J^{l-1} is contained in $\mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$. Let $R(u, w)$ be an arbitrary fact in J^l. If $R(u, w) \in J^{l-1}$ holds, then $R(u, w) \in \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$ follows from the induction assumption. Otherwise, $R(u, w)$ is added to J^l in the lth iteration of lines 194–197. Consequently, there exist $R(u, v) \in X_{i,j} \subseteq I_{i,j}$ and $R(v, w) \in Q^{l-1}$ such that $R(u, w) \notin I_{i,j} \cup J^{l-1}$ holds. Lines 191, 193, and 197 imply $R(v, w) \in R[\Delta^{\mathsf{p}}_{i,j}] \cup J^{l-1}$. We discuss the following possibilities.

– If $R(v,w) \in R[\Delta^{\mathsf{p}}_{i,j}]$ holds, then $R(u,v) \in I_{i,j}$ and $R(u,w) \notin I_{i,j}$ jointly imply $R(u,w) \in \mathsf{tc}(R)^{\uparrow}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \subseteq \mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$, as required.

– If $R(v,w) \in J^{l-1}$, then by the induction assumption for $l-1$ and the definition of $\mathsf{tc}(R)^{\uparrow}_{\infty}$ there exists index l' such that $R(v,w) \in \Delta_{l'}$ holds. Consequently, $R(u,v) \in I_{i,j}$, $R(u,w) \notin I_{i,j}$, and the definition of $\mathsf{tc}(R)^{\uparrow}_{\infty}$ jointly imply $R(u,w) \in \bigcup_{1 \leq i \leq l'+1} \Delta_i$, which in turn implies that $R(u,w)$ belongs to $\mathsf{tc}(R)^{\uparrow}_{\infty}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$, as required. □

Claim 62. *For each $1 \leq i < m$, if properties (a)–(d) holds for $i-1$, then property (d) holds for i as well.*

Proof. Consider arbitrary i with $1 \leq i < m$ such that properties (a)–(d) hold for $i-1$, we show that property (d) holds for i as well. By the induction assumption we have $X_{i-1,n_{i-1}} \subseteq U_{i-1}$. Definition 16 implies that there exist exactly one k with $1 \leq k \leq n_i$ such that condition (5.12) holds, and so the way X_R is updated in line 210 ensures $X_{i,k-1} \subseteq U_i$. Furthermore, Definition 19 and $\mathsf{tc}(R) \subseteq \Pi^s$ jointly imply $Y_{i,k-1} \subseteq U_i$. Therefore, line 212 ensures $X_{i,k} = X_{i,k-1} \cup Y_{i,k-1} \subseteq U_i$. Together with condition (5.13) and the way X_R is updated in line 191 we have $X_{i,n_i} \subseteq U_i$, as required. □

B.5 Proof of Theorem 21

Theorem 21. *Functions $\mathsf{Add}^{\mathsf{stc}(R)}$, $\mathsf{Del}^{\mathsf{stc}(R)}$, and $\mathsf{Red}^{\mathsf{stc}(R)}$ are correct for the module that axiomatises R as symmetric–transitive, provided that the oracle function T is correct.*

Let R be an arbitrary binary relation, and let $\mathsf{stc}(R)$ be the module that axiomatises R as symmetric-transitive. For C_R a set of sets used in Algorithms 11 and 12 to represent connected components, let $\mathsf{Close}(C_R) := \bigcup_{U \in C_R} \bigcup_{u,v \in U} \{R(u,v)\}$. Now consider arbitrary program Π, stratification λ of Π, call history H for $\mathsf{stc}(R)$ of the form (5.6), and vector $\vec{E} = E_0, \ldots, E_m$ of datasets such that Π, λ, $\mathsf{stc}(R)$, H, and $\vec{E}$ are compatible; let s be the stratum index such that $\mathsf{stc}(R) \subseteq \Pi^s$ holds. Moreover, for each $0 \leq i \leq m$ and each $1 \leq j \leq n_i$, let $C_R^{i,j}$ and $Y_R^{i,j}$ be the values of C_R and Y_R, respectively, after the call of

$C_{i,j}$. We show by induction on i with $0 \leq i \leq m$ that the following properties hold. Then, Definition 17 implies the correctness of the theorem.

(a) For each $1 \leq j \leq n_i$, if call $C_{i,j}$ is of type Del, then the following properties hold.

$$R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup J_{i,j})] \subseteq \mathsf{Close}(C^{i,j}_R) \cup Y^{i,j}_R \tag{B.36}$$

$$\mathsf{Close}(C^{i,j}_R) \subseteq I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup J_{i,j}) \tag{B.37}$$

$$\begin{aligned}\mathsf{stc}(R)_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \setminus \mathsf{mat}(\Pi, E_i) &\subseteq J_{i,j} \\ \subseteq \mathsf{stc}(R)^{\infty}_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]&\end{aligned} \tag{B.38}$$

(b) For each $1 \leq j \leq n_i$, if call $C_{i,j}$ is of type Red, then the following properties hold.

$$\mathsf{Close}(C^{i,j}_R) = R[I^{\mathsf{p}}_{i,j} \cup J_{i,j}] \tag{B.39}$$

$$\mathsf{stc}(R)\left[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j}\right] \cap \Delta_{i,j} \subseteq J_{i,j} \subseteq \mathsf{mat}(\Pi, E_i) \cap \Delta_{i,j} \tag{B.40}$$

(c) For each $1 \leq j \leq n_i$, if call $C_{i,j}$ is of type Add, then the following properties hold.

$$\mathsf{Close}(C^{i,j}_R) = R[I_{i,j} \cup J_{i,j}] \tag{B.41}$$

$$\mathsf{stc}(R)^{\uparrow}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \subseteq J_{i,j} \subseteq \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{i,j} : \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \tag{B.42}$$

The proof is lengthy, so we break it into several claims.

Claim 63. *Properties (a)–(c) hold for $i = 0$.*

Proof. H is a call history, so by Definition 15 each call $C_{0,j}$ with $1 \leq j \leq n_0$ is of type Add. Therefore, properties (a) and (b) trivially hold. We next show by induction on j with $1 \leq j \leq n_0$ that $\mathsf{Close}(C^{0,j}_R) = R[I_{0,j} \cup J_{0,j}]$ holds.

- For the induction base where $j = 1$, we show that $\mathsf{Close}(C^{0,1}_R) = R[I_{0,1} \cup J_{0,1}]$ holds.
  - For the $\subseteq$ direction, consider arbitrary $R(u, v) \in \mathsf{Close}(C^{0,1}_R)$. There are two possibilities. For the case where $u = v$, note that C_R is empty before the call $C_{0,1}$, and so u can only be introduced into $C^{0,1}_R$ via line 222 or line 224. In the first case, lines 222 and 217 ensure $R(u, v) \in R[I_{0,1} \cup J_{0,1}]$; in the second case, lines 224 and 217 ensure $R(u, v) \in R[I_{0,1} \cup J_{0,1}]$. For the case where $u \neq v$, the definition of Close implies that there exists a component $P \in C^{0,1}_R$ such

that $u \in P$ and $v \in P$ hold. Since C_R is empty before the call, there exist a component U containing u and a component V containing v where U and V are merged in line 226 during the execution of $C_{0,1}$. But then, lines 227, 228 and 217 jointly ensure $R(u, v) \in R[I_{0,1} \cup J_{0,1}]$. The choice of $R(u, v)$ is arbitrary, so the $\subseteq$ direction holds.

– For the $\supseteq$ direction, consider arbitrary $R(u, v) \in R[I_{0,1} \cup J_{0,1}]$. Definition 16 ensures $R(u, v) \in R[(\Delta^{\mathsf{p}}_{0,1} \setminus \Delta^{\mathsf{m}}_{0,1}) \cup J_{0,1}]$. There are two posssibilities.

  * For the case where $R(u, v) \in R[\Delta^{\mathsf{p}}_{0,1} \setminus \Delta^{\mathsf{m}}_{0,1}]$, the fact will be considered in line 220. Then, lines 221–226 ensure that u and v are in the same component in $C^{0,1}_R$, and so $R(u, v) \in \mathsf{Close}(C^{0,1}_R)$ holds.

  * For the case where $R(u, v) \in R[J_{0,1}]$, note that the fact can only be added to $J_{0,1}$ via line 222, 224, or 228. In each of these cases, u and v will be in the same component in $C^{0,1}_R$, and so $R(u, v) \in \mathsf{Close}(C^{0,1}_R)$ holds.

  The choice of $R(u, v)$ is arbitrary, so the $\supseteq$ direction holds as well.

- For the inductive step, consider arbitrary index j with $1 < j \leq n_0$ such that $\mathsf{Close}(C^{0,j-1}_R) = R[I_{0,j-1} \cup J_{0,j-1}]$ holds, and we show that $\mathsf{Close}(C^{0,j}_R) = R[I_{0,j} \cup J_{0,j}]$ holds as well.

  – For the $\subseteq$ direction, consider an arbitrary fact $R(u, v)$ that belongs to $\mathsf{Close}(C^{0,j}_R)$. If $R(u, v) \in \mathsf{Close}(C^{0,j-1}_R)$, then $R(u, v) \in R[I_{0,j-1} \cup J_{0,j-1}]$ follows from the induction assumption; Definition 15 ensures $I_{0,j} = I_{0,j-1} \cup J_{0,j-1} \cup \Delta^{\mathsf{p}}_{0,j}$, and so we have $R(u, v) \in R[I_{0,j} \cup J_{0,j}]$. Otherwise, we have $R(u, v) \in \mathsf{Close}(C^{0,j}_R) \setminus \mathsf{Close}(C^{0,j-1}_R)$. Now if $u = v$, then u belongs to a component in $C^{0,j}_R$ but does not belong to any component in $C^{0,j-1}_R$, and so u is introduced into $C^{0,j}_R$ in either line 222 or line 224. Either way, $R(u, v) \in R[I_{0,j} \cup J_{0,j}]$ holds. If $u \neq v$, then u and v are put into the same component in line 226, and so $R(u, v) \in R[I_{0,j} \cup J_{0,j}]$ holds as well. The choice of $R(u, v)$ is arbitrary, so the $\subseteq$ direction holds, as required.

  – For the $\supseteq$ direction, consider arbitrary $R(u, v) \in R[I_{0,j} \cup J_{0,j}]$. Then, Definition 15 implies $R(u, v) \in R[I_{0,j-1} \cup J_{0,j-1} \cup \Delta^{\mathsf{p}}_{0,j} \cup J_{0,j}]$. There are three possibilities.

* If $R(u,v) \in R[I_{0,j-1} \cup J_{0,j-1}]$, then $R(u,v) \in \mathsf{Close}(C_R^{0,j-1}) \subseteq \mathsf{Close}(C_R^{0,j})$ follows from the induction assumption for $j-1$ and the way C_R is updated in Algorithm 11.

* If $R(u,v) \in R[\Delta_{0,j}^{\mathsf{p}} \setminus J_{0,j-1}] = R[\Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}]$, it will be considered in line 220 during the execution of $C_{0,j}$. If $u = v$, lines 221 and 222 ensure $R(u,v) \in \mathsf{Close}(C_R^{0,j})$; if $u \neq v$, lines 225 and 226 ensure that u and v are in the same component in $C_R^{0,j}$, and so $R(u,v) \in \mathsf{Close}(C_R^{0,j})$ holds.

* If $R(u,v) \in R[J_{0,j}]$, then it must be added to $J_{0,j}$ via line 222, 224, or 228. In each of these cases, u and v will be guaranteed to be in the same component in $C_R^{0,j}$, and so $R(u,v) \in \mathsf{Close}(C_R^{0,j})$ holds.

The choice of $R(u,v)$ is arbitrary, so the $\supseteq$ direction holds as well.

We next prove the left-hand side $\subseteq$ of property (B.42) for $i = 0$ and $1 \leq j \leq n_0$.

- For $i = 0$ and $j = 1$, consider arbitrary $R(u,w) \in \mathsf{stc}(R)^{\uparrow}[I_{0,1} \vdots \Delta_{0,1}^{\mathsf{p}} \cup \Delta_{0,1}^{\mathsf{m}}, \Delta_{0,1}^{\mathsf{n}}]$. By the definition of $\mathsf{stc}(R)^{\uparrow}$ we have $R(u,w) \in \mathsf{stc}(R)\Big[I_{0,1} \vdots \Delta_{0,1}^{\mathsf{p}} \cup \Delta_{0,1}^{\mathsf{m}}, \Delta_{0,1}^{\mathsf{n}}\Big] \setminus I_{0,1}$. Note that $\mathsf{stc}(R)$ involves no negation; moreover, by Definition 16 we have $\Delta_{0,1}^{\mathsf{m}} = \emptyset$ and $I_{0,1} = \Delta_{0,1}^{\mathsf{p}}$. Consequently, $R(u,w) \in \mathsf{stc}(R)\Big[\Delta_{0,1}^{\mathsf{p}} \setminus \Delta_{0,1}^{\mathsf{m}} \vdots \Delta_{0,1}^{\mathsf{p}} \setminus \Delta_{0,1}^{\mathsf{m}}\Big] \setminus I_{0,1}$ holds, and so we discuss the following possibilities.

  – $R(u,w)$ is derived by the symmetric rule—that is, $R(w,u) \in \Delta_{0,1}^{\mathsf{p}} \setminus \Delta_{0,1}^{\mathsf{m}}$ holds. In this case, $R(w,u)$ is considered at some point in line 220 during the execution of $C_{0,1}$. Then, lines 221–228 ensure that w and u belong to the same component in $C_R^{0,1}$, and so $R(u,w) \in \mathsf{Close}(C_R^{0,1})$ holds. But then, $\mathsf{Close}(C_R^{0,1}) = R[I_{0,1} \cup J_{0,1}]$ and $R(u,w) \notin I_{0,1}$ jointly imply $R(u,w) \in J_{0,1}$.

  – $R(u,w)$ is derived by the transitive rule—that is, there exist v such that $R(u,v) \in \Delta_{0,1}^{\mathsf{p}} \setminus \Delta_{0,1}^{\mathsf{m}}$, $R(v,w) \in \Delta_{0,1}^{\mathsf{p}} \setminus \Delta_{0,1}^{\mathsf{m}}$, and $R(u,w) \notin I_{0,1}$ all hold. There are two cases.

    * $R(u,v) = R(v,w)$. In this case, $R(u,w) \in \Delta_{0,1}^{\mathsf{p}} \setminus \Delta_{0,1}^{\mathsf{m}}$ ensures that $R(u,w)$ is considered at some point in line 220 during the execution of $C_{0,1}$. Conse-

quently, lines 221 and 222 ensure that $R(u,w) \in \mathsf{Close}(C_R^{0,1})$ holds. But then, $\mathsf{Close}(C_R^{0,1}) = R[I_{0,1} \cup J_{0,1}]$ and $R(u,w) \notin I_{0,1}$ jointly imply $R(u,w) \in J_{0,1}$.

* $R(u,v) \neq R(v,w)$. In this case, let m and n be the indexes of the iteration of lines 221–228 in which $R(u,v)$ and $R(v,w)$ are considered, respectively. Without loss of generality we assume that $m < n$ holds. Lines 221–228 ensure that u and v belong to the same component in C_R after the mth iteration; similarly, v and w will belong to the same component after the nth iteration. Consequently, we have $R(u,w) \in \mathsf{Close}(C_R^{0,1})$, and so in the same way as in the previous case we have $R(u,w) \in J_{0,1}$.

- For $i = 0$ and $1 < j \leq n_0$, consider arbitrary $R(u,w) \in \mathsf{stc}(R)^{\uparrow}[I_{0,j} : \Delta_{0,j}^{\mathsf{p}} \cup \Delta_{0,j}^{\mathsf{m}}, \Delta_{0,j}^{\mathsf{n}}]$. By the definition of $\mathsf{stc}(R)^{\uparrow}$ we have $R(u,w) \in \mathsf{stc}(R)\left[I_{0,j} : \Delta_{0,j}^{\mathsf{p}} \cup \Delta_{0,j}^{\mathsf{m}}, \Delta_{0,j}^{\mathsf{n}}\right] \setminus I_{0,j}$. Note that $\mathsf{stc}(R)$ involves no negation, so we have $R(u,w) \in \mathsf{stc}(R)\left[I_{0,j} : \Delta_{0,j}^{\mathsf{p}} \cup \Delta_{0,j}^{\mathsf{m}}\right] \setminus I_{0,j}$. We discuss the following possibilities.
  - $R(w,u) \in \Delta_{0,j}^{\mathsf{p}} \cup \Delta_{0,j}^{\mathsf{m}}$ holds. There are two cases.
    * If $R(w,u) \in \Delta_{0,j}^{\mathsf{m}}$ holds, then by Definition 15 we have $R(w,u) \in J_{0,j-1}$. We have already shown $\mathsf{Close}(C_R^{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$, and so w and u are in one component in $C_R^{0,j-1}$. Thus, $R(u,w) \in \mathsf{Close}(C_R^{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$ holds. Then, Definition 15 ensures $R(u,w) \in I_{0,j}$, which is a contradiction.
    * If $R(w,u) \in \Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}$ holds, then $R(w,u)$ is considered at some point in line 220 during the execution of $C_{0,j}$. Thus, lines 221–228 ensure that w and u are in one component in $C_R^{0,j}$, and so $R(u,w) \in \mathsf{Close}(C_R^{0,j}) = R[I_{0,j} \cup J_{0,j}]$ holds. Then, $R(u,w) \notin I_{0,j}$ implies $R(u,w) \in J_{0,j}$.
  - There exists v such that $R(u,v) \in \Delta_{0,j}^{\mathsf{p}} \cup \Delta_{0,j}^{\mathsf{m}}$ and $R(v,w) \in I_{0,j}$ hold. Then, Definition 15 implies $R(v,w) \in I_{0,j-1} \cup J_{0,j-1} \cup \Delta_{0,j}^{\mathsf{p}}$. There are several cases.
    * If $R(u,v) \in \Delta_{0,j}^{\mathsf{m}}$ and $R(v,w) \in I_{0,j-1} \cup J_{0,j-1}$ hold, then Definition 15 implies $R(u,v) \in J_{0,j-1}$. Now $\mathsf{Close}(C_R^{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$ holds, so $R(u,v), R(v,w) \in \mathsf{Close}(C_R^{0,j-1})$ holds. Thus, u, v, and w are all in one

component in $C_R^{0,j-1}$, so $R(u,w) \in \mathsf{Close}(C_R^{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$ holds. Then, by Definition 15 we have $R(u,w) \in I_{0,j}$, which is a contradiction.

* If $R(u,v) \in \Delta_{0,j}^{\mathsf{m}} = J_{0,j-1}$ and $R(v,w) \in \Delta_{0,j}^{\mathsf{p}} \setminus J_{0,j-1} = \Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}$ hold, then $\mathsf{Close}(C_R^{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$ implies $R(u,v) \in \mathsf{Close}(C_R^{0,j-1})$. Consequently, u and v belong to one component in $C_R^{0,j-1}$. The way C_R is updated in Algorithm 11 ensures that u and v still belong to one component in $C_R^{0,j}$. Now $R(v,w) \in \Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}$ ensures that $R(v,w)$ is considered at some point in line 220 during the execution of $C_{0,j}$, and so lines 221–228 ensure that v and w end up in one component in $C_R^{0,j}$. Since we have already shown that u and v belong to one component in $C_R^{0,j}$, we have $R(u,w) \in \mathsf{Close}(C_R^{0,j}) = R[I_{0,j} \cup J_{0,j}]$. But then, $R(u,w) \notin I_{0,j}$ implies $R(u,w) \in J_{0,j}$, as required.

* If $R(u,v) \in \Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}$ and $R(v,w) \in I_{0,j-1} \cup J_{0,j-1}$ both hold, then $\mathsf{Close}(C_R^{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$ implies $R(v,w) \in \mathsf{Close}(C_R^{0,j-1})$. In the same way as in the previous case, v and w belong to one component in $C_R^{0,j}$; moreover, $R(u,v) \in \Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}$ ensures that u belongs to the same component. Consequently, we have $R(u,w) \in \mathsf{Close}(C_R^{0,j}) = R[I_{0,j} \cup J_{0,j}]$, and so $R(u,w) \notin I_{0,j}$ implies $R(u,w) \in J_{0,j}$, as required.

* If $R(u,v) \in \Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}$ and $R(v,w) \in \Delta_{0,j}^{\mathsf{p}} \setminus J_{0,j-1} = \Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}$ hold, then both $R(u,v)$ and $R(v,w)$ are considered at some point in line 220 during the execution of $C_{0,j}$. Consequently, u, v, and w all belong to one component in $C_R^{0,j}$, and so in a way similar to the previous cases we have $R(u,w) \in J_{0,j}$.

Finally, we prove that the right-hand side $\subseteq$ of property (B.42) holds for $i = 0$ and $1 \leq j \leq n_0$. To this end, consider arbitrary j with $1 \leq j \leq n_0$ and the corresponding Add call $C_{0,j}$. For $1 \leq k \leq |\Delta_{0,j}^{\mathsf{p}} \setminus \Delta_{0,j}^{\mathsf{m}}|$, let J^k be the value of J right after the kth iteration of lines 221–228 during the execution of $C_{0,j}$. Moreover, let J_i be defined with respect to $\mathsf{stc}(R)_\infty^\uparrow[I_{0,j} \,\vdots\, \Delta_{0,j}^{\mathsf{p}} \cup \Delta_{0,j}^{\mathsf{m}}, \Delta_{0,j}^{\mathsf{n}}]$ as in Definition 9. We show by induction on k that $J^k \subseteq I_{0,j} \cup \mathsf{stc}(R)_\infty^\uparrow[I_{0,j} \,\vdots\, \Delta_{0,j}^{\mathsf{p}} \cup \Delta_{0,j}^{\mathsf{m}}, \Delta_{0,j}^{\mathsf{n}}]$ holds. Then, line 217 ensures $J_{0,j} \subseteq \mathsf{stc}(R)_\infty^\uparrow[I_{0,j} \,\vdots\, \Delta_{0,j}^{\mathsf{p}} \cup \Delta_{0,j}^{\mathsf{m}}, \Delta_{0,j}^{\mathsf{n}}]$.

- For the induction base where $k = 1$, let $R(u, v) \in \Delta^{\mathsf{p}}_{0,j} \setminus \Delta^{\mathsf{m}}_{0,j}$ be the fact considered in this iteration of lines 221–228, and consider arbitrary $G \in J^1$. We discuss the following possibilities.
  - G is added to J^1 in line 222, then clearly $G = R(u, u)$ holds. Then, by the definition of J_i we have $R(v, u) \in I_{0,j} \cup J_1$. Consequently, $R(u, v) \in \Delta^{\mathsf{p}}_{0,j} \setminus \Delta^{\mathsf{m}}_{0,j}$ implies that $R(u, u)$ can be derived by the transitive rule, and so we clearly have $G = R(u, u) \in I_{0,j} \cup J_2 \subseteq I_{0,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{0,j} : \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$, as required.
  - G is added to J^1 in line 222. This is analogous to the previous case.
  - G is added to J^1 in line 228. Let $G = R(u', v')$. There are several cases.
    - $u' = u$ and $v' = v$. In this case, $G \in \Delta^{\mathsf{p}}_{0,j} \setminus \Delta^{\mathsf{m}}_{0,j} \subseteq I_{0,j}$ clearly holds.
    - $u' \neq u$ and $v' = v$. Then, u' and u must be put into the same component during a previous call to Algorithm 11. Consequently, we have $R(u', u) \in \mathsf{Close}(C_R^{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$. By Definition 15 we have $R(u', u) \in I_{0,j}$. Together with $R(u, v) \in \Delta^{\mathsf{p}}_{0,j} \setminus \Delta^{\mathsf{m}}_{0,j}$ this clearly implies $R(u', v) \in I_{0,j} \cup J_1 \subseteq I_{0,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{0,j} : \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$, as required.
    - $u' = u$ and $v' \neq v$. This is analogous to the above case.
    - $u' \neq u$ and $v' \neq v$. It can be shown in a similar way as in previous cases that $R(u', v) \in I_{0,j} \cup J_1$ and $R(v, v') \in I_{0,j-1} \cup J_{0,j-1}$ hold. Assume for the sake of a contradiction that $R(u', v) \in I_{0,j-1} \cup J_{0,j-1}$ holds, then, u', v, and v' all belong to one component in $C_R^{0,j-1}$, which is a contradiction. Consequently, $R(u', v) \in (\Delta^{\mathsf{p}}_{0,j} \setminus \Delta^{\mathsf{m}}_{0,j}) \cup J_1$ holds, and so we have $R(u', v') \in I_{0,j} \cup J_2 \subseteq I_{0,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{0,j} : \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$, as required.
- For the inductive step of the proof, consider an arbitrary index k with $k > 1$ such that $J^{k-1} \subseteq I_{0,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{0,j} : \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$ holds. Let G be an arbitrary fact in J^k. If $G \in J^{k-1}$ holds, then $G \in I_{0,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{0,j} : \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$ follows from the induction assumption. Otherwise, let $R(u, v)$ be the fact considered in the kth iteration of lines 221–228, and we discuss the following possibilities.

– G is added to J^k in line 222. This is analogous to the corresponding case for $k = 1$, so we omit the details.

– G is added to J^k in line 224. This is analogous to the previous case.

– G is added to J^k in line 228. Let $G = R(u', v')$. There are several cases.

  * $u' = u$ and $v' = v$. In this case, $G \in \Delta^{\mathsf{p}}_{0,j} \setminus \Delta^{\mathsf{m}}_{0,j} \subseteq I_{0,j}$ clearly holds.

  * $u' \neq u$ and $v' = v$. Then, u' and u must be put into the same component during a previous call to Algorithm 11 or in a previous iteration of lines 221–228. For the former case, we have $R(u', u) \in \mathsf{Close}(C_R^{0,j-1}) = R[I_{0,j-1} \cup J_{0,j-1}]$. By Definition 15 we have $R(u', u) \in I_{0,j}$, and so $R(u, v) \in \Delta^{\mathsf{p}}_{0,j} \setminus \Delta^{\mathsf{m}}_{0,j}$ implies $R(u', v) \in I_{0,j} \cup J_1 \subseteq I_{0,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$, as required. For the latter case, we have $R(u', u) \in J^{k-1}$. By the induction assumption there exists k' such that $R(u', u) \in I_{0,j} \cup J_{k'}$ holds. Together with $R(u, v) \in \Delta^{\mathsf{p}}_{0,j} \setminus \Delta^{\mathsf{m}}_{0,j}$ we have $R(u', v) \in I_{0,j} \cup J_{k'+1} \subseteq I_{0,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$.

  * $u' = u$ and $v' \neq v$. This is analogous to the above case.

  * $u' \neq u$ and $v' \neq v$. It can be shown in a similar way as in previous cases that there exist k' and k'' such that $R(u', v) \in I_{0,j} \cup J_{k'}$ and $R(v, v') \in I_{0,j} \cup J_{k''}$ hold. Without loss of generality assume that $k'' \geq k'$ holds, then we have $R(u', v') \in I_{0,j} \cup J_{k''+1} \subseteq I_{0,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{0,j} \vdots \Delta^{\mathsf{p}}_{0,j} \cup \Delta^{\mathsf{m}}_{0,j}, \Delta^{\mathsf{n}}_{0,j}]$. □

Claim 64. *For each $1 \leq i \leq m$, if properties (a)–(c) hold for $i - 1$, then property (a) holds for i as well.*

Proof. Consider arbitrary $1 \leq i \leq m$ such that properties (a)–(c) hold for $i - 1$, and we show that property (a) holds for i as well. To this end, we prove by induction on $1 \leq j \leq n_i$ that if call $C_{i,j}$ is of type Del, then the following property holds.

$$R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup J_{i,j})] \subseteq \mathsf{Close}(C_R^{i,j}) \cup Y_R^{i,j} \tag{B.43}$$

For the base case where $j = 1$, by Definition 16 call $C_{i,1}$ is of type Del and call $C_{i-1,n_{i-1}}$ is of type Add. Consequently, $\mathsf{Close}(C_R^{i-1,n_{i-1}}) = R[I_{i-1,n_{i-1}} \cup J_{i-1,n_{i-1}}]$ follows from the induction assumption for $i - 1$; by Definition 16 we have $\mathsf{Close}(C_R^{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$.

We now show that $R[I^{\mathsf{p}}_{i,1} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1} \cup J_{i,1})] \subseteq \mathsf{Close}(C_R^{i,1}) \cup Y_R^{i,1}$ holds. To this end, consider arbitrary $R(u,v) \in R[I^{\mathsf{p}}_{i,1} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1} \cup J_{i,1})]$. Then, $R(u,v) \in R[I^{\mathsf{p}}_{i,1}]$ implies $R(u,v) \in \mathsf{Close}(C_R^{i-1,n_{i-1}})$. Assume for the sake of a contradiction that $R(u,v)$ is not in $\mathsf{Close}(C_R^{i,1}) \cup Y_R^{i,1}$. Then, u and v must be in a component that is removed in line 235 during the execution of $C_{i,1}$. But then, $R(u,v)$ is considered in line 233, and so $R(u,v) \notin Y_R^{i,1}$, $R(u,v) \in R[I^{\mathsf{p}}_{i,1} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1})]$, and line 236 jointly imply $R(u,v) \in J_{i,1}$, which is a contradiction. Consequently, the base case $R[I^{\mathsf{p}}_{i,1} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1} \cup J_{i,1})] \subseteq \mathsf{Close}(C_R^{i,1}) \cup Y_R^{i,1}$ holds, as required.

For the inductive step, consider arbitrary $1 < j \leq n_i$ such that property (B.43) holds for $j-1$ if call $C_{i,j-1}$ is of type Del, and we show that the property holds for j as well if call $C_{i,j}$ is of type Del. Assume that call $C_{i,j}$ is indeed of type Del. Then, Definition 15 ensures that $C_{i,j-1}$ is of type Del as well, and so property (B.43) holds for $j-1$. Now consider arbitrary $R(u,v) \in R[I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup J_{i,j})]$. By Definition 15 we have $R(u,v) \in R[I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}] = R[I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1} \cup J_{i,j-1})]$. Consequently, property (B.43) for $j-1$ implies $R(u,v) \in \mathsf{Close}(C_R^{i,j-1}) \cup Y_R^{i,j-1}$. Assume for the sake of a contradiction that $R(u,v) \notin \mathsf{Close}(C_R^{i,j}) \cup Y_R^{i,j}$ holds. Since $Y_R^{i,j-1} \subseteq Y_R^{i,j}$ holds, we have $R(u,v) \in \mathsf{Close}(C_R^{i,j-1})$, and so in the same way as in the base case we have $R(u,v) \in J_{i,j}$, which is a contradiction. Therefore, the inductive step holds as well.

Next we prove by induction on $1 \leq j \leq n_i$ that the following statement is true: if call $C_{i,j}$ is of type Del, then property (B.44) holds.

$$\mathsf{Close}(C_R^{i,j}) \subseteq I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup J_{i,j}) \tag{B.44}$$

For the base case where $j = 1$, by Definition 16 call $C_{i,1}$ is of type Del and call $C_{i-1,n_{i-1}}$ is of type Add. Consider arbitrary $R(u,v) \in C_R^{i,1}$, then, the way C_R is updated in Algorithm 12 ensures $R(u,v) \in C_R^{i-1,n_{i-1}}$. By the induction assumption for $i-1$ and Definition 16 we have $R(u,v) \in I^{\mathsf{p}}_{i,1} = I^{\mathsf{p}}_{i,1} \setminus \Delta^{\mathsf{m}}_{i,1}$. Assume for the sake of a contradiction that $R(u,v) \in (\Delta^{\mathsf{p}}_{i,1} \setminus \Delta^{\mathsf{m}}_{i,1}) \cup J_{i,1}$ holds, then, u and v must be in a component that is removed in line 235, and so $R(u,v) \notin C_R^{i,1}$ holds, which is a contradiction. Consequently, we have $R(u,v) \notin (\Delta^{\mathsf{p}}_{i,1} \setminus \Delta^{\mathsf{m}}_{i,1}) \cup J_{i,1}$, and so $R(u,v) \in I^{\mathsf{p}}_{i,1} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1} \cup J_{i,1})$ holds.

For the inductive step, consider arbitrary $1 < j \leq n_i$ such that property (B.44) holds for $j-1$ if call $C_{i,j-1}$ is of type Del, and we show that the property holds for j as well if call $C_{i,j}$ is of type Del. Assume that call $C_{i,j}$ is indeed of type Del. Then, Definition 15 ensures that $C_{i,j-1}$ is of type Del as well, and so property (B.44) holds for $j-1$. Now consider arbitrary $R(u,v) \in C_R^{i,j}$, then, the way C_R is updated in Algorithm 12 implies $R(u,v) \in C_R^{i,j-1}$. Consequently, property (B.44) for $j-1$ ensures $R(u,v) \in I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1} \cup J_{i,j-1})$. By Definition 15 we have $R(u,v) \in I^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$. Assume for the sake of a contradiction that $R(u,v) \in (\Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}) \cup J_{i,j}$ holds, then u and v must be in a component that is removed in line 235 during the execution of $C_{i,j}$, which is a contradiction to $R(u,v) \in \mathsf{Close}(C_R^{i,j})$. Consequently, the inductive step holds as well.

We next show by induction on $1 \leq j \leq n_i$ that the following statement is true: if call $C_{i,j}$ is of type Del, then property (B.45) holds.

$$\mathsf{stc}(R)_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \setminus \mathsf{mat}(\Pi, E_i) \subseteq J_{i,j} \tag{B.45}$$

- For the induction base where $j = 1$, consider the corresponding Del call $C_{i,1}$. Let F be an arbitrary fact in $\mathsf{stc}(R)_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \setminus \mathsf{mat}(\Pi, E_i)$. Then, $F \in \mathsf{stc}(R)\Big[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}\Big] \cap I^{\mathsf{p}}_{i,1}$ and $F \notin \Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1} \cup \mathsf{mat}(\Pi, E_i)$ hold by the definition of $\mathsf{stc}_{\downarrow}$. Module $\mathsf{stc}(R)$ involves no negation; moreover, $\Delta^{\mathsf{m}}_{i,1} = \emptyset$ holds by Definition 16. Consequently, we have $F \in \mathsf{stc}(R)\Big[I^{\mathsf{p}}_{i,1} \vdots \Delta^{\mathsf{p}}_{i,1}\Big] \cap I^{\mathsf{p}}_{i,1}$, and we discuss the following possibilities.
  - F is derived by the symmetric rule in the $\mathsf{stc}(R)$ module. Then, there exists fact $R(v,u) \in \Delta^{\mathsf{p}}_{i,1} \subseteq I^{\mathsf{p}}_{i,1}$ such that $F = R(u,v)$ holds. Definition 16 and property (c) for $i-1$ jointly imply $\mathsf{Close}(C_R^{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$. Consequently, we have $R(v,u) \in \mathsf{Close}(C_R^{i-1,n_{i-1}})$—that is, v and u belong to a component in $C_R^{i-1,n_{i-1}}$. But then, $\Delta^{\mathsf{m}}_{i,1} = \emptyset$ and line 231 ensure that this component is examined in lines 232–235, and so $F = R(u,v)$ will be checked in line 233. Now $R(u,v) \notin \mathsf{mat}(\Pi, E_i)$ holds, and so Definition 19 ensures $T(R(u,v)) = \mathsf{f}$. Consequently, $R(u,v)$ is added to J in line 233. Then, $F \in I^{\mathsf{p}}_{i,1} \setminus (\Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1})$ and line 236 jointly imply $F \in J_{i,1}$, as required.

– F is derived by the transitive rule in the $\mathsf{stc}(R)$ module, and there exists v such that $R(u,v) \in \Delta^{\mathsf{p}}_{i,1}$, $R(v,w) \in I^{\mathsf{p}}_{i,1}$, and $F = R(u,w)$ all hold. Then, Definition 16 and property (c) jointly imply that u, v, and w all belong to a component in $C_R^{i-1,n_{i-1}}$; moreover, $R(u,v) \in \Delta^{\mathsf{p}}_{i,1} = \Delta^{\mathsf{p}}_{i,1} \setminus \Delta^{\mathsf{m}}_{i,1}$ implies that this component is examined in lines 232–235. Consequently, $F = R(u,w)$ will be checked in line 233. In a similar way as in the previous case we have $F \in J_{i,1}$, as required.

– F is derived by the transitive rule in the $\mathsf{stc}(R)$ module, and there exists v such that $R(u,v) \in I^{\mathsf{p}}_{i,1}$, $R(v,w) \in \Delta^{\mathsf{p}}_{i,1}$, and $F = R(u,w)$ all hold. This is analogous to the previous case, so we omit the details.

- For the inductive step, consider arbitrary $1 < j \leq n_i$ such that the statement is true for $j-1$. Now if call $C_{i,j}$ is not of type Del, then the statement is clearly true for j as well. Assume that call $C_{i,j}$ is indeed of type Del, and we show that property (B.45) holds. To this end, consider arbitrary $F \in \mathsf{stc}(R)_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \setminus \mathsf{mat}(\Pi, E_i)$. We have $F \in \mathsf{stc}(R)\Big[I^{\mathsf{p}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}\Big] \cap I^{\mathsf{p}}_{i,j}$ and $F \notin \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \cup \mathsf{mat}(\Pi, E_i)$ in a similar way as in the base case. We discuss the following possibilities.

  – F is derived by the symmetric rule in the $\mathsf{stc}(R)$ module. Then, there exists fact $R(v,u) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \subseteq I^{\mathsf{p}}_{i,j}$ such that $F = R(u,v)$ holds. Definition 16 and property (c) for $i-1$ jointly imply $\mathsf{Close}(C_R^{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$. Consequently, we have $R(v,u) \in \mathsf{Close}(C_R^{i-1,n_{i-1}})$—that is, v and u belong to a component in $C_R^{i-1,n_{i-1}}$. Let this component be U. We discuss the following two cases.

    * $R(v,u) \in \Delta^{\mathsf{m}}_{i,j}$ holds. Then, Definition 15 implies $R(v,u) \in J_{i,j-1}$. Consequently, component U is examined in lines 232–235 during the Del call of $C_{i,j-1}$. But then, since u and v both belong to U, $F = R(u,v)$ is checked in line 233 at some point during the execution of $C_{i,j-1}$. Now $R(u,v) \notin \mathsf{mat}(\Pi, E_i)$ and Definition 19 imply $T(R(u,v)) = \mathsf{f}$, and so $F = R(u,v)$ is added to J at some point in line 233. There are two possibilities, which we discuss below.

      · $F \in I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1})$ holds. In this case, due to line 236 we have $F \in J_{i,j-1} = \Delta^{\mathsf{m}}_{i,j}$, which is a contradiction.

· $F \notin I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1})$ holds. But then, by Definition 15 we have $F \notin I^{\mathsf{p}}_{i,j}$, which is a contradiction.

∗ $R(v,u) \in \Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$ holds. Then, line 231 ensures that U is examined in lines 232–235 at some point during the execution of $C_{i,j}$, and so $F = R(u,v)$ will be checked in line 233. Now $R(u,v) \notin \mathsf{mat}(\Pi, E_i)$ holds, and so Definition 19 ensures $T(R(u,v)) = \mathsf{f}$. Thus, $R(u,v)$ is added to J in line 233. Then, $F \in I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j})$ and line 236 jointly imply $F \in J_{i,j}$.

– F is derived by the transitive rule in the $\mathsf{stc}(R)$ module, and there exists v such that $R(u,v) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j} \subseteq I^{\mathsf{p}}_{i,j}$, $R(v,w) \in I^{\mathsf{p}}_{i,j}$, and $F = R(u,w)$ all hold. Then, Definition 15 implies $R(u,v), R(v,w) \in I^{\mathsf{p}}_{i,1}$, and so Definition 16 and property (c) for $i-1$ jointly imply that u, v, and w all belong to a component in $C_R^{i-1,n_{i-1}}$. Let this component be U. We discuss the following two cases.

∗ $R(u,v) \in \Delta^{\mathsf{m}}_{i,j}$ holds. Then, Definition 15 implies $R(u,v) \in J_{i,j-1}$. Consequently, component U is examined in lines 232–235 during the Del call of $C_{i,j-1}$. But then, since u, v, and w all both belong to U, $F = R(u,w)$ is checked in line 233 at some point during the execution of $C_{i,j-1}$. Now $R(u,w) \notin \mathsf{mat}(\Pi, E_i)$ and Definition 19 imply $T(R(u,w)) = \mathsf{f}$, and so $F = R(u,w)$ is added to J at some point in line 233. There are two possibilities, which we discuss below.

· $F \in I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1})$ holds. In this case, due to line 236 we have $F \in J_{i,j-1} = \Delta^{\mathsf{m}}_{i,j}$, which is a contradiction.

· $F \notin I^{\mathsf{p}}_{i,j-1} \setminus (\Delta^{\mathsf{p}}_{i,j-1} \cup \Delta^{\mathsf{m}}_{i,j-1})$ holds. But then, by Definition 15 we have $F \notin I^{\mathsf{p}}_{i,j}$, which is a contradiction.

∗ $R(u,v) \in \Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$ holds. Then, line 231 ensures that U is examined in lines 232–235 at some point during the execution of $C_{i,j}$, and so $F = R(u,w)$ will be checked in line 233. Now $R(u,w) \notin \mathsf{mat}(\Pi, E_i)$ holds, and so Definition 19 ensures $T(R(u,w)) = \mathsf{f}$. Thus, $R(u,v)$ is added to J in line 233. Then, $F \in I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j})$ and line 236 jointly imply $F \in J_{i,j}$.

– F is derived by the transitive rule in the $\mathsf{stc}(R)$ module, and there exists v such that $R(u,v) \in I^{\mathsf{p}}_{i,j}$, $R(v,w) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$, and $F = R(u,w)$ all hold. This is analogous to the previous case, so we omit the details.

Finally, we show that for each $1 \leq j \leq n_i$, the following statement holds: if call $C_{i,j}$ is of type Del, then property B.46 holds.

$$J_{i,j} \subseteq \mathsf{stc}(R)^{\infty}_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \tag{B.46}$$

To this end, consider arbitrary j with $1 \leq j \leq n_i$ such that call $C_{i,j}$ is of type Del. Let L be the total number of iterations for the loop of lines 231–235; moreover, let J^0 be the value of J right before line 231, and for each $1 \leq l \leq L$, let J^l be the value of J right after the lth iteration of lines 231–235; finally, let Δ_i be defined with respect to $\mathsf{stc}(R)^{\infty}_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$ as in Definition 10. We show by induction on l that $J^l \cap (I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j})) \subseteq \mathsf{stc}(R)^{\infty}_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$ holds. For simplicity we denote $\mathsf{stc}(R)^{\infty}_{\downarrow}[I^{\mathsf{p}}_{i,j}, I^{\mathsf{n}}_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$ by J_s.

- For the induction base where $l = 0$, we have $J^0 = \emptyset$, and so the required condition clearly holds.
- For the inductive step, consider arbitrary $l > 0$ such that $J^{l-1} \cap (I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j})) \subseteq J_s$ holds. Let F be an arbitrary fact in $J^l \cap (I^{\mathsf{p}}_{i,j} \setminus (\Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}))$. If $F \in J^{l-1}$ holds, then by the induction assumption we clearly have $F \in J_s$. Otherwise, F is added to J^l in line 233 during the lth iteration of lines 231–235. Let $F = R(u', v')$. We discuss the following two possibilities.
  - $j = 1$. In this case, due to line 231 there exists component $U \in C_R^{i-1,n_{i-1}}$ such that $u, v, u', v' \in U$ and $R(u,v) \in \Delta^{\mathsf{p}}_{i,1} \setminus \Delta^{\mathsf{m}}_{i,1}$ hold. Definition 16 and property (c) for $i-1$ jointly imply $\mathsf{Close}(C_R^{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$. Consequently we have $R(u',u), R(u',v), R(v,v') \in I^{\mathsf{p}}_{i,1}$. Then, the definition of $\mathsf{stc}(R)^{\infty}_{\downarrow}$ and $R(u,v) \in \Delta^{\mathsf{p}}_{i,1} \setminus \Delta^{\mathsf{m}}_{i,1}$ jointly imply $R(u',v) \in \Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1} \cup \Delta_1$, which in turn implies $R(u',v') \in \Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1} \cup \Delta_1 \cup \Delta_2$. Now $F = R(u',v') \notin \Delta^{\mathsf{p}}_{i,1} \cup \Delta^{\mathsf{m}}_{i,1}$ implies $F \in J_s$, as required.

– $1 < j \leq n_i$. In this case, due to line 231 there exists component $U \in C_R^{i,j-1}$ such that $u, v, u', v' \in U$ and $R(u, v) \in \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$ hold. Property (B.44) for $j-1$ and Definition 15 jointly imply $R(u', u), R(u', v), R(v, v') \in I_{i,j}^{\mathsf{p}}$. Then, the definition of $\mathsf{stc}(R)_{\downarrow}^{\infty}$ and $R(u, v) \in \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$ jointly imply $R(u', v) \in \Delta_{i,j}^{\mathsf{p}} \cup \Delta_{i,j}^{\mathsf{m}} \cup \Delta_1$, which in turn implies $R(u', v') \in \Delta_{i,j}^{\mathsf{p}} \cup \Delta_{i,j}^{\mathsf{m}} \cup \Delta_1 \cup \Delta_2$. Now $F = R(u', v') \notin \Delta_{i,j}^{\mathsf{p}} \cup \Delta_{i,j}^{\mathsf{m}}$ implies $F \in J_s$, as required.

The choice of F is arbitrary, and so $J^l \cap (I_{i,j}^{\mathsf{p}} \setminus (\Delta_{i,j}^{\mathsf{p}} \cup \Delta_{i,j}^{\mathsf{m}})) \subseteq J_s$ holds, as required. □

Claim 65. *For each $1 \leq i \leq m$, if properties (a)–(c) hold for $i-1$, then property (b) holds for i as well.*

Proof. Consider arbitrary i with $1 \leq i \leq m$ such that properties (a)–(c) hold for $i-1$, we show that property (b) holds for i as well. In particular, we prove that the following statement is true for each $1 \leq j \leq n_i$: if call $C_{i,j}$ is of type Red, then (B.47) holds.

$$\mathsf{Close}(C_R^{i,j}) = R[I_{i,j}^{\mathsf{p}} \cup J_{i,j}] \tag{B.47}$$

Note that Definition 16 ensures that there exists at most one index k with $1 \leq k \leq n_i$ such that $C_{i,k}$ is of type Red. Therefore, if no such k exists, then the statement mentioned above clearly holds for each $1 \leq j \leq n_i$. Otherwise, the statement holds for each j with $1 \leq j \leq n_i$ and $j \neq k$, and we show that $\mathsf{Close}(C_R^{i,k}) = R[I_{i,k}^{\mathsf{p}} \cup J_{i,k}]$ holds.

For the $\subseteq$ direction, consider arbitrary $R(u, v) \in \mathsf{Close}(C_R^{i,k})$. There are two cases.

- $R(u, v) \in \mathsf{Close}(C_R^{i,k-1})$. By Definition 15 call $C_{i,k-1}$ is of type Del, and so Claim 64 ensures $R(u, v) \in R[I_{i,k-1}^{\mathsf{p}} \setminus (\Delta_{i,k-1}^{\mathsf{p}} \cup \Delta_{i,k-1}^{\mathsf{m}} \cup J_{i,k-1})]$. Then, Definition 15 ensures $R(u, v) \in R[I_{i,k}^{\mathsf{p}}]$.
- $R(u, v) \notin \mathsf{Close}(C_R^{i,k-1})$. Then, u and v are guaranteed to be in the same component in $\mathsf{Close}(C_R^{i,k})$ due to line 222, line 224, or line 226. In each of these cases, $R(u, v)$ is added to J, and so line 237 implies $R(u, v) \in R[I_{i,k}^{\mathsf{p}} \cup J_{i,k}]$, as required.

For the $\supseteq$ direction, We discuss the following two cases.

- If $R(u,v) \in R[I^{\mathsf{p}}_{i,k}]$, then Claim 64 implies $R(u,v) \in \mathsf{Close}(C^{i,k-1}_R) \cup Y^{i,k-1}_R$. There are two possibilities. If $R(u,v) \in \mathsf{Close}(C^{i,k-1}_R)$, then $R(u,v) \in \mathsf{Close}(C^{i,k}_R)$ clearly holds. If $R(u,v) \in Y^{i,k-1}_R$, then line 237 ensures that $R(u,v)$ is considered in line 220. Then, lines 221–228 ensure that u and v end up in the same component in $C^{i,k}_R$, and so $R(u,v) \in \mathsf{Close}(C^{i,k}_R)$ holds.
- If $R(u,v) \in J_{i,k}$, then $R(u,v)$ must be added to J in line 222, 224, or 228. In each of these cases, u and v must end up in the same component in $C^{i,k}_R$, and so $R(u,v) \in \mathsf{Close}(C^{i,k}_R)$ holds. □

Next we show that $\mathsf{stc}(R)\left[I^{\mathsf{p}}_{i,k}, I^{\mathsf{n}}_{i,k}\right] \cap \Delta_{i,k} \subseteq J_{i,k}$ holds. To this end, consider arbitrary $R(u,w) \in \mathsf{stc}(R)\left[I^{\mathsf{p}}_{i,k}, I^{\mathsf{n}}_{i,k}\right] \cap \Delta_{i,k}$. There are two possibilities.

- $R(u,w)$ is derived by the symmetric rule. Then, $R(w,u) \in I^{\mathsf{p}}_{i,k}$ holds. Call $C_{i,k}$ is of type Red, and so by Definition 15 we have $I^{\mathsf{p}}_{i,k} = I^{\mathsf{p}}_{i,k-1} \setminus (\Delta^{\mathsf{p}}_{i,k-1} \cup \Delta^{\mathsf{m}}_{i,k-1})$ and $J_{i,k-1} = \emptyset$. Consequently, we have $R(w,u) \in I^{\mathsf{p}}_{i,k-1} \setminus (\Delta^{\mathsf{p}}_{i,k-1} \cup \Delta^{\mathsf{m}}_{i,k-1} \cup J_{i,k-1})$. But then, Claim 64 implies $R(w,u) \in \mathsf{Close}(C^{i,k-1}_R) \cup Y^{i,k-1}_R$. Now $R(u,w) \in \Delta_{i,k}$ and Definition 15 jointly imply $R(u,w) \notin I^{\mathsf{p}}_{i,k} = I^{\mathsf{p}}_{i,k-1} \setminus (\Delta^{\mathsf{p}}_{i,k-1} \cup \Delta^{\mathsf{m}}_{i,k-1} \cup J_{i,k-1})$, which together with Claim 64 implies $R(u,w) \notin \mathsf{Close}(C^{i,k-1}_R)$. Consequently, $R(w,u) \notin \mathsf{Close}(C^{i,k-1}_R)$, and so we have $R(w,u) \in Y^{i,k-1}_R$. This fact will thus be considered in line 237, and so w and u will belong to one component in $C^{i,k}_R$. Consequently, $R(u,w) \in \mathsf{Close}(C^{i,k}_R) = R[I^{\mathsf{p}}_{i,k} \cup J_{i,k}]$ holds, and so $R(u,w) \notin I^{\mathsf{p}}_{i,k}$ implies $R(u,w) \in J_{i,k}$, as required.
- $R(u,w)$ is derived by the transitive rule. Then, there exists v such that $R(u,v) \in I^{\mathsf{p}}_{i,k}$ and $R(v,w) \in I^{\mathsf{p}}_{i,k}$ hold. Similarly to the previous case, $R(u,v) \in \mathsf{Close}(C^{i,k-1}_R) \cup Y^{i,k-1}_R$ and $R(v,w) \in \mathsf{Close}(C^{i,k-1}_R) \cup Y^{i,k-1}_R$ hold. Moreover, Definitions 15 and 16, and the induction assumption for $i-1$ jointly imply $R(u,v), R(v,w) \in R[I^{\mathsf{p}}_{i,1}] = \mathsf{Close}(C^{i-1,n_{i-1}}_R)$. The way C_R is updated ensures that each node could only belong to one component in C_R, and so u, v, and w all belong to one component in $C^{i-1,n_{i-1}}_R$. Assume for the sake of a contradiction that this component does not get removed from C_R in line 235 before call $C_{i,k}$. Then we have $R(u,w) \in \mathsf{Close}(C^{i,k-1}_R) \subseteq I^{\mathsf{p}}_{i,k}$, which is a contradiction to $R(u,w) \in \Delta_{i,k}$. Consequently, we have $R(u,v) \in Y^{i,k-1}_R$ and $R(v,w) \in Y^{i,k-1}_R$, and so both $R(u,v)$ and $R(v,w)$ will be considered in line 237 during the execution of $C_{i,k}$.

Therefore, $R(u,w) \in \mathsf{Close}(C_R^{i,k})$ holds, which together with $R(u,w) \notin I_{i,k}^{\mathsf{p}}$ implies $R(u,w) \in J_{i,k}$.

The choice of $R(u,w)$ is arbitrary, so $\mathsf{stc}(R)\left[I_{i,k}^{\mathsf{p}}, I_{i,k}^{\mathsf{n}}\right] \cap \Delta_{i,k} \subseteq J_{i,k}$ holds.

Finally we show that $J_{i,k} \subseteq \mathsf{mat}(\Pi, E_i) \cap \Delta_{i,k}$. To this end, consider the execution of function CloseEdges(Y_R) in line 237 during the call of $C_{i,k}$. For each l with $1 \leq l \leq |Y_R^{i,k-1}|$, let J^l be the value of J after the lth iteration of lines 221–228. We show by induction on l that $J^l \subseteq I_{i,1}^{\mathsf{p}}$ and $J^l \subseteq \mathsf{mat}(\Pi, E_i)$ hold.

- For the induction base where $l = 1$, let $R(u,v) \in Y_R^{i,k-1}$ be the fact considered in this iteration, and consider arbitrary $G \in J^1$. We discuss the following possibilities.
  - G is added to J^1 in line 222, then clearly $G = R(u,u)$ holds. The way Y_R is updated ensures $Y_R^{i,k-1} \subseteq I_{i,1}^{\mathsf{p}}$, and so Definitions 15 and 16, and the induction assumption for $i-1$ jointly imply $Y_R^{i,k-1} \subseteq R[I_{i,1}^{\mathsf{p}}] = R[I_{i-1,n_{n-1}}] = \mathsf{Close}(C_R^{i-1,n_{i-1}})$. Therefore, $R(u,v) \in Y_R^{i,k-1}$ implies that u and v belong to one component in $C_R^{i-1,n_{i-1}}$. Consequently, $R(u,u) \in \mathsf{Close}(C_R^{i-1,n_{i-1}}) = R[I_{i,1}^{\mathsf{p}}]$ holds, as required. Moreover, Definition 19 and $R(u,v) \in Y_R^{i,k-1}$ jointly imply $R(u,v) \in \mathsf{mat}(\Pi, E_i)$. Note that $R(u,u)$ can be derived from $R(u,v)$ in two steps using the symmetric rule first and then the transitive rule, and so $R(u,u) \in \mathsf{mat}(\Pi, E_i)$ holds.
  - G is added to J^1 in line 224. This is analogous to the previous case.
  - G is added to J^1 in line 228. As argued previously, u and v belong to one component in $C_R^{i-1,n_{i-1}}$; moreover, $R(u,v) \in Y_R^{i,k-1}$ and line 235 ensure that this component gets removed before the call of $C_{i,k}$. Consequently, U and V contain only u and v, respectively, in line 225. Therefore, we have $G = R(u,v)$ or $G = R(v,u)$; either way, $G \in I_{i,1}^{\mathsf{p}}$ and $G \in \mathsf{mat}(\Pi, E_i)$ hold.
- For the inductive step, consider arbitrary $1 < l \leq |Y_R^{i,k-1}|$ such that $J^{l-1} \subseteq I_{i,1}^{\mathsf{p}}$ and $J^{l-1} \subseteq \mathsf{mat}(\Pi, E_i)$ hold. Let $R(u,v) \in Y_R^{i,k-1}$ be the fact considered in the lth iteration of lines 221–228, and let G be an arbitrary fact in J^l. If $G \in J^{l-1}$, then the induction assumption for $l-1$ implies $G \in I_{i,1}^{\mathsf{p}}$ and $G \in \mathsf{mat}(\Pi, E_i)$. Otherwise, we discuss the following possibilities.

- G is added to J^l in line 222 or line 224. This is analogous to the corresponding case in the induction base for $l = 1$, so we omit the details.
- G is added to J^l in line 228. Let $G = R(u', v')$. There are two cases.
  - u' and u belong to one component, and v' and v belong to a distinct component. Then, u' and u are put into the same component either in the current iteration of lines 221–228, or in a previous iteration. Either way, we have $R(u', u) \in R[I^{\mathsf{p}}_{i,1}] = \mathsf{Close}(C_R^{i-1,n_{i-1}})$ and $R(u', u) \in \mathsf{mat}(\Pi, E_i)$. Similarly, we have $R(v, v') \in \mathsf{Close}(C_R^{i-1,n_{i-1}})$ and $R(v, v') \in \mathsf{mat}(\Pi, E_i)$. Together with $R(u, v) \in Y_R^{i,k-1} \subseteq \mathsf{Close}(C_R^{i-1,n_{i-1}})$ and $R(u, v) \in \mathsf{mat}(\Pi, E_i)$ we have $R(u', v') \in \mathsf{Close}(C_R^{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$ and $R(u', v') \in \mathsf{mat}(\Pi, E_i)$.
  - u' and v belong to one component, and v' and u belong to a distinct component. Then, similarly to the previous case, $R(v', u') \in \mathsf{Close}(C_R^{i-1,n_{i-1}})$ and $R(v', u') \in \mathsf{mat}(\Pi, E_i)$ hold. Thus, $R(u', v') \in \mathsf{Close}(C_R^{i-1,n_{i-1}}) = R[I^{\mathsf{p}}_{i,1}]$ and $R(u', v') \in \mathsf{mat}(\Pi, E_i)$ hold as well.

$J^l \subseteq I^{\mathsf{p}}_{i,1}$ and $J^l \subseteq \mathsf{mat}(\Pi, E_i)$ hold for $l = |Y_R^{i,k-1}|$, and so due to line 237 we have $J_{i,k} \subseteq I^{\mathsf{p}}_{i,1} \setminus I^{\mathsf{p}}_{i,k}$ and $J_{i,k} \subseteq \mathsf{mat}(\Pi, E_i)$. Consequently, Definition 15 ensures $J_{i,k} \subseteq \mathsf{mat}(\Pi, E_i) \cap \Delta_{i,k}$, as required.

Claim 66. *For each $1 \leq i \leq m$, if properties (a)–(c) hold for $i - 1$, then property (c) holds for i as well.*

Proof. Consider arbitrary i with $1 \leq i \leq m$ such that properties (a)–(c) hold for $i - 1$, we show that property (c) holds for i as well. In particular, we prove that the following statement is true for each $1 \leq j \leq n_i$: if call $C_{i,j}$ is of type Add, then properties (B.48)–(B.49) hold.

$$\mathsf{Close}(C_R^{i,j}) = R[I_{i,j} \cup J_{i,j}] \tag{B.48}$$

$$\mathsf{stc}(R)^{\uparrow}[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \subseteq J_{i,j} \subseteq \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}] \tag{B.49}$$

Note that Definition 16 ensures that there exists at most one index k with $1 \leq k \leq n_i$ such that $C_{i,k}$ is of type Red. If no such k exists, then by Definition 15 each call $C_{i,j}$ with

$1 \leq j \leq n_i$ is of type Del, and so the statement trivially holds for each j. If such a k does exist, then the statement is clearly true for each j with $1 \leq j \leq k$ since call $C_{i,j}$ is not of type Add; in contrast, each call $C_{i,j}$ with $k < j \leq n_i$ is of type Add by Definitions 15 and 16, and so we first prove (B.48) by induction on j with $k < j \leq n_i$.

- For the induction base where $j = k + 1$, note that call $C_{i,k}$ is of type Red, and so by Claim 65 we have $\mathsf{Close}(C_R^{i,k}) = R[I_{i,k}^{\mathsf{p}} \cup J_{i,k}]$. We would like to show that $\mathsf{Close}(C_R^{i,k+1}) = R[I_{i,k+1} \cup J_{i,k+1}]$ holds.
  - For the $\subseteq$ direction, consider an arbitrary fact $R(u, v)$ in $\mathsf{Close}(C_R^{i,k+1})$. If $R(u, v) \in \mathsf{Close}(C_R^{i,k})$, then Claim 65 ensures $R(u, v) \in R[I_{i,k}^{\mathsf{p}} \cup J_{i,k}]$; Definition 15 ensures $I_{i,k+1} = I_{i,k}^{\mathsf{p}} \cup J_{i,k} \cup \Delta_{i,k+1}^{\mathsf{p}}$, and so $R(u, v) \in R[I_{i,k+1} \cup J_{i,k+1}]$ holds. Otherwise, we have $R(u, v) \in \mathsf{Close}(C_R^{i,k+1}) \setminus \mathsf{Close}(C_R^{i,k})$. Now if $u = v$, then u belongs to a component in $C_R^{i,k+1}$ but does not belong to any component in $C_R^{i,k}$, and so u is introduced into $C_R^{i,k+1}$ in either line 222 or line 224. Either way, $R(u, v) \in R[I_{i,k+1} \cup J_{i,k+1}]$ holds. If $u \neq v$, then u and v are put into the same component in line 226, and so $R(u, v) \in R[I_{i,k+1} \cup J_{i,k+1}]$ holds as well. The choice of $R(u, v)$ is arbitrary, so the $\subseteq$ direction holds, as required.
  - For the $\supseteq$ direction, consider arbitrary $R(u, v) \in R[I_{i,k+1} \cup J_{i,k+1}]$. Then, Definition 15 implies $R(u, v) \in R[I_{i,k}^{\mathsf{p}} \cup J_{i,k} \cup \Delta_{i,k+1}^{\mathsf{p}} \cup J_{i,k+1}]$. There are three possibilities, which we discuss below.
    - If $R(u, v) \in R[I_{i,k}^{\mathsf{p}} \cup J_{i,k}]$, then Claim 65 and the way C_R is updated in Algorithm 11 jointly imply $R(u, v) \in \mathsf{Close}(C_R^{i,k}) \subseteq \mathsf{Close}(C_R^{i,k+1})$.
    - If $R(u, v) \in R[\Delta_{i,k+1}^{\mathsf{p}} \setminus J_{i,k}] = R[\Delta_{i,k+1}^{\mathsf{p}} \setminus \Delta_{i,k+1}^{\mathsf{m}}]$, it will be considered in line 220 during the execution of $C_{i,k+1}$. If $u = v$, lines 221 and 222 ensure $R(u, v) \in \mathsf{Close}(C_R^{i,k+1})$; if $u \neq v$, lines 225 and 226 ensure that u and v are in the same component in $C_R^{i,k+1}$, and so $R(u, v) \in \mathsf{Close}(C_R^{i,k+1})$ holds in this case as well.
    - If $R(u, v) \in R[J_{i,k+1}]$, then it must be added to $J_{i,k+1}$ via line 222, 224, or 228. In each of these cases, u and v will be guaranteed to be in the same component in $C_R^{i,k+1}$, and so $R(u, v) \in \mathsf{Close}(C_R^{i,k+1})$ holds.

The choice of $R(u, v)$ is arbitrary, so the $\supseteq$ direction holds as well.

- For the inductive step, consider arbitrary j with $k + 1 < j \leq n_i$ such that property (B.48) holds for $j - 1$, and we show that the property holds for j as well. The proof is analogous to the above, so we omit the details.

We next prove that the left-hand side $\subseteq$ of property (B.49) holds for each $k < j \leq n_i$. To this end, we first consider $j = k + 1$ and the corresponding Add call $C_{i,j}$. Let $R(u, w)$ be an arbitrary fact in $\mathsf{stc}(R)^{\uparrow}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$. By the definition of $\mathsf{stc}(R)^{\uparrow}$ we have $R(u, w) \in \mathsf{stc}(R)\Big[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}\Big] \setminus I_{i,j}$. Note that $\mathsf{stc}(R)$ involves no negation, so we have $R(u, w) \in \mathsf{stc}(R)\Big[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}\Big] \setminus I_{i,j}$. We discuss the following possibilities.

- $R(w, u) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$ holds. There are two cases.
  - If $R(w, u) \in \Delta^{\mathsf{m}}_{i,j}$ holds, then by Definition 15 we have $R(w, u) \in J_{i,j-1}$. By Claim 65 we have $\mathsf{Close}(C_R^{i,j-1}) = R[I^{\mathsf{p}}_{i,j-1} \cup J_{0,j-1}]$, and so w and u belong to one component in $C_R^{i,j-1}$. Consequently, $R(u, w) \in \mathsf{Close}(C_R^{i,j-1}) = R[I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}]$ holds. Then, Definition 15 ensures $R(u, w) \in I_{i,j}$, which is a contradiction.
  - If $R(w, u) \in \Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$ holds, then $R(w, u)$ is considered at some point in line 220 during the execution of $C_{i,j}$. Consequently, Lines 221–228 ensure that w and u belong to one component in $C_R^{i,j}$, and so $R(u, w) \in \mathsf{Close}(C_R^{i,j}) = R[I_{i,j} \cup J_{i,j}]$ holds. Then, $R(u, w) \notin I_{i,j}$ implies $R(u, w) \in J_{i,j}$.
- There exists v such that $R(u, v) \in \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}$ and $R(v, w) \in I_{i,j}$ hold. Then, Definition 15 implies $R(v, w) \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1} \cup \Delta^{\mathsf{p}}_{i,j}$. There are several cases.
  - If $R(u, v) \in \Delta^{\mathsf{m}}_{i,j}$ and $R(v, w) \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}$ hold, then Definition 15 implies $R(u, v) \in J_{i,j-1}$. Now $\mathsf{Close}(C_R^{i,j-1}) = R[I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}]$ holds, and so $R(u, v), R(v, w) \in \mathsf{Close}(C_R^{i,j-1})$ holds. Consequently, u, v, and w all belong to one component in $C_R^{i,j-1}$, and so $R(u, w) \in \mathsf{Close}(C_R^{i,j-1}) = R[I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}]$. Then, by Definition 15 we have $R(u, w) \in I_{i,j}$, which is a contradiction.
  - If $R(u, v) \in \Delta^{\mathsf{m}}_{i,j} = J_{i,j-1}$ and $R(v, w) \in \Delta^{\mathsf{p}}_{i,j} \setminus J_{i,j-1} = \Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$ hold, then $\mathsf{Close}(C_R^{i,j-1}) = R[I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}]$ implies $R(u, v) \in \mathsf{Close}(C_R^{i,j-1})$. Consequently, u and v belong to one component in $C_R^{i,j-1}$. The way C_R is updated in

Algorithm 11 ensures that u and v still belong to one component in $C_R^{i,j}$. Now $R(v,w) \in \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$ ensures that $R(v,w)$ is considered at some point in line 220 during the execution of $C_{i,j}$, and so lines 221–228 ensure that v and w end up in one component in $C_R^{i,j}$. Since we have already shown that u and v belong to one component in $C_R^{i,j}$, we have $R(u,w) \in \mathsf{Close}(C_R^{i,j}) = R[I_{i,j} \cup J_{i,j}]$. But then, $R(u,w) \notin I_{i,j}$ implies $R(u,w) \in J_{i,j}$, as required.

- If $R(u,v) \in \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$ and $R(v,w) \in I_{i,j-1}^{\mathsf{p}} \cup J_{i,j-1}$ both hold, then clearly $\mathsf{Close}(C_R^{i,j-1}) = R[I_{i,j-1}^{\mathsf{p}} \cup J_{i,j-1}]$ implies $R(v,w) \in \mathsf{Close}(C_R^{i,j-1})$. In the same way as in the previous case, v and w belong to one component in $C_R^{i,j}$; moreover, $R(u,v) \in \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$ ensures that u belongs to the same component. Consequently, we have $R(u,w) \in \mathsf{Close}(C_R^{i,j}) = R[I_{i,j} \cup J_{i,j}]$, and so $R(u,w) \notin I_{i,j}$ implies $R(u,w) \in J_{i,j}$, as required.
- If $R(u,v) \in \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$ and $R(v,w) \in \Delta_{i,j}^{\mathsf{p}} \setminus J_{i,j-1} = \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$ hold, then both $R(u,v)$ and $R(v,w)$ are considered at some point in line 220 during the execution of $C_{i,j}$. Consequently, u, v, and w all belong to one component in $C_R^{i,j}$, and so in a way similar to the previous cases we have $R(u,w) \in J_{i,j}$.

The proof for j with $k+1 < j \leq n_i$ is analogous to the above, so we omit the details.

Finally, we show that the right-hand side $\subseteq$ of property (B.49) holds for each j with $k < j \leq n_i$. To this end, consider first $j = k+1$ and the corresponding Add call $C_{i,j}$. For $1 \leq l \leq |\Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}|$, let J^l be the value of J right after the lth iteration of lines 221–228 during the execution of $C_{i,j}$. Moreover, let J_i be defined with respect to $\mathsf{stc}(R)_\infty^\uparrow[I_{i,j} : \Delta_{i,j}^{\mathsf{p}} \cup \Delta_{i,j}^{\mathsf{m}}, \Delta_{i,j}^{\mathsf{n}}]$ as in Definition 9. We show by induction on l that $J^l \subseteq I_{i,j} \cup \mathsf{stc}(R)_\infty^\uparrow[I_{i,j} : \Delta_{i,j}^{\mathsf{p}} \cup \Delta_{i,j}^{\mathsf{m}}, \Delta_{i,j}^{\mathsf{n}}]$ holds. Then, line 217 ensures $J_{i,j} \subseteq \mathsf{stc}(R)_\infty^\uparrow[I_{i,j} : \Delta_{i,j}^{\mathsf{p}} \cup \Delta_{i,j}^{\mathsf{m}}, \Delta_{i,j}^{\mathsf{n}}]$.

- For the induction base where $l = 1$, let $R(u,v) \in \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$ be the fact considered in this iteration of lines 221–228, and consider arbitrary $G \in J^1$. We discuss the following possibilities.
  - G is added to J^1 in line 222, then clearly $G = R(u,u)$ holds. Then, by the definition of J_i we have $R(v,u) \in I_{i,j} \cup J_1$. Consequently, $R(u,v) \in \Delta_{i,j}^{\mathsf{p}} \setminus \Delta_{i,j}^{\mathsf{m}}$

implies that $R(u,u)$ can be derived by the transitive rule. This clearly implies $G = R(u,u) \in I_{i,j} \cup J_2 \subseteq I_{i,j} \cup \mathsf{stc}(R)^{\uparrow}_{\omega}[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$, as required.

- G is added to J^1 in line 222. This is analogous to the previous case.
- G is added to J^1 in line 228. Let $G = R(u',v')$. There are several cases.
  - $u' = u$ and $v' = v$. In this case, $G \in \Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j} \subseteq I_{i,j}$ clearly holds.
  - $u' \neq u$ and $v' = v$. Then, u' and u must already belong to one component in $C_R^{i,j-1}$. Consequently, we have $R(u',u) \in \mathsf{Close}(C_R^{i,j-1}) = R[I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}]$. By Definition 15 we have $R(u',u) \in I_{i,j}$, and so $R(u,v) \in \Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$ implies $R(u',v) \in I_{i,j} \cup J_1 \subseteq I_{i,j} \cup \mathsf{stc}(R)^{\uparrow}_{\omega}[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$, as required.
  - $u' = u$ and $v' \neq v$. This is analogous to the above case, so we omit the details.
  - $u' \neq u$ and $v' \neq v$. It can be shown in a similar way as in previous cases that $R(u',v) \in I_{i,j} \cup J_1$ and $R(v,v') \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}$ hold. Assume for the sake of a contradiction that $R(u',v) \in I^{\mathsf{p}}_{i,j-1} \cup J_{i,j-1}$ holds, then, u', v, and v' all belong to one component in $C_R^{i,j-1}$, which is a contradiction. Consequently, $R(u',v) \in (\Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}) \cup J_1$, and so we have $R(u',v') \in I_{i,j} \cup J_2 \subseteq I_{i,j} \cup \mathsf{stc}(R)^{\uparrow}_{\omega}[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$, as required.

- For the inductive step, consider arbitrary $l > 1$ such that $J^{l-1} \subseteq I_{i,j} \cup \mathsf{stc}(R)^{\uparrow}_{\omega}[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$ holds. Let G be an arbitrary fact in J^l. If $G \in J^{l-1}$ holds, then by the induction assumption we have $G \in I_{i,j} \cup \mathsf{stc}(R)^{\uparrow}_{\omega}[I_{i,j} \colon \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$. Otherwise, let $R(u,v)$ be the fact considered in the kth iteration of lines 221–228, and we discuss the following possibilities.
  - G is added to J^l in line 222. This is analogous to the corresponding case for $l = 1$, so we omit the details.
  - G is added to J^l in line 224. This is analogous to the previous case.
  - G is added to J^l in line 228. Let $G = R(u',v')$. There are several cases.
    - $u' = u$ and $v' = v$. In this case, $G \in \Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j} \subseteq I_{i,j}$ clearly holds.

- $u' \neq u$ and $v' = v$. Then, u' and u either belong to one component in $C_R^{i,j-1}$, or are put into the same component in a previous iteration of lines 221–228. For the former case, we have $R(u', u) \in \mathsf{Close}(C_R^{i,j-1}) \subseteq R[I_{i,j}]$, and so $R(u', v) \in I_{i,j} \cup J_1 \subseteq I_{i,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$, as required. For the latter case, we have $R(u', u) \in J^{l-1}$. By the induction assumption there exists k' such that $R(u', u) \in I_{i,j} \cup J_{k'}$ holds. Together with $R(u, v) \in \Delta^{\mathsf{p}}_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}$ we have $R(u', v) \in I_{i,j} \cup J_{k'+1} \subseteq I_{i,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$, as required.
- $u' = u$ and $v' \neq v$. This is analogous to the above case.
- $u' \neq u$ and $v' \neq v$. It can be shown in a similar way as in previous cases that there exist k' and k'' such that $R(u', v) \in I_{i,j} \cup J_{k'}$ and $R(v, v') \in I_{i,j} \cup J_{k''}$ hold. Without loss of generality assume that $k'' \geq k'$ holds, then we have $R(u', v') \in I_{i,j} \cup J_{k''+1} \subseteq I_{i,j} \cup \mathsf{stc}(R)^{\uparrow}_{\infty}[I_{i,j} \vdots \Delta^{\mathsf{p}}_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}, \Delta^{\mathsf{n}}_{i,j}]$, as required.

The proof for j with $k + 1 < j \leq n_i$ is analogous to the above. □

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Peter Alvaro, Tyson Condie, Neil Conway, Khaled Elmeleegy, Joseph M. Hellerstein, and Russell Sears. "Boom analytics: exploring data-centric, declarative programming for the cloud". In: *European Conference on Computer Systems, Proceedings of the 5th European conference on Computer systems*. ACM, 2010, pp. 223–236.

[3] Peter Alvaro, Neil Conway, Joseph M. Hellerstein, and William R. Marczak. "Consistency Analysis in Bloom: a CALM and Collected Approach". In: *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Online Proceedings*. www.cidrdb.org, 2011, pp. 249–260.

[4] Molham Aref, Balder ten Cate, Todd J. Green, Benny Kimelfeld, Dan Olteanu, Emir Pasalic, Todd L. Veldhuizen, and Geoffrey Washburn. "Design and Implementation of the LogicBlox System". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1371–1382.

[5] Faiz Arni, KayLiang Ong, Shalom Tsur, Haixun Wang, and Carlo Zaniolo. "The Deductive Database System LDL++". In: *TPLP* 3.1 (2003), pp. 61–94.

[6] François Bancilhon. "Naive Evaluation of Recursively Defined Relations". In: *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*. Springer, 1985, pp. 165–178.

[7] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. "The Vadalog System: Datalog-based Reasoning for Knowledge Graphs". In: *PVLDB* 11.9 (2018), pp. 975–987.

[8] Barry Bishop, Atanas Kiryakov, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. "OWLIM: A family of scalable semantic repositories". In: *Semantic Web* 2.1 (2011), pp. 33–42.

[9] Barry Bishop, Atanas Kiryakov, Damyan Ognyanov, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. "FactForge: A fast track to the Web of data". In: *Semantic Web* 2.2 (2011), pp. 157–166.

[10] José A. Blakeley, Per-Åke Larson, and Frank Wm. Tompa. "Efficiently Updating Materialized Views". In: *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*. ACM Press, 1986, pp. 61–71.

[11] Martin Bravenboer and Yannis Smaragdakis. "Strictly declarative specification of sophisticated points-to analyses". In: *Proceedings of the 24th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2009*. ACM, 2009, pp. 243–262.

[12] Stefano Ceri, Georg Gottlob, and Letizia Tanca. “What you Always Wanted to Know About Datalog (And Never Dared to Ask)”. In: *IEEE Trans. Knowl. Data Eng.* 1.1 (1989), pp. 146–166.

[13] Stefano Ceri and Jennifer Widom. “Deriving Production Rules for Incremental View Maintenance”. In: *17th International Conference on Very Large Data Bases*. Morgan Kaufmann, 1991, pp. 577–589.

[14] The UniProt Consortium. “UniProt: a hub for protein information”. In: *Nucleic Acids Research* 43.D1 (2015), pp. D204–D212.

[15] David Croft, Antonio Fabregat Mundo, Robin Haw, Marija Milacic, Joel Weiser, Guanming Wu, Michael Caudy, Phani Garapati, Marc Gillespie, Maulik R. Kamdar, Bijay Jassal, Steven Jupe, Lisa Matthews, Bruce May, Stanislav Palatnik, Karen Rothfels, Veronica Shamovsky, Heeyeon Song, Mark Williams, Ewan Birney, Henning Hermjakob, Lincoln Stein, and Peter D’Eustachio. “The Reactome pathway knowledgebase”. In: *Nucleic Acids Research* 42.Database-Issue (2014), pp. 472–477.

[16] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. “Complexity and expressive power of logic programming”. In: *ACM Comput. Surv.* 33.3 (2001), pp. 374–425.

[17] *Datomic*. `https://www.datomic.com`. Accessed: 2019-09-25.

[18] Camil Demetrescu and Giuseppe F. Italiano. “Fully Dynamic Transitive Closure: Breaking Through the O(n^2) Barrier”. In: *41st Annual Symposium on Foundations of Computer Science, FOCS 2000*. IEEE Computer Society, 2000, pp. 381–389.

[19] John DeTreville. “Binder, a Logic-Based Security Language”. In: *2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2002, pp. 105–113.

[20] Hasanat M. Dewan, David Ohsie, Salvatore J. Stolfo, Ouri Wolfson, and Sushil Da Silva. “Incremental Database Rule Processing In PARADISER”. In: *J. Intell. Inf. Syst.* 1.2 (1992), pp. 177–209.

[21] Guozhu Dong, Jianwen Su, and Rodney W. Topor. “Nonrecursive Incremental Evaluation of Datalog Queries”. In: *Ann. Math. Artif. Intell.* 14.2-4 (1995), pp. 187–223.

[22] Paul Doran, Valentina A. M. Tamma, and Luigi Iannone. “Ontology module extraction for ontology reuse: an ontology engineering perspective”. In: *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007*. ACM, 2007, pp. 61–70.

[23] Jason Eisner and Nathaniel Wesley Filardo. “Dyna: Extending Datalog for Modern AI”. In: *Datalog Reloaded - First International Workshop, Datalog 2010. Revised Selected Papers*. Springer, 2010, pp. 181–220.

[24] Wolfgang Faber, Gerald Pfeifer, and Nicola Leone. “Semantics and complexity of recursive aggregates in answer set programming”. In: *Artif. Intell.* 175.1 (2011), pp. 278–298.

[25] Sumit Ganguly, Abraham Silberschatz, and Shalom Tsur. “A Framework for the Parallel Processing of Datalog Queries”. In: *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*. ACM Press, 1990, pp. 143–152.

[26] Anna Gaulton, Louisa J. Bellis, A. Patrícia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington. “ChEMBL: a large-scale bioactivity database for drug discovery”. In: *Nucleic Acids Research* 40.Database-Issue (2012), pp. 1100–1107.

[27] Georg Gottlob, Christoph Koch, Robert Baumgartner, Marcus Herzog, and Sergio Flesca. "The Lixto Data Extraction Project - Back and Forth between Theory and Practice". In: *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM, 2004, pp. 1–12.

[28] *GraphDB*. `graphdb.ontotext.com`. Accessed: 2019-09-25.

[29] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. "A Logical Framework for Modularity of Ontologies". In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*. 2007, pp. 298–303.

[30] Todd J. Green, Shan Shan Huang, Boon Thau Loo, and Wenchao Zhou. "Datalog and Recursive Query Processing". In: *Foundations and Trends in Databases* 5.2 (2013), pp. 105–195.

[31] Timothy Griffin and Leonid Libkin. "Incremental Maintenance of Views with Duplicates". In: *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*. ACM Press, 1995, pp. 328–339.

[32] Timothy Griffin, Leonid Libkin, and Howard Trickey. "An Improved Algorithm for the Incremental Recomputation of Active Relational Expressions". In: *IEEE Trans. Knowl. Data Eng.* 9.3 (1997), pp. 508–511.

[33] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. "LUBM: A benchmark for OWL knowledge base systems". In: *J. Web Semant.* 3.2-3 (2005), pp. 158–182.

[34] Ashish Gupta, Dinesh Katiyar, and Inderpal Singh Mumick. "Counting solutions to the View Maintenance Problem". In: *Proceedings of the Workshop on Deductive Databases held in conjunction with the Joint International Conference and Symposium on Logic Programming*. Department of Computer Science, University of Melbourne, 1992, pp. 185–194.

[35] 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*. ACM Press, 1993, pp. 157–166.

[36] Eric N. Hanson. "A Performance Analysis of View Materialization Strategies". In: *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference*. ACM Press, 1987, pp. 440–453.

[37] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission. 2004.

[38] Pan Hu, Jacopo Urbani, Boris Motik, and Ian Horrocks. "Datalog Reasoning over Compressed RDF Knowledge Bases". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 2019, pp. 2065–2068.

[39] Toshihide Ibaraki and Naoki Katoh. "On-Line Computation of Transitive Closures of Graphs". In: *Inf. Process. Lett.* 16.2 (1983), pp. 95–97.

[40] Trevor Jim. "SD3: A Trust Management System with Certified Evaluation". In: *2001 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2001, pp. 106–115.

[41] Mark Kaminski, Bernardo Cuenca Grau, Egor V. Kostylev, Boris Motik, and Ian Horrocks. "Foundations of Declarative Data Analysis Using Limit Datalog Programs". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*. AAAI Press. 2017, pp. 1123–1130.

[42] Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. "Datalog rewritability of Disjunctive Datalog programs and non-Horn ontologies". In: *Artif. Intell.* 236 (2016), pp. 90–118.

[43] Valerie King. "Fully Dynamic Algorithms for Maintaining All-Pairs Shortest Paths and Transitive Closure in Digraphs". In: *40th Annual Symposium on Foundations of Computer Science, FOCS '99*. IEEE Computer Society, 1999, pp. 81–91.

[44] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. "DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia". In: *Semantic Web* 6.2 (2015), pp. 167–195.

[45] Boon Thau Loo, Tyson Condie, Minos N. Garofalakis, David E. Gay, Joseph M. Hellerstein, Petros Maniatis, Raghu Ramakrishnan, Timothy Roscoe, and Ion Stoica. "Declarative networking: language, execution and optimization". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 2006, pp. 97–108.

[46] Boon Thau Loo, Tyson Condie, Joseph M. Hellerstein, Petros Maniatis, Timothy Roscoe, and Ion Stoica. "Implementing declarative overlays". In: *Proceedings of the 20th ACM Symposium on Operating Systems Principles 2005, SOSP 2005*. ACM, 2005, pp. 75–90.

[47] Boon Thau Loo, Joseph M. Hellerstein, Ion Stoica, and Raghu Ramakrishnan. "Declarative routing: extensible routing with declarative queries". In: *Proceedings of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM, 2005, pp. 289–300.

[48] Li Ma, Yang Yang, Zhaoming Qiu, Guo Tong Xie, Yue Pan, and Shengping Liu. "Towards a Complete OWL Ontology Benchmark". In: *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Proceedings*. Springer, 2006, pp. 125–139.

[49] Alberto O. Mendelzon and Peter T. Wood. "Finding Regular Simple Paths in Graph Databases". In: *SIAM J. Comput.* 24.6 (1995), pp. 1235–1258.

[50] Alistair Miles and Sean Bechhofer. "SKOS simple knowledge organization system reference". In: *W3C recommendation* 18 (2009), W3C.

[51] Boris Motik, Yavor Nenov, Robert Edgar Felix Piro, and Ian Horrocks. "Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 1560–1568.

[52] Boris Motik, Yavor Nenov, Robert Piro, and Ian Horrocks. "Maintenance of datalog materialisations revisited". In: *Artif. Intell.* 269 (2019), pp. 76–136.

[53] Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, and Dan Olteanu. "Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, 2014, pp. 129–137.

[54] Boris Motik, Peter Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, and Michael Smith. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. W3C. 2009.

[55] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. "RDFox: A Highly-Scalable RDF Store". In: *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Proceedings, Part II*. Springer, 2015, pp. 3–20.

[56] Jean-Marie Nicolas and Kioumars Yazdanian. "An Outline of BDGEN: A Deductive DBMS". In: *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress*. North-Holland/IFIP, 1983, pp. 711–717.

[57] Esko Nuutila. "Efficient transitive closure computation in large digraphs". PhD thesis. Helsinki University of Technology, 1995.

[58] Robert Piro, Yavor Nenov, Boris Motik, Ian Horrocks, Peter Hendler, Scott Kimberly, and Michael Rossman. "Semantic Technologies for Data Analysis in Health Care". In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Proceedings, Part II*. Lecture Notes in Computer Science. 2016, pp. 400–417.

[59] Johannes A. La Poutré and Jan van Leeuwen. "Maintenance of Transitive Closures and Transitive Reductions of Graphs". In: *Graph-Theoretic Concepts in Computer Science, International Workshop, WG '87, Proceedings*. Springer, 1987, pp. 106–120.

[60] Xiaolei Qian and Gio Wiederhold. "Incremental Recomputation of Active Relational Expressions". In: *IEEE Trans. Knowl. Data Eng.* 3.3 (1991), pp. 337–341.

[61] Raghu Ramakrishnan, Divesh Srivastava, and S. Sudarshan. "Rule Ordering in Bottom-Up Fixpoint Evaluation of Logic Programs". In: *IEEE Trans. Knowl. Data Eng.* 6.4 (1994), pp. 501–517.

[62] Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan, and Praveen Seshadri. "The CORAL Deductive System". In: *VLDB J.* 3.2 (1994), pp. 161–210.

[63] Yuan Ren and Jeff Z. Pan. "Optimising ontology stream reasoning with truth maintenance system". In: *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011*. ACM, 2011, pp. 831–836.

[64] Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. "Declarative Information Extraction Using Datalog with Embedded Extraction Predicates". In: *Proceedings of the 33rd International Conference on Very Large Data Bases*. ACM, 2007, pp. 1033–1044.

[65] Martin Staudt and Matthias Jarke. "Incremental Maintenance of Externally Materialized Views". In: *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases*. Morgan Kaufmann, 1996, pp. 75–86.

[66] Heiner Stuckenschmidt and Michel C. A. Klein. "Structure-Based Partitioning of Large Concept Hierarchies". In: *The Semantic Web - ISWC 2004: Third International Semantic Web Conference, Proceedings*. Springer, 2004, pp. 289–303.

[67] Julien Subercaze, Christophe Gravier, Jules Chevalier, and Frédérique Laforest. "Inferray: fast in-memory RDF inference". In: *PVLDB* 9.6 (2016), pp. 468–479.

[68] Jacopo Urbani, Ceriel J. H. Jacobs, and Markus Krötzsch. "Column-Oriented Datalog Materialization for Large Knowledge Graphs". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 258–264.

[69] Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri E. Bal. "WebPIE: A Web-scale Parallel Inference Engine using MapReduce". In: *J. Web Semant.* 10 (2012), pp. 59–75.

[70] Jacopo Urbani, Alessandro Margara, Ceriel J. H. Jacobs, Frank van Harmelen, and Henri E. Bal. "DynamiTE: Parallel Materialization of Dynamic RDF Data". In: *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Proceedings, Part I*. Springer, 2013, pp. 657–672.

[71] Dimitra Vista. "Optimizing Incremental View Maintenance Expressions in Relational Databases". PhD thesis. University of Toronto, Ont., Canada, 1997.

[72] Dimitra Vista. "Integration of Incremental View Maintenance into Query Optimizers". In: *Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology, Proceedings*. Springer, 1998, pp. 374–388.

[73] John Whaley, Dzintars Avots, Michael Carbin, and Monica S. Lam. "Using Datalog with Binary Decision Diagrams for Program Analysis". In: *Programming Languages and Systems, Third Asian Symposium, APLAS 2005, Proceedings*. Springer, 2005, pp. 97–118.

[74] Ouri Wolfson, Hasanat M. Dewan, Salvatore J. Stolfo, and Yechiam Yemini. "Incremental Evaluation of Rules and its Relationship to Parallelism". In: *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*. ACM Press, 1991, pp. 78–87.

[75] Zhe Wu, George Eadon, Souripriya Das, Eugene Inseok Chong, Vladimir Kolovski, Melliyal Annamalai, and Jagannathan Srinivasan. "Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle". In: *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008*. IEEE Computer Society, 2008, pp. 1239–1248.

[76] Mihalis Yannakakis. "Graph-Theoretic Methods in Database Theory". In: *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM Press, 1990, pp. 230–242.

[77] Weining Zhang, Ke Wang, and Siu-Cheung Chau. "Data Partition and Parallel Evaluation of Datalog Programs". In: *IEEE Trans. Knowl. Data Eng.* 7.1 (1995), pp. 163–176.

[78] Yujiao Zhou, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, and Jay Banerjee. "Making the most of your triple store: query answering in OWL 2 using an RL reasoner". In: *22nd International World Wide Web Conference, WWW '13*. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 1569–1580.