Modular materialisation of Datalog programs

Pan Hu [a,b], Boris Motik [b,*], Ian Horrocks [b]

[a] *School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China*
[b] *Department of Computer Science, University of Oxford, Oxford, United Kingdom*

A R T I C L E   I N F O

A B S T R A C T

Answering queries over large datasets extended with Datalog rules plays a key role in numerous data management applications, and it has been implemented in several highly optimised Datalog systems in both academic and commercial contexts. Many systems implement reasoning via *materialisation*, which involves precomputing all consequences of the rules and the dataset in a preprocessing step. Some systems also use *incremental reasoning* algorithms, which can update the materialisation efficiently when the input dataset changes. Such techniques allow queries to be processed without any reference to the rules, so they are often used in applications where the performance of query answering is critical.

Existing materialisation and incremental reasoning techniques enumerate all possible ways to apply rules to the data in order to derive all relevant consequences. This, however, can be inefficient because derivations of rules commonly used in practice are redundant; for example, rules axiomatising a binary predicate as symmetric and transitive can have a cubic number of applications, yet they can derive at most a quadratic number of facts. Such redundancy can be a significant source of overhead in practice and can prevent Datalog systems from successfully processing large datasets. To address this issue, in this paper we present a novel framework for *modular materialisation and incremental reasoning*. Our key idea is that, for certain combinations of rules commonly used in practice, all consequences can be derived using specialised procedures that do not necessarily enumerate all possible rule applications. Thus, our framework supports materialisation and incremental reasoning via a collection of *modules*. Each module is responsible for deriving consequences of a subset of the program, by using either standard rule application or proprietary algorithms. We prove that such an approach is complete as long as each module satisfies certain properties. Our formalisation of a module is very general, and in fact it allows modules to keep arbitrary auxiliary information.

We also show how to realise custom procedures for four types of modules: transitivity, symmetry–transitivity, chain rules, and sequencing elements of a total order. Finally, we demonstrate empirically that using our custom procedures can speed up materialisation and incremental reasoning by several orders of magnitude on several well-known benchmarks. Thus, our technique has the potential to significantly improve the scalability of Datalog reasoners.

---

* Corresponding author.
E-mail addresses: pan.hu@sjtu.edu.cn (P. Hu), boris.motik@cs.ox.ac.uk (B. Motik), ian.horrocks@cs.ox.ac.uk (I. Horrocks).

1. Introduction

Datalog [1] is a prominent formalism that can be used to describe a domain of interest using a set of 'if-then' rules that can be applied to a set of explicitly given facts to produce fresh facts describing the domain. Rules are iteratively applied up to a fixpoint, which allows Datalog to express common second-order properties such as reachability and transitive closure. This expressive power has made Datalog very popular in academia and practice. In the database community, Datalog is often seen as a quintessential recursive query language [8,18,34,74]. In the artificial intelligence community, it is often used as a knowledge representation formalism [26]. In the Semantic Web community, Datalog is frequently used as mechanism for answering queries over ontologies; for example, answering queries over OWL 2 RL [59] ontologies, possibly extended with SWRL rules [42], can be realised using Datalog. The computational properties and expressive power of Datalog are well understood [19]. Moreover, many useful extensions of basic Datalog have been proposed, such as stratified [71], well-founded [84], or stable [30] negation-as-failure; disjunction [31,27]; and aggregation [49,28,75].

A key computational task in Datalog applications is answering queries over facts derived from a set of Datalog rules and a set of explicitly given facts; this is also known as *Datalog reasoning* in knowledge representation and reasoning—a prominent branch of AI. While not the only one, a common way to solve this problem is to precompute and store all derived facts so that queries can be directly evaluated over all (i.e., both explicitly given and derived) facts without further consulting the rules. We refer to both this process and its output as *materialisation*. Materialisation-based query answering has been implemented in many systems, including but not limited to WebPIE [82], VLog [16], Oracle's RDF Store [89], OWLIM [12], and RDFox [66]. The *naïve* materialisation strategy involves repeatedly apply the Datalog rules until no new fact can be derived. The *seminaïve evaluation strategy* [1] optimises this process by requiring each inference to involve at least one fact derived in the previous iteration; this ensures that each combination of a rule and a set of facts matching the 'if' part are considered exactly once.

Recomputing the materialisation 'from scratch' whenever the input facts change is often infeasible in practice. Thus, a key challenge in materialisation-based systems is the *incremental update* problem, which is to update the materialisation efficiently (i.e., without repeating most of the work). Fact addition can be handled effectively using the seminaïve algorithm, but fact deletion is much more involved since one has to check whether deleted facts have derivations that persist after the update. The *Counting* [39] and *Delete/Rederive* (DRed) [39,78] algorithms are two widely known and used solutions to the incremental update problem, with the former one being applicable to nonrecursive rules only. The basic ideas behind Counting and DRed have been further extended, and several highly optimised algorithms such as the *Backward/Forward* (B/F) algorithm [64], the *Forward–Backward–Forward* (FBF) algorithm [65], and DRed^c and B/F^c [43] have been presented recently. All of these approaches use a variant of the seminaïve evaluation to apply rules to sets of facts.

These techniques have successfully been used in many practical applications. For example, Datalog has been used to succinctly express and evaluate network protocols [56,54,55], describe complex quality measures from healthcare records [70], express data analytics tasks in distributed computing [5,6], specify information extraction tasks over unstructured and semi-structured data [32,76], capture program analysis tasks [15,87], and encode security policies [22,47]. This growing interest in Datalog has motivated the development of many highly optimised academic and commercial systems, such as VLog, Oracle's RDF store, RDFox, LogicBlox [7], Vadalog [11], GraphDB [33], and Datomic [20].

This growing body of experience has uncovered a new source of inefficiency in Datalog-based applications: Datalog rules commonly used in practice are redundant in the sense that one fact is often derived by several different rule applications. Thus, although seminaïve evaluation ensures that each rule application is considered just once, the same fact can nevertheless be derived afresh for each distinct rule application; for example, when applied to facts $R(a, b_1), \ldots, R(a, b_n)$, rule $R(x, y) \rightarrow S(x)$ derives $S(a)$ using n distinct rule applications. To see how this can be detrimental to the scalability of Datalog systems, consider Datalog rules that axiomatise a binary predicate as symmetric and transitive. In Section 4 we show that, given facts that encode a graph consisting of n connected vertices, these rules have $O(n^3)$ possible applications. Note, however, that the symmetric–transitive closure of a binary relation over n elements contains only n^2 facts; thus, each fact is repeatedly derived by n rule applications. Since the seminaïve algorithm enumerates all possible rule applications, its running time is also cubic. In contrast, the symmetric–transitive closure can be computed in quadratic time: we first compute the strongly connected components of the graph in $O(n)$ steps, and then for each component we introduce an edge between all component vertices. In other words, using a customised procedure we can solve the problem without enumerating all possible rule applications. In Sections 4, 6.4, and 6.5 we show further examples of commonly used rules and facts on which seminaïve evaluation suffers from similar redundancy.

In this paper we explore ways of improving the performance of materialisation and incremental updates by handling specific Datalog rules using specialised algorithms (instead of using generic seminaïve evaluation). Similar attempts have already been made in the literature. For example, several techniques have been developed for maintaining transitive closure of a binary relation [46,51,50,21], and they have been used to compute closure of transitive and symmetric properties in RDFS-Plus [80]. An extensive body of research has been devoted to the investigation of database properties (e.g., connectivity, transitive closure, or domain parity) by evaluating first-order queries over the database and the updates [69,24,25,53]. While these approaches have proved to be very effective, all of them handle only a specific set of rules, and it is unclear how to integrate them into a general framework that supports arbitrary rules. For example, if a set of Datalog rules axiomatises a predicate as transitive and the same predicate is used in other rules, then the facts derived by transitivity may trigger other rules and vice versa; thus, a custom procedure for transitivity needs to exchange information with other custom

procedures and/or seminaïve evaluation. As we show in this paper, organising this communication in an efficient way is far from straightforward. Moreover, many of these approaches cannot handle deletion of input facts, which is a key problem in incremental updates.

To address these issues, in Section 5 we present a framework for materialisation and incremental updates that integrates specialised algorithms with the seminaïve evaluation in a truly modular way. In our approach, a Datalog program is partitioned into disjoint subsets called *modules*. For each module, one must provide three functions that compute certain consequences of the module's rules. There are no restrictions on how these functions are realised, and, if no specialised algorithms are available, one can use seminaïve evaluation for this purpose. We then present algorithms for computing the materialisation and incremental updates that, instead of using seminaïve evaluation directly, call the three functions for the modules of the program. Effectively, the key aspect of our framework is organising the exchange of information between different modules. We prove that our algorithms are sound, complete, and terminating as long as all module functions satisfy certain formal properties. Moreover, our framework retains the good properties of standard materialisation algorithms in the sense that, when all module functions are implemented using seminaïve evaluation, then each rule application is considered exactly once.

A key challenge in our work is to capture the formal requirements on module functions in a sufficiently general way. A key problem is to allow modules to maintain auxiliary information that can speed up the computation of updates. For example, in the case of the symmetric–transitive closure mentioned earlier, it is beneficial to maintain the list of strongly connected components. To obtain a general framework, we should not make any assumptions about the structure of information maintained by each module, which in turn makes formalising the requirements on module functions difficult. We address this issue by imposing conditions not just on each individual function call, but also on the sequence of calls over the lifetime of a module.

Next, in Section 6.1 we introduce a concept that allows modules to exchange information about the truth of facts after the update; most modules we discuss in this paper use this concept to avoid overdeleting facts that can be easily shown to hold after the update. Then, in Sections 6.2 to 6.5 we show how to realise the three functions for four modules commonly used in practice: transitive closure, symmetric–transitive closure, chain rules, and sequencing elements of a total order. In each case, we prove that our functions satisfy the properties required by our framework.

Finally, in Section 7 we validate our approach empirically. After implementing all four modules in a prototype Datalog reasoner, we have evaluated the performance of modular materialisation and incremental reasoning in several real-life and synthetic datasets. Our results show that custom algorithms can often improve the performance of Datalog systems by several orders of magnitude. We identified several cases where both materialisation and incremental updates times drop from several hours to only a few seconds when our optimisations are enabled. Thus, our techniques seem to make an important contribution to pushing the boundary of scalability of practical Datalog systems.

This is a substantial extension of our earlier work published at the AAAI 2019 conference [44]. New material includes a completely new formalisation of the modular reasoning framework that allows us to decouple the framework correctness argument from module function correctness proofs, algorithms for two additional modules along with their correctness proofs, evaluation results on three additional benchmarks, and an empirical comparison of our approach with the B/F^c algorithm.

2. Related work

In this section, we present an overview of existing approaches to materialisation and incremental update, and we also survey related approaches that handle specific types of rules using custom algorithms. The incremental update problem has been studied extensively in relational databases as the problem of *view maintenance*, where views over base relations are defined using queries and materialised for query evaluation; thus, views need to be updated when base relations change.

2.1. Datalog materialisation

Bancilhon [10] originally introduced the seminaïve evaluation strategy for materialising recursively defined relations. Ramakrishnan et al. [73] studied the impact of rule orderings on the performance of seminaïve evaluation, and proposed variants of seminaïve evaluation capable of handling user-defined rule orderings (but without clarifying how to automatically identify 'good' orderings). Ramakrishnan et al. [74] have implemented the seminaïve evaluation strategy in the CORAL Datalog system. Ganguly et al. [29] and Zhang et al. [90] considered parallelising seminaïve evaluation of Datalog programs by statically assigning rule instantiations to different processors. In contrast, Motik et al. [61] proposed a parallel variant of the seminaïve evaluation that dynamically partitions rule instantiations and is thus less susceptible to workload skew. Urbani et al. [83] and Hu et al. [45] presented variants of seminaïve evaluation that exploit column-oriented data formats.

2.2. Maintenance of nonrecursive views

Blakeley et al. [14] presented an approach for maintaining views defined by Select-Project-Join (SPJ) queries based on derivation counting. The idea is to associate with each tuple in the materialised view a counter that tracks the number of the

tuple's derivations. The counter gets incremented when a new derivation becomes available, and it gets decremented when an existing derivation no longer holds. Thus, a tuple can be safely deleted when its counter drops to zero. Hanson [40] refined this approach further, and compared the cost of counting with the cost of view rematerialisation. Nicolas and Yazdanian [67] and Gupta et al. [39] presented similar approaches for maintaining views defined by nonrecursive Datalog programs.

Ceri and Widom [17] presented an approach that incrementally maintains duplicate-free views by using production rules automatically generated from view definitions. Qian and Wiederhold [72] presented an approach that handles views defined by relational algebra. This approach does not use any additional bookkeeping; rather, it computes the necessary updates by evaluating maintenance queries derived from view definitions. Griffin et al. [36] showed that this method does not always compute minimal maintenance queries, and they presented an improved variant that preserves minimality. Griffin and Libkin [35] also extended this idea to relational algebra with bag semantics, which also allows for view definitions with aggregate functions. A key advantage of algebraic approaches over algorithmic solutions is that view updates are computed using queries, and the evaluation of the latter can be optimised using standard techniques. Vista [85,86] implemented a query optimiser that extended standard query optimisation techniques to support maintenance queries.

2.3. Maintenance of recursive views

Gupta et al. [38] extended the counting approaches by Nicolas and Yazdanian [67] and Gupta et al. [39] to recursive Datalog rules, but this approach is incorrect when a fact recursively derives itself. Wolfson et al. [88] proposed a counting-based algorithm that overcomes the above problem and is capable of handling recursion correctly for arbitrary datasets, and Dewan et al. [23] reformulated this algorithm and proved its correctness. The main idea is to maintain, for each fact, several derivation counters, where the i-th counter reflects the number of derivations of the fact in the i-th round of rule application during materialisation. A major drawback of this algorithm is that it is based on the naïve evaluation strategy for Datalog, which is inherently inefficient. Motik et al. [65] described an optimised variant of this algorithm based on the more efficient seminaïve evaluation strategy.

Gupta et al. [39] proposed the Delete/Rederive (DRed) algorithm that can update views defined by general (i.e., recursive) Datalog programs without any bookkeeping. To handle fact deletion (which is usually more difficult than fact addition), the algorithm first deletes all consequences of the deleted facts, and then rederives facts that still hold (i.e., that have alternative derivations) after the update. Staudt and Jarke [78] presented a closely related algorithm, but formalised it declaratively: instead of presenting a procedure, updates are computed by evaluating maintenance Datalog rules. Motik et al. [65] identified several inefficiencies of these algorithms and presented a significantly optimised variant of DRed.

Recently, Motik et al. [64] proposed the Backward/Forward (B/F) incremental update algorithm, which eagerly identifies alternative derivations of facts during deletion so that deletion becomes exact. Motik et al. [65] further presented the Forward/Backward/Forward (FBF) algorithm, which generalises both DRed and B/F.

DRed, B/F, and FBF all use 'backward' rule evaluation to find alternative derivations of a fact, which involves matching a fact with the rule head, instantiating the matched variables in the body, and evaluating the resulting body as a query. This query, however, can be difficult to evaluate in some cases [43], which can prevent efficient incremental updates. To overcome this, Hu et al. [43] recently proposed the DRed^c and B/F^c algorithms, which combine counting with DRed and B/F. In particular, the B/F^c algorithm associates with each fact a counter of nonrecursive derivations, which is used to prevent overdeletion of facts that clearly hold; this, in turn, avoids unnecessary 'backward' rule evaluation. Moreover, the DRed^c algorithm additionally keeps track of recursive derivations for each fact; then, after overdeletion, precisely the facts with nonzero recursive derivations need to be rederived, and this can be done without any 'backward' rule evaluation.

2.4. Using custom solutions for specific types of rules

Dong et al. [24] presented an algorithm that can incrementally update query answers of a regular chain Datalog program by constructing and evaluating a nonrecursive Datalog program. Subercaze et al. [80] applied Nuutila's algorithm for transitive closure [68] to handle transitive and symmetric properties in RDFS-Plus. While these approaches provide custom algorithms for specific rule sets, it is unclear whether and how they can be combined with arbitrary rules. In contrast, the main objective of our work is to devise a general framework that can integrate custom solutions with standard seminaïve evaluation; thus, custom solutions can be used to improve the performance of reasoning without sacrificing the expressivity and generality of Datalog.

To optimise reasoning with Datalog programs containing the equality predicate, Motik et al. [63] devised a rewriting based approach that involves choosing a representative resource for each clique of elements that are equal and replacing all resources with their representatives; this approach has also been extended to support incremental materialisation maintenance [62].

In their GLog engine (which is an extension of VLog), Tsamoura et al. [81] used an earlier version of our work [62] to optimise materialisation (but not incremental update) of transitive and symmetric–transitive rules. Their experimental results agree with ours and show that using custom algorithms can indeed speed up materialisation by orders of magnitude.

3. Preliminaries

We now recapitulate the definitions of the syntax and the semantics of Datalog. We consider the variant of Datalog with *stratified negation*, which we further extend slightly by allowing negation over existentially quantified conjunctions of atoms. This extension allows us to capture rules for sequencing elements of a total order that we consider in Section 6.5.

A Datalog *signature* consists of infinite and disjoint sets of *constants* and *predicates*, where each predicate is associated with a nonnegative integer *arity*. In addition, the set of *variables* is infinite and disjoint from the signature. Unless otherwise stated, in this paper we denote constants using (possibly indexed) lowercase letters from the beginning of the alphabet $(a, b, c, \ldots)$, variables using lowercase letters from the end of the alphabet $(x, y, z, \ldots)$, and predicates using uppercase letters usually taken from the middle of the alphabet $(P, R, S, \ldots)$. A *term* is a constant or a variable, and an *atom* is an expression of the form $P(t_1, \ldots, t_k)$, where P is predicate of arity k, and $t_1, \ldots, t_k$ are terms. A *fact* is a variable-free atom, and a *dataset* is a finite set of facts. Occasionally, we call a fact/atom a P-*fact* or P-*atom* to stress that the predicate of the fact/atom is P. A *negative literal* is an expression the form

$$\text{not}\,\exists x_1, \ldots, x_k.[C_1 \wedge \cdots \wedge C_\ell],$$

where $x_1, \ldots, x_k$ are variables, and each C_i with $1 \leq i \leq \ell$ is an atom; when $k = 0$, we simply write $\text{not}\,[C_1 \wedge \cdots \wedge C_\ell]$. A variable occurring in some C_i but not in $x_1, \ldots, x_k$ is said to be *free* in the literal. Analogously, each variable occurring in an atom is said to be *free* in that atom. A rule is an expression of the form

$$B_1 \wedge \cdots \wedge B_m \wedge L_{m+1} \wedge \cdots \wedge L_n \rightarrow H,$$

where $0 \leq m \leq n$, each B_i with $1 \leq i \leq m$ is an atom, each L_i with $m + 1 \leq i \leq n$ is a negative literal, and H is an atom. Each rule must be *safe*—that is, each variable free in H or some L_i must also be free in some B_j. For r a rule, $\text{h}(r) = H$ is the *head*, $\text{b}^+(r) = \{B_1, \ldots, B_m\}$ is the set of *positive body atoms*, $\text{b}^-(r) = \{L_{m+1}, \ldots, L_n\}$ is the set of *negative body literals*, and $\text{b}(r) = \text{b}^+(r) \cup \text{b}^-(r)$. A *(Datalog) program* is a finite set of rules.

Variants of Datalog typically considered in the literature do not allow negative literals to contain existentially quantified conjunctions—that is, negative literals are usually of the form $\text{not}\,C$ where C is an atom. Example 1 illustrates why we consider this extension and shows that doing so does not affect the properties of Datalog in any substantial way.

Example 1. In Section 6.5 we discuss a common practical use case that involves evaluating rules of the following form:

$$P(x) \wedge P(y) \wedge R(x, y) \wedge \text{not}\,\exists z.[P(z) \wedge R(x, z) \wedge R(z, y)] \rightarrow S(x, y). \tag{1}$$

To reduce rule (1) to 'standard' Datalog, we can replace the conjunction with an atom containing a fresh binary predicate Q that is defined to contain the same tuples as the conjunction. In other words, we can replace rule (1) with the following rules:

$$P(x) \wedge P(y) \wedge R(x, y) \wedge \text{not}\,Q(x, y) \rightarrow S(x, y) \tag{2}$$

$$P(z) \wedge R(x, z) \wedge R(z, y) \rightarrow Q(x, y). \tag{3}$$

It is straightforward to see that, on any set of explicitly given facts, rule (1) produces the same facts as rules (2)–(3) for all predicates apart from Q. The main drawback of such a rewriting is practical: the arity of the replacement predicate must be equal to the number of free variables of the conjunction and materialising predicates can incur a considerable overhead. In the setting we consider in Section 6.5, we can evaluate rule (1) easily, whereas materialising predicate Q would be prohibitive. Thus, our extension of negation to existentially quantified conjunctions allows us to discuss an interesting use case from both the theoretical and practical perspective.

Some definitions of Datalog, particularly those in the database literature, distinguish *extensional* and *intensional* predicates: the former are allowed to occur in the explicitly given facts and rule bodies, and the latter are allowed to occur in rule bodies and heads. In this paper we use the knowledge representation perspective on Datalog where this distinction is typically not made.

We next recapitulate conditions on the structure of Datalog programs that allow negation to be interpreted in an intuitive way. A *stratification* λ of a program Π is a surjection from the predicates in the signature to the set $\{1, 2, \ldots, S\}$ for some integer S such that, for each rule $r \in \Pi$ with predicate P occurring in $\text{h}(r)$ and each predicate R occurring in $\text{b}^+(r)$ (resp. $\text{b}^-(r)$), we have $\lambda(P) \geq \lambda(R)$ (resp. $\lambda(P) > \lambda(R)$). Program Π is *stratifiable* if a stratification λ of Π exists; note that more than one such λ may exist. In this paper, we shall consider only stratifiable programs. A rule r with predicate P occurring in $\text{h}(r)$ is *recursive* w.r.t. λ if $\lambda(P) = \lambda(R)$ holds for some predicate R occurring in $\text{b}^+(r)$; otherwise, r is *nonrecursive* w.r.t. λ. For s in the range of λ, we define Π^s as the subset of the program Π that contains each rule $r \in \Pi$ such that $\lambda(P) = s$ where P is the predicate of $\text{h}(r)$. We call program Π^s a *stratum*, and we call s a *stratum index*. We define Π_r^s and Π_{nr}^s as the recursive and the nonrecursive subsets, respectively, of Π^s. For each s with $1 \leq s \leq S$, let

$$O^s = \{P(c_1, \ldots, c_n) \mid P \text{ is a predicate of arity } n \text{ with } \lambda(P) = s, \text{ and } c_i \text{ are constants in the signature}\}.$$

Thus, O^s contain all facts that can be constructed from constants in the signature and predicates with stratum index s; thus, each fact in O^s can be defined only using the rules in Π^s. Moreover, to simplify the presentation of certain technical results, we extend this definition to $s = 0$—that is, we define $O^0 = \emptyset$. Finally, we define $O^{\leq s} = \bigcup_{0 \leq s' \leq s} O^{s'}$ and $O^{<s} = \bigcup_{0 \leq s' < s} O^{s'}$.

A *substitution* σ is a function from variables to terms that is not an identity on finitely many variables. For α a term, an atom, a negative literal, a rule, or a set thereof, $\alpha\sigma$ is the result of replacing each free occurrence of a variable x in α with $\sigma(x)$. For r a rule and σ a substitution mapping all free variables of r to constants, rule $r\sigma$ is an *instance* of r. A *unifier* of atoms α and β is a substitution σ such that $\alpha\sigma = \beta\sigma$; such σ is a *most general unifier* if, for each unifier ρ of α and β, there exists a substitution η such that $\rho(x) = \eta(\sigma(x))$ for each variable x. The most general unifier of α and β is unique up to variable renaming.

Now let I be a dataset. For F a fact, we write $I \models F$ if $F \in I$ holds. For $L = \text{not } \exists x_1, \ldots, x_k.(C_1 \wedge \cdots \wedge C_\ell)$ a negative literal with no free variables, we write $I \models L$ if no substitution σ of $x_1, \ldots, x_k$ exists such that $I \models C_i\sigma$ holds for each $1 \leq i \leq \ell$. For S a set consisting of facts and negative literals with no free variables, we write $I \models S$ if $I \models \alpha$ for each $\alpha \in S$. For Π a program, the set $\Pi[I]$ of all facts obtained by applying the rules of Π to I is defined as

$$\Pi[I] = \bigcup_{r \in \Pi} \{h(r\sigma) \mid r\sigma \text{ is an instance of } r \text{ such that } I \models b(r\sigma)\}. \tag{4}$$

Let E be a dataset of *explicitly given* facts and let λ be a stratification of Π with maximum stratum index S. Then, let $I_\infty^0 = \emptyset$; for each s with $1 \leq s \leq S$, let

$$I_0^s = I_\infty^{s-1} \cup (E \cap O^s), \qquad I_i^s = I_{i-1}^s \cup \Pi^s[I_{i-1}^s] \text{ for } i > 0, \qquad \text{and} \qquad I_\infty^s = \bigcup_{i \geq 0} I_i^s. \tag{5}$$

Set I_∞^S is the *materialisation* of Π w.r.t. E and λ. It is known that I_∞^S does not depend on λ, so we write it as $\Pi_\infty[E]$. For s with $0 \leq s \leq S$, we often use the following abbreviations for various subsets of the materialisation:

$$\Pi_\infty[E]^s = \Pi_\infty[E] \cap O^s \qquad \Pi_\infty[E]^{<s} = \Pi_\infty[E] \cap O^{<s} \qquad \Pi_\infty[E]^{\leq s} = \Pi_\infty[E] \cap O^{\leq s}. \tag{6}$$

The inferences of a negation-free Datalog program for each predicate can be characterised by a possibly infinite set of rules obtained by the process of rule unfolding. Specifically, let r and r' be negation-free rules of the forms $B_1 \wedge \cdots \wedge B_m \to H$ and $B_1' \wedge \cdots \wedge B_n' \to H'$, respectively, that do not share a variable. Then, the *unfolding* of r at position i with r' is the rule

$$B_1\sigma \wedge \ldots B_{i-1}\sigma \wedge B_1'\sigma \wedge \cdots \wedge B_n'\sigma \wedge B_{i+1}\sigma \wedge \cdots \wedge B_m\sigma \to H\sigma \tag{7}$$

where σ is the most general unifier of H' and B_i. If r and r' share variables, we simply rename variables in r and apply the unfolding. Now let Π be a negation-free Datalog program, let P be an n-ary predicate, and let U^P be the smallest set of rules that contains $P(x_1, \ldots, x_n) \to P(x_1, \ldots, x_n)$, as well as each unfolding of each rule $r \in U^P$ at each position with each rule $r' \in \Pi$. Then, for each dataset E and constants $a_1, \ldots, a_n$, we have $P(a_1, \ldots, a_n) \in \Pi_\infty[E]$ if and only if $P(a_1, \ldots, a_n) \in U^P[E]$.

4. Motivation

In this section, we discuss the motivation for our work. To this end, in Section 4.1 we present a brief summary of the seminaïve evaluation and its properties. Then, in Section 4.2 we show how seminaïve evaluation can be inefficient on several types of rules commonly used in practice. Finally, in Section 4.3 we show how these inefficiencies can be overcome by deriving the relevant consequences using custom algorithms, and we outline the main challenges to doing so.

4.1. Seminaïve evaluation

The materialisation of a program Π on a set E of explicitly given facts can be computed by applying the rules of Π to E iteratively as suggested in Section 3. Now consider the dataset $I_1 = E \cup \Pi[E]$ obtained after the first application of Π to E: since $E \subseteq I_1$, each rule that is applicable to E is also applicable to I_1; thus, a straightforward computation of $I_2 = I_1 \cup \Pi[I_1]$ would repeat all the work done to compute I_1. Because of this, such a *naïve* evaluation algorithm is not suitable for practical use.

The objective of the seminaïve evaluation [1] is to make materialisation-based reasoning practically feasible by eliminating this source of inefficiency. Its pseudocode is shown in Algorithm 1. The algorithm takes as input a set of explicitly given facts E, a program Π, and a stratification λ of Π, and it computes the materialisation $\Pi_\infty[E]$.

The seminaïve algorithm considers each stratum index s with $1 \leq s \leq S$ (lines 2–7), and for each it applies the rules of stratum Π^s in rounds as long as new facts are derived (lines 4–7). During this process, the auxiliary set Δ contains the facts freshly derived in the most recent round of rule application. Before the first round, set Δ is initialised (line 3) as the

---

Algorithm 1 MAT(Π, λ, E).

1: $I := \emptyset$
2: **for each** stratum index s with $1 \leq s \leq S$ **do**
3:     $\Delta := (E \cap O^s) \cup \Pi_{nr}^s[I]$
4:     **while** $\Delta \neq \emptyset$ **do**
5:         $N := \{h(r\sigma) \mid r \in \Pi_r^s$ and $r\sigma$ is an instance of rule r such that $I \cup \Delta \models b(r\sigma)$ and $I \not\models b^+(r\sigma)\}$
6:         $I := I \cup \Delta$
7:         $\Delta := N \setminus I$

---

union of $E \cap O^s$ (i.e., the explicitly given facts that belong to the stratum with index s and should thus be added to the materialisation) and the consequences of the nonrecursive rules Π_{nr}^s on the facts derived thus far. Since the nonrecursive rules are applied just once, we can use $\Pi_{nr}^s[I]$ from equation (4) without the danger of repeating derivations. Only the recursive rules Π_r^s need to be applied iteratively, and the freshly derived consequences of such rules are added to Δ as input to the next round (line 5). Conditions $I \cup \Delta \models b(r\sigma)$ and $I \not\models b^+(r\sigma)$ express the so-called *nonrepetition property*: in each round, only instances $r\sigma$ of a rule r that become applicable as a result of deriving Δ in the immediately preceding round need to be considered. Motik et al. [65, Section 9.2] discuss several ways to implement this condition efficiently in practice, but the more abstract formulation in line 5 is sufficient for this paper. It is straightforward to see that $I = \Pi_\infty[E]$ holds when the algorithm terminates.

4.2. Shortcomings of seminaïve evaluation

Although seminaïve evaluation does not repeat derivations, it always considers all applicable rule instances. However, in many programs commonly used in practice, the same fact is often derived via multiple, distinct rule instances; this is particularly common with recursive rules, examples of which we present next. Such redundancy, however, is not restricted to just recursive rules: in Section 6.4 we consider a class of rules that can be susceptible to redundant derivations even when they are nonrecursive.

Example 2. Let Π^1 be the program containing rule (8) that axiomatises a binary predicate R as being transitive. Moreover, consider the set of explicitly given facts $E^1 = \{R(a_i, a_{i+1}) \mid 0 \leq i \leq n\}$.

$$R(x, y) \land R(y, z) \to R(x, z) \tag{8}$$

Clearly, $I = \Pi_\infty^1[E^1] = \{R(a_i, a_j) \mid 0 \leq i < j \leq n\}$, so each rule instance of the form

$$R(a_i, a_k) \land R(a_k, a_j) \to R(a_i, a_j) \tag{9}$$

with $1 \leq i < k < j \leq n$ is applicable to I. Thus, each fact $R(a_i, a_j)$ is derived $j - i$ times using *different* instances of rule (8).

Example 3. Let Π^2 be the program containing rules (8) and (10), and let $E^2 = \{R(a_i, a_{i+1}) \mid 1 \leq i < n\} \cup \{R(a_n, a_1)\}$.

$$R(x, y) \to R(y, x) \tag{10}$$

Clearly, $I = \Pi_\infty^2[E^2] = \{R(a_i, a_j) \mid 1 \leq i, j \leq n\}$, so each rule instance of the form (9) with $1 \leq i, j, k \leq n$ is applicable to I. Thus, each fact $R(a_i, a_j)$ is derived n times using *different* instances of rule (8).

Seminaïve evaluation considers each applicable rule instance exactly once, so it runs in time $O(n^3)$ in both cases. In our practical experience, this can be unfeasible when n is in the order of tens of thousands. Thus, Datalog reasoners based on seminaïve evaluation face significant scalability challenges in practice.

4.3. Avoiding redundant derivations

While the programs in Examples 2 and 3 both incur a cubic number of derivations, they derive only quadratically many facts. Thus, it is natural to ask whether the materialisation can be computed without necessarily enumerating all applicable inferences. Transitive closure of a graph with n vertices and m edges can be computed using the Floyd-Warshall algorithm in $O(n^3)$ time, using breath- or depth-first search to identify vertices reachable from every vertex in the graph in $O(n^2 + n \cdot m)$ time, or using $\log n$ matrix multiplications. The current best asymptotic complexity of matrix multiplication is $O(n^{2.3728596})$ [3], but the constant factors make this algorithm unsuitable for practice; moreover, m can be quadratic in n, which suggests that finding an algorithm with worst-case running time better than $O(n^3)$ might be hard. However, we show in Example 4 that, for the dataset from Example 2 (which is arguably relatively common in practice), transitive closure can be computed in $O(n^2)$ time. Moreover, we show in Example 5 that the symmetric–transitive closure of a graph can always be computed in at most $O(n^2)$ steps.

Example 4. The key to evaluating Π^1 more efficiently on E^1 is to distinguish the set of 'external' facts that are given to Π^1 as input from the 'internal' facts derived by Π. If we denote the set of 'external' facts by X, 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 $X = E^1$, so the algorithm considers only rule instances of the form

$$R(a_i, a_{i+1}) \wedge R(a_{i+1}, a_k) \rightarrow R(a_i, a_k) \tag{11}$$

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 (12) instead of rule (8).

$$X(x, y) \wedge R(y, z) \rightarrow R(x, z) \tag{12}$$

Example 5. To evaluate Π^2 more efficiently, we can view predicate R as an undirected graph with n vertices and m edges. Then, we simply compute the connected components of the graph using breadth- or depth-first search, and, for each connected component C, we enumerate all $u, v \in C$ and derive $R(u, v)$. The first step takes $O(n + m)$ time and second step requires $O(n^2)$ time in the worst case; since $m \leq n^2$, the algorithm runs in $O(n^2)$ time on any input.

To summarise, the performance of Datalog materialisation can be significantly improved in many cases by using custom algorithms that do not consider all applicable rule instances. In this paper we address the challenge of integrating such procedures with standard Datalog reasoning. Specifically, our techniques will be applicable when programs such as Π^1 and Π^2 are used together with other rules. Thus, the facts derived by our custom procedures need to be processed by other rules in the program and vice versa. Because of this, we cannot optimise each set of rules in isolation; for example, we cannot simply replace rule (8) with rule (12) as the consequences involving the R predicate derived by other rules would then not match to the first body atom of (8). Moreover, such exchange of facts may need to be repeated iteratively until a fixpoint is reached, so it is important to exchange information in a way that can prevent repeated derivations. Thus, a key challenge in our work is to organise communication of facts between various program subsets in a way that on the one hand is generic and allows for a wide spectrum of custom procedures, and on the other hand guarantees correctness and termination in all cases.

Such inefficiencies can also affect incremental update algorithms, which are critical to practical success of materialisation-based Datalog. Correct handling of fact deletion is a key challenge for incremental updates. For example, consider applying rules $R(x, y) \rightarrow T(x)$ and $S(x, y) \rightarrow T(x)$ to a set of explicitly given facts $R(a, b)$ and $S(a, c)$. When $R(a, b)$ is deleted, fact $T(a)$ may need to be deleted as well because it is a consequence of $R(a, b)$ via the first rule. However, deletion should be performed only if there are no alternative derivations from the remaining explicitly given facts. In this example, $T(a)$ is derivable from $S(x, y) \rightarrow T(x)$ and $S(a, c)$, and hence $T(a)$ does not need to be deleted. Various algorithms differ in how they verify the existence of alternative derivations. The DRed algorithm solves this problem by first *overdeleting* all consequences of the deleted facts, and then *rederiving* the facts that still hold after deletion. In our example, $T(a)$ would first be deleted; however, since $T(a)$ can be derived using rules $R(a, y) \rightarrow T(a)$ and $S(a, y) \rightarrow T(a)$, the algorithm would evaluate the bodies of these rules as queries, and the evaluation of the second rule would reveal that the fact should be rederived. The rederived facts are then added back to the materialisation alongside any explicitly added facts, and the rules are applied iteratively until a fixpoint is reached. The other incremental update algorithms, such as B/F, FBF, DRed^c, and B/F^c (see Section 2), refine this process to various degrees with the objective of making the overdeletion phase more precise and thus reducing the overall work.

All incremental update algorithms known to us realise overdeletion and rederivation using variants of seminaïve evaluation, and so they are also susceptible to redundant derivations. We evaluated DRed and B/F experimentally in our earlier work [65], and our results show that, despite all optimisations, certain rule combinations can be very hard: the time to delete even a fraction of a dataset can be of the same order of magnitude as the time for the initial materialisation. Intuitively, when rules are complex, deleting even just a few facts can lead to overdeletion of a large number of derived facts, most of which need to be rederived. Each of these phases can thus suffer from the drawbacks outlined Section 4.2. As we show in this paper, these drawbacks can be overcome by using custom rules; however, supporting incremental updates is also the main source of complexity in our work.

5. A framework for modular reasoning

We now present a framework for materialisation and incremental updates that can avoid the deficiencies outlined in Section 4. In Section 5.1 we discuss the principles and intuitions, and in Sections 5.2–5.5 we present and discuss various technical details.

5.1. Roadmap

To handle certain rule combinations using custom algorithms, we require the input Datalog program to be partitioned into disjoint sets of rules that we call *modules*. In this paper, we assume that a partition of a program into modules is given

explicitly in the input: developing algorithms for automatic extraction of modules from a given Datalog program is an interesting problem that can be considered independently in future. Our materialisation and incremental update algorithms do not apply the modules' rules directly (e.g., using seminaïve evaluation). Instead, the computation of certain consequences of the rules is delegated to three *module functions*, which handle addition, overdeletion, and rederivation of facts. For example, the module function for addition is given all facts derived thus far and the facts derived in the most recent round of rule application, and the function must produce the consequences of the module rules for the next round. The main task of our modular reasoning framework is to orchestrate the exchange of facts among functions of different modules in a way that guarantees correctness and efficiency.

To ensure that we can combine different modules in an arbitrary way, we will formalise properties that module functions must satisfy to guarantee correctness of our algorithms. Note that the optimisations described in Examples 2 and 3 do not compute just the immediate consequences of the module's rules; for example, our optimisation for symmetric–transitive closure is effective because the entire closure can be computed at once. Thus, module functions must be allowed to compute more than just the immediate consequences of the module's rules. At the same time, we do not wish to impose too stringent requirements on what needs to be computed. This is particularly important for overdeletion: a fact can be deleted only if no alternative derivations remain after the update, and there are many ways to balance the tradeoffs of identifying such derivations eagerly versus overdeleting and then rederiving the fact. To capture the many variants that can be used, we formalise our conditions on the output of module functions in terms of the lower and upper bounds, J_l and J_u, respectively. Roughly speaking, the lower bound J_l will correspond to the facts obtained by applying the rules only once, and the upper bound J_u will correspond to the maximum set of facts that can be returned while maintaining correctness. We prove that, as long as the output of each module function satisfies the respective lower and upper bounds, our modular algorithms are correct and terminating.

To simplify the presentation of our framework, we assume that the input program is partitioned completely into modules—that is, our framework never applies the rules directly. However, we can always collect all rules for which no specialised algorithms are available in one module per stratum, and we can realise the appropriate module functions using seminaïve evaluation. In fact, the module functions can also incorporate optimisations of overdeletion and rederivation from algorithms such as DRed^c, B/F, B/F^c, and FBF, all of which are captured by our lower and upper bounds. Thus, using customised procedures for certain rules does not need to come at the expense of suboptimal handling of general Datalog rules.

The lower and upper bound requirements provide us with a basis for proving correctness of module functions. Contrary to what one might intuitively expect, correctness of a module function can usually not be proved by considering each function call in isolation. First, to capture the many variants of incremental reasoning considered in the literature, the bounds for overdeletion and rederivation do not depend on just the arguments passed in each call of the module function, but also on the target state of the materialisation. Since realistic module functions have no access to the target materialisation state, it may be hard to show that the output of a function call satisfies the required conditions based solely on the function call arguments. Second, to support efficient updates, most module functions will often need to maintain auxiliary information of arbitrary structure, and the output of a module function will generally depend on the state of this auxiliary information before each function call.

We will therefore need a more sophisticated approach to proving correctness of module functions, where we consider how successive calls to module functions interact. In particular, we shall introduce the notion of a *compatible call history*, which is effectively a sequence of possible calls to the module functions over a module's lifetime. This notion will allow us to prove the correctness of module functions more easily: we consider an arbitrary compatible call history, and we show by induction on the call history length that each call in the history satisfies the lower and upper bounds identified earlier.

The rest of this section is organised as follows. In Section 5.2 we define the notions of modules and module functions, and we also formalise the lower and upper bound requirements. Next, in Sections 5.3 and 5.4 we present our materialisation and incremental update algorithms, respectively. Finally, in Section 5.5 we formulate conditions that calls to module functions must satisfy, and we show that these conditions guarantee correctness of our algorithms.

5.2. Modules and module functions

We now define a key notion of a module. This notion should not be confused with ontology modules, which are subsets of an ontology that are semantically independent from each other in a well-defined way.

Definition 6. A *module* is a Datalog program such that no predicate occurring in a negative literal also occurs in a rule head.

Definition 7 introduces a notion of a module partition, which captures how modules are used in our framework: a Datalog program must be partitioned into disjoint modules so that each module is fully contained in a single stratum.

Definition 7. Let Π be a program, let λ be a stratification of Π, and let S the maximum stratum index of λ. A *module partition* of Π w.r.t. λ consists of modules $M^{s,k}$ with $1 \leq s \leq S$ and $1 \leq k \leq n_s$ such that $M^{s,1} \cup \cdots \cup M^{s,n_s} = \Pi^s$ and $M^{s,k} \cap M^{s,k'} = \emptyset$ for each $1 \leq s \leq S$ and all $1 \leq k < k' \leq n_s$, where n_s is the number of modules used for the stratum with index s.

In other words, each stratum Π^s is partitioned into modules $M^{s,1}, \dots, M^{s,n_s}$. Our framework supports reasoning with Π using custom functions that compute the relevant consequences of modules $M^{s,k}$. As we explained in Section 4.3, most incremental update algorithms consist of an overdeletion, rederivation, and addition phase. Our framework follows these principles and requires each module $M^{s,k}$ to supply three *module functions*, $\mathsf{Add}^{M^{s,k}}$, $\mathsf{Del}^{M^{s,k}}$, and $\mathsf{Red}^{M^{s,k}}$, that will be used in the respective phases. Before presenting a definition of these functions, we first introduce three auxiliary operators that we will use to formalise the conditions on the correctness of module functions.

Definition 8. For M a module and I, Δ^-, and Δ^+ datasets such that $\Delta^- \subseteq I$ and $\Delta^+ \cap I = \emptyset$, operator $M_\mathsf{A}[I : \Delta^-, \Delta^+]$ is defined as

$$M_\mathsf{A}[I : \Delta^-, \Delta^+] = \bigcup_{r \in M} \left\{ \begin{array}{l} \mathsf{h}(r\sigma) \mid r\sigma \text{ is an instance of } r \text{ such that } I \not\models \mathsf{b}(r\sigma), \\ \quad (I \setminus \Delta^-) \cup \Delta^+ \models \mathsf{b}(r\sigma), \text{ and } \mathsf{h}(r\sigma) \notin (I \setminus \Delta^-) \cup \Delta^+ \end{array} \right\}. \tag{13}$$

Intuitively, operator $M_\mathsf{A}[I : \Delta^-, \Delta^+]$ computes the newly derived facts in one round of rule application of the seminaïve algorithm. It takes as input a materialisation I as computed thus far, and sets of facts Δ^- and Δ^+ that were scheduled in the last round of rule application to be removed from and added to I, respectively; thus, $(I \setminus \Delta^-) \cup \Delta^+$ is the materialisation after applying the changes from last round of rule application. The operator identifies the facts that should be added to $(I \setminus \Delta^-) \cup \Delta^+$ as a result of this change; intuitively, these are obtained by instances of the rules of M whose bodies hold after, but not before the change.

Definition 9. For M a module and I^o, I^n, Δ^-, and Δ^+ datasets such that $\Delta^- \subseteq I^\mathsf{n}$ and $\Delta^+ \cap I^\mathsf{n} = \emptyset$, operator $M_\mathsf{D}[I^\mathsf{o}, I^\mathsf{n} : \Delta^-, \Delta^+]$ is defined as

$$M_\mathsf{D}[I^\mathsf{o}, I^\mathsf{n} : \Delta^-, \Delta^+] = \bigcup_{r \in M} \left\{ \begin{array}{l} \mathsf{h}(r\sigma) \mid r\sigma \text{ is an instance of } r \text{ such that } I^\mathsf{o} \models \mathsf{b}(r\sigma), \ I^\mathsf{n} \models \mathsf{b}(r\sigma), \\ \quad (I^\mathsf{n} \setminus \Delta^-) \cup \Delta^+ \not\models \mathsf{b}(r\sigma), \text{ and } \mathsf{h}(r\sigma) \in (I^\mathsf{n} \setminus \Delta^-) \cup \Delta^+ \end{array} \right\}. \tag{14}$$

Intuitively, operator $M_\mathsf{D}[I^\mathsf{o}, I^\mathsf{n} : \Delta^-, \Delta^+]$ captures one-step overdeletion from the original DRed algorithm. It takes as input the 'old' materialisation I^o (i.e., the materialisation before any updates were applied), the 'new' materialisation I^n as computed thus far, and sets of facts Δ^- and Δ^+ that were scheduled in the last round of rule application to be removed from and added to I^n, respectively; thus, $(I^\mathsf{n} \setminus \Delta^-) \cup \Delta^+$ is the 'new' materialisation after applying the changes from last round of rule application. The operator identifies the facts that should be removed from $(I^\mathsf{n} \setminus \Delta^-) \cup \Delta^+$ as a result of this change; intuitively, these are obtained by instances of the rules of M whose bodies hold in I^o (thus ensuring that overdeletion is restricted to the consequences of M before any incremental updates are applied), as well as in I^n before, but not after the change.

Definition 10. For M a module and I^o, I^n, and Δ datasets such that $\Delta \subseteq I^\mathsf{o} \setminus I^\mathsf{n}$, operator $M_\mathsf{R}[I^\mathsf{o}, I^\mathsf{n} : \Delta]$ is defined as

$$M_\mathsf{R}[I^\mathsf{o}, I^\mathsf{n} : \Delta] = \bigcup_{r \in M} \left\{ \mathsf{h}(r\sigma) \mid r\sigma \text{ is an instance of } r \text{ such that } I^\mathsf{o} \models \mathsf{b}(r\sigma), \ I^\mathsf{n} \models \mathsf{b}(r\sigma), \text{ and } \mathsf{h}(r\sigma) \in \Delta \right\}. \tag{15}$$

Intuitively, operator $M_\mathsf{R}[I^\mathsf{o}, I^\mathsf{n} : \Delta]$ handles rederivation. It takes as input the 'old' materialisation I^o, the 'new' materialisation I^n after overdeletion, and a set of overdeleted facts Δ. It returns the facts of Δ that should be rederived; intuitively, these are obtained by instances of the rules in M that hold in both I^o and I^n.

We are now ready to formalise our notion of module functions. Definition 11 formalises the conditions on the arguments and the result of each module function call.

Definition 11. The *implementation* of a module M consists of *module functions* Add^M, Del^M, and Red^M. These functions can be called as shown in the first column of Table 1, where datasets I, I^o, I^n, Δ^-, Δ^+, Δ^m, and Δ are the arguments and J is the result of a call. Given a dataset E, a program Π, a stratification λ, and a stratum index s of λ such that $M \subseteq \Pi^s$, a call is *correct in the context of* E, Π, λ, and s if the arguments to the call satisfy the conditions from the second column of Table 1, and the output J satisfies $J_l \subseteq J \subseteq J_u$ for J_l the *lower bound* and J_u the *upper bound* from the third and fourth column of Table 1, respectively.[1]

An important objective of our work is to devise a general framework that can capture and be combined with most known reasoning algorithms. To attain this goal, we cannot define the lower and upper bounds on module function results

---

[1] Please remember that abbreviations $\Pi_\infty[E]^s$, $\Pi_\infty[E]^{<s}$, and $\Pi_\infty[E]^{\leq s}$ used in Table 2 were defined in equation (6) in Section 3.

Table 1
Module function calls.

Call	Conditions on arguments	Lower bound J_l	Upper bound J_u
$J := \mathsf{Add}^M[I \vdots \Delta^-, \Delta^+ \vdots \Delta^m]$	$\Delta^- \subseteq I,\ \Delta^+ \cap I = \emptyset,\ \Delta^m \subseteq \Delta^+$	$M_\mathsf{A}[I \vdots \Delta^-, \Delta^+]$	$\Pi_\infty[E]^s \setminus ((I \setminus \Delta^-) \cup \Delta^+)$
$J := \mathsf{Del}^M[I^o, I^n \vdots \Delta^-, \Delta^+ \vdots \Delta^m]$	$\Delta^- \subseteq I^n,\ \Delta^+ \cap I^n = \emptyset,\ \Delta^m \subseteq \Delta^-$	$M_\mathsf{D}[I^o, I^n \vdots \Delta^-, \Delta^+] \setminus \Pi_\infty[E]$	$I^o \cap O^s \cap ((I^n \setminus \Delta^-) \cup \Delta^+)$
$J := \mathsf{Red}^M[I^o, I^n \vdots \Delta]$	$\Delta \subseteq (I^o \setminus I^n) \cap O^s$	$M_\mathsf{R}[I^o, I^n \vdots \Delta]$	$\Pi_\infty[E]^s \setminus I^n$

purely in terms of function arguments; rather, the correctness of a call to a module function can only be judged in context of the entire program Π and the 'current' set of explicit facts E whose materialisation is being computed. This is the main difficulty in the technical development of our framework, and we discuss it in depth in Section 5.5. First, however, we discuss the intuition behind our functions and present our materialisation and incremental reasoning algorithms.

Function Add^M handles fact addition and is used by both the materialisation and incremental update algorithms. Intuitively, it computes facts that should be added to the materialisation as a result of deleting facts in Δ^- and adding facts in Δ^+. Set Δ^m will contain the facts derived by module M in the last round of rule application, which will allow the function to avoid recomputing its own consequences and thus realise a form of nonrepetition property. The function must return at least $M_\mathsf{A}[I \vdots \Delta^-, \Delta^+]$—that is, the facts that would be added by applying seminaïve evaluation to M. However, the function can return more: correctness is guaranteed as long as the result is contained in $\Pi_\infty[E]^s$, and in fact Add^M can even return facts derivable by rules outside M (although this is unlikely in practice). Finally, Add^M is not allowed to return facts contained in $(I \setminus \Delta^-) \cup \Delta^+$, which ensures termination by preventing the function from returning the same fact twice during materialisation or incremental reasoning.

Function Del^M handles fact overdeletion and is used only by our incremental update algorithm. Intuitively, it computes facts that should be deleted from the materialisation as a result of changing the current 'new' materialisation I^n by deleting Δ^- and adding Δ^+. The function also accepts the 'old' materialisation I^o, and set Δ^m plays the same role as in the case of Add^M. The function must return any fact that (i) would be overdeleted in the original DRed algorithm and (ii) does not hold in the new materialisation $\Pi_\infty[E]$. The latter condition is essential for generality as it allows Del^M to incorporate various optimisations of overdeletion. Again, the function is allowed to overdelete more: any fact from the 'old' materialisation can be returned. Finally, to ensure termination, the function is not allowed to overdelete facts that have already been overdeleted.

Function Red^M handles rederivation and is also used only by our incremental update algorithm. Intuitively, it computes the facts that hold in the 'old' materialisation I^o and the 'new' materialisation I^n as computed thus far. Operator $M_\mathsf{R}[I^o, I^n \vdots \Delta]$ provides us with the lower bound, where set Δ identifies the facts whose status must be determined. However, function Red^M is allowed to rederive other facts that hold after deletion, and the upper bound is analogous to that of Add^M.

If no specialised procedures are available for a module M, functions Add^M, Del^M, and Red^M can be implemented generically as $M_\mathsf{A}[I \vdots \Delta^-, \Delta^+]$, $M_\mathsf{D}[I^o, I^n \vdots \Delta^-, \Delta^+]$, and $M_\mathsf{R}[I^o, I^n \vdots \Delta]$, respectively; if no other modules are used, our algorithms reduce to seminaïve evaluation and DRed. The B/F [64] and FBF [65] algorithms are obtained by modifying $M_\mathsf{D}[I^o, I^n \vdots \Delta^-, \Delta^+]$ so that it searches eagerly for alternative derivations, and the B/F^c and DRed^c [43] algorithms can be obtained by associating facts with derivation counters that are maintained by all functions. Such generic modules can be freely combined with optimised modules that we discuss later in this paper, which makes our framework applicable to a wide range of scenarios.

5.3. Computing the materialisation

Algorithm 2 takes as input a set of explicitly given facts E and a module partition of a Datalog program Π w.r.t. a stratification λ, and it computes the materialisation I of Π and E. Correctness of the algorithm follows from Theorem 15 in Section 5.5. The algorithm proceeds in a stratum-by-stratum manner similar to Algorithm 1. For each stratum, the algorithm first adds the explicit facts in the current stratum ($E \cap O^s$) to the materialisation I (line 10). Function $\mathsf{Add}^{M^{s,k}}$ is then called once for each module to identify the consequences of both the nonrecursive and the recursive rules of $M^{s,k}$ on I (line 12), followed by a loop (lines 13–18) where the recursive rules are evaluated up to the least fixpoint. In each iteration, the algorithm first combines the consequences of all modules (line 15) and adds them to the materialisation I (line 14). Then, function $\mathsf{Add}^{M^{s,k}}$ is called for each module to identify the consequences of $M^{s,k}$ that should be added to I. In this step, Δ_k contains the consequences of the module computed in the most recent completed round; a module function can use this set to prevent redundant derivations, as well as to detect that only recursive rules of $M^{s,k}$ need to be considered. The loop terminates when no new facts are derived (line 16).

5.4. Incremental updates

Algorithm 3 shows how to compute incremental updates in our framework. Correctness of the algorithm follows from Theorem 15 in Section 5.5. The algorithm takes as input a module partition of a program Π w.r.t. a stratification λ, a set of explicitly given facts E, the materialisation $I = \Pi_\infty[E]$, and two sets of facts E^- and E^+ that are to be deleted from and

Algorithm 2 M-Mat(Π, λ, E).

8:   $I := \emptyset$
9:   **for each** stratum index s with $1 \leq s \leq S$ **do**
10:     $\Delta := E \cap O^s$
11:     **for each** k with $1 \leq k \leq n_s$ **do**
12:       $\Delta^k := \mathsf{Add}^{M^{s,k}}[\emptyset \vdots \emptyset, I \cup \Delta \vdots \emptyset]$
13:     **loop**
14:       $I := I \cup \Delta$
15:       $\Delta := \Delta^1 \cup \cdots \cup \Delta^{n_s}$
16:       **if** $\Delta = \emptyset$ **then break**
17:       **for each** k with $1 \leq k \leq n_s$ **do**
18:         $\Delta^k := \mathsf{Add}^{M^{s,k}}[I \vdots \emptyset, \Delta \vdots \Delta^k]$

Algorithm 3 M-Upd($\Pi, \lambda, E, I, E^-, E^+$).

19:   $D := A := \emptyset, \quad E^- := (E^- \cap E) \setminus E^+, \quad E^+ := E^+ \setminus E$
20:   **for each** stratum index s with $1 \leq s \leq S$ **do**
21:     Overdelete
22:     Rederive
23:     Add
24:   $E := (E \setminus E^-) \cup E^+, \quad I := (I \setminus D) \cup A$

25:   **procedure** Overdelete
26:     $\Delta := E^- \cap O^s$
27:     **for each** k with $1 \leq k \leq n_s$ **do**
28:       $\Delta^k := \mathsf{Del}^{M^{s,k}}[I, I \vdots (D \setminus A) \cup \Delta, A \setminus D \vdots \emptyset]$
29:     **loop**
30:       $D := D \cup \Delta$
31:       $\Delta := \Delta^1 \cup \cdots \cup \Delta^{n_s}$
32:       **if** $\Delta = \emptyset$ **then break**
33:       **for each** k with $1 \leq k \leq n_s$ **do**
34:         $\Delta^k := \mathsf{Del}^{M^{s,k}}[I, (I \setminus D) \cup A \vdots \Delta, \emptyset \vdots \Delta^k]$

35:   **procedure** Rederive
36:     **for each** i with $1 \leq k \leq n_s$ **do**
37:       $\Delta^k := \mathsf{Red}^{M^{s,k}}[I, (I \setminus D) \cup A \vdots (D \cap O^s) \setminus (E \setminus E^-)]$
38:     $\Delta := ((E \setminus E^-) \cap D \cap O^s) \cup \Delta^1 \cup \cdots \cup \Delta^{n_s}$

39:   **procedure** Add
40:     $\Delta := \Delta \cup (E^+ \cap O^s) \setminus ((I \setminus D) \cup A)$
41:     **for each** k with $1 \leq k \leq n_s$ **do**
42:       $\Delta^k := \mathsf{Add}^{M^{s,k}}[I \vdots D \setminus A, (A \setminus D) \cup \Delta \vdots \Delta_k]$
43:     **loop**
44:       $A := A \cup \Delta$
45:       $\Delta := \Delta^1 \cup \cdots \cup \Delta^{n_s}$
46:       **if** $\Delta = \emptyset$ **then break**
47:       **for each** k with $1 \leq k \leq n_s$ **do**
48:         $\Delta^k := \mathsf{Add}^{M^{s,k}}[(I \setminus D) \cup A \vdots \emptyset, \Delta \vdots \Delta^k]$

added to E, respectively. The algorithm updates I to the 'new' materialisation $\Pi_\infty[(E \setminus E^-) \cup E^+]$. During the computation, the algorithm frequently needs to refer to both the 'old' and the 'new' materialisation—that is, the materialisation before and after update. To support this, the algorithm computes sets D and A of facts to be deleted from and added to I, respectively. Thus, during the algorithm's computation, I is the 'old' materialisation, and, after a stratum with index s has been processed, $((I \cap O^{\leq s}) \setminus D) \cup A$ is the 'new' materialisation up to stratum index s—that is, $((I \cap O^{\leq s}) \setminus D) \cup A = \Pi_\infty[(E \setminus E^-) \cup E^+]^{\leq s}$ holds at that point. Thus, upon termination, $(I \setminus D) \cup A$ is the 'new' materialisation so the algorithm updates E and I accordingly in line 24. Our algorithm processes the strata of Π and, in each one, it overdeletes facts, rederives facts that hold after overdeletion, and finally inserts the explicitly added facts and computes their closure. We next discuss these three phases in more detail.

At the start of the overdeletion phase for a stratum with index s, sets $D \setminus A$ and $A \setminus D$ contain facts deleted from and added to the previous strata, respectively; moreover, in line 26, Δ is set to the set of facts deleted from the stratum with index s. Thus, the initial calls to $\mathsf{Del}^{M^{s,k}}$ (lines 27–28) identify (at least) each fact that no longer holds because it is derived by an instance of a rule in $M^{s,k}$ that depends positively on $(D \setminus A) \cup \Delta$ or negatively on $A \setminus D$. The facts obtained from each module $M^{s,k}$ are stored in a separate set Δ^k, which will be passed back to $\mathsf{Del}^{M^{s,k}}$ on the subsequent call (lines 33–34); this allows each module to identify its own conclusions from the most recent round of rule application and thus possibly prevent redundant derivations. Next, the algorithm computes iteratively further facts that might need to be removed (lines 29–34). In each iteration, the current set Δ is rolled into the set D of deleted facts (line 30), the consequences of all modules from the previous round of rule application are combined into a new set Δ (line 31), and function $\mathsf{Del}^{M^{s,k}}$ is called for each module to identify new consequences of module $M^{s,k}$ that should be overdeleted due to the deletion of Δ (lines 33–34). These steps are repeated as long as at least one module identifies further deletion candidates.

Next, the algorithm identifies the rederivable facts by calling $\mathsf{Red}^{M^{s,k}}$ for each module (lines 36–37). All facts that are explicitly given after the update are rederived as well by adding them to Δ (line 38).

Finally, the algorithm proceeds with the addition phase. First, Δ is extended with all explicitly inserted facts (line 40). Then, function $\mathsf{Add}^{M^{s,k}}$ is called once for each module (lines 41–42) to identify consequences of $M^{s,k}$ that should be added either because facts in $D \setminus A$ from a previous stratum were deleted, or facts in $(A \setminus D) \cup \Delta$ were added. The current set Δ is added to A (line 44), the consequences of all modules from the previous round of rule application are combined into a new set Δ (line 45), and function $\mathsf{Add}^{M^{s,k}}$ is called for each module to identify new consequences of module $M^{s,k}$ that should be added due to the addition of Δ (lines 47–48). These steps are repeated as long as at least one module identifies facts that need to be inserted.

5.5. Correctness of the modular reasoning framework

To ensure correctness of our algorithms, each call to a module function made during an execution of Algorithm 2 or Algorithm 3 must satisfy the lower and upper bounds from Table 1. However, proving this is usually not straightforward.

First, the definitions of the lower bound for Del^M and the upper bounds for Add^M and Red^M refer to the context in which module M is used. This context involves the Datalog program Π being materialised, the stratum s of Π to which module M belongs, and the target set of explicitly given facts E (i.e., the set of explicitly given facts after the update); this information is used to determine the 'new' materialisation $\Pi_\infty[E]$. Note, however, that the context information is not passed to the module functions: a module function should concern itself solely with the rules of its module, rather than how a module is used. For example, if we implement Add^M using rule application as in Definition 8, the implementation just needs to apply the rules of M, and not concern itself with the rules in $\Pi \setminus M$. Moreover, module functions are unlikely to have access to the 'new' materialisation $\Pi_\infty[E]$. These observations, however, prevent us from proving correctness of a module function by considering each function call in isolation. Instead, for each context in which a module function might be called, we need to show that each call made in such a context satisfies the lower and upper bounds. For example, assume we wish to prove that the generic implementation of Add^M satisfies the upper bound—that is, $M_\mathsf{A}[I \vdots \Delta^-, \Delta^+] \subseteq \Pi_\infty[E]^s \setminus ((I \setminus \Delta^-) \cup \Delta^+)$. First, we need to consider any program Π such that $M \subseteq \Pi$, any stratification λ that places M in stratum s of Π, and any set of 'target' explicitly given facts E. Moreover, we need to consider only arguments that can actually be encountered during a run of Algorithm 2 or 3: the required property does not hold if, for example, I contains facts outside $\Pi_\infty[E]$. This, however, raises the question of what I, Δ^-, and Δ^+ can function Add^M legitimately encounter when it is called.

Second, all module functions we present in Sections 6.2–6.5 maintain various internal data structures. Thus, the correctness of a module function call will generally depend on the state of these data structures at the time of a call, and this state will further depend on all previous calls to functions of this and other modules. For example, as we discussed in Example 5, computing the symmetric–transitive closure of a binary predicate can be optimised by computing the set of connected components of the predicate and then deriving connections among the elements of each component. Recomputing the set of connected components each time a module function is called would be very inefficient. To overcome this drawback, our module functions in Section 6.3 maintain the list of connected components in private data structures. Hence, the correctness of each function call critically depends on this list correctly reflecting the state of the materialisation computed thus far. Moreover, to ensure that the list is correctly maintained, the rederivation must be called exactly once after overdeletion and before addition. Note however, that we cannot make any general assumptions about the nature of such information.

The above discussion suggests that proving correctness of each function call in isolation might be difficult or even impossible. Instead, we need to consider all possible contexts in which a module can be used, and we also need to consider all possible sequences of module function calls that can be produced by our algorithm; then, for each call in such a sequence, we need to prove that the call satisfies the lower and upper bound from Table 1. At the same time, we should be able to prove correctness of each module in isolation—that is, without considering other modules that might be used alongside the module of interest.

To achieve these objectives, we next analyse the behaviour of Algorithms 2 and 3 in order to capture the structure of allowed sequences of calls to the functions of one module. This will involve specifying the allowed order of calls to the functions of one module, as well as identifying conditions that relate the arguments of adjacent function calls. Towards this goal, we first introduce the notion of a *call history* for a module.

Definition 12. A *call history* H of *length* m for a module M is a finite, nonempty sequence of the form (16), where each Q_i with $0 \leq i \leq m$ is a finite, nonempty sequence of the form (17), and each $C_{i,j}$ with $1 \leq j \leq h_i$ is a call to a module function for M.

$$H = Q_0, \ldots, Q_m \tag{16}$$

$$Q_i = C_{i,1}, \ldots, C_{i,h_i} \tag{17}$$

Intuitively, a call history H consists of a series of runs Q_i with $0 \leq i \leq m$. Run Q_0 captures the calls to the functions of module M made during the initial materialisation, whereas each Q_i with $i \geq 1$ captures the calls made during subsequent incremental updates. Each run Q_i is simply a sequence of calls $C_{i,j}$ of the form shown in Table 1. The number of calls made in different runs can clearly vary, so h_i provides us with the number of calls in run Q_i.

As we have already suggested, our algorithms will not produce arbitrary call histories: only histories satisfying the conditions from the following definition need to be considered.

Definition 13. A call history H for a module M is *compatible* with a program Π, a stratification λ of Π, a stratum index s of λ, and a sequence of datasets $E_0, \ldots, E_m$ if

- $M \subseteq \Pi^s$,
- the length of H is m,
- each call $C_{i,j}$ in H for $0 \leq i \leq m$ and $1 \leq j \leq h_i$ satisfies the conditions from one row of Table 2.

Table 2
Conditions on the structure of a call sequence.

	Call $C_{i,j}$ can be of the form...	...in which case its arguments must satisfy conditions from Table 1 and...	
A1	$J_{0,1} := \mathsf{Add}^M[\emptyset \vdots \emptyset, \Delta_{0,1}^+ \vdots \emptyset]$ for $i = 0$ and $j = 1$	• $\Delta_{0,1}^+ = \Pi_\infty[E_0]^{<s} \cup (E_0 \cap \mathsf{O}^s)$	(A1.a)
A2	$J_{i,j} := \mathsf{Add}^M[I_{i,j-1} \vdots \emptyset, \Delta_{i,j}^+ \vdots \Delta_{i,j}^m]$ for $i \geq 0$ and $j > 1$	• $C_{i,j-1}$ is of the form $J_{i,j-1} := \mathsf{Add}^M[I_{i,j-1} \vdots \Delta_{i,j-1}^-, \Delta_{i,j-1}^+ \vdots \Delta_{i,j-1}^m]$	(A2.a)
		• $I_{i,j} = (I_{i,j-1} \setminus \Delta_{i,j-1}^-) \cup \Delta_{i,j-1}^+$	(A2.b)
		• $\Delta_{i,j}^m = J_{i,j-1} \subseteq \Delta_{i,j}^+ \subseteq \Pi_\infty[E_i]^s$	(A2.c)
D1	$J_{i,1} := \mathsf{Del}^M[I_{i,1}^o, I_{i,1}^n \vdots \Delta_{i,1}^-, \Delta_{i,1}^+ \vdots \emptyset]$ for $i \geq 1$ and $j = 1$	• $C_{i-1,h_{i-1}}$ is of the form $J_{i-1,h_{i-1}} := \mathsf{Add}^M[I_{i-1,h_{i-1}} \vdots \Delta_{i-1,h_{i-1}}^-, \Delta_{i-1,h_{i-1}}^+ \vdots \Delta_{i-1,h_{i-1}}^m]$	(D1.a)
		• $J_{i-1,h_{i-1}} = \emptyset$	(D1.b)
		• $I_{i,1}^o \cap \mathsf{O}^{\leq s} = ((I_{i-1,h_{i-1}} \setminus \Delta_{i-1,h_{i-1}}^-) \cup \Delta_{i-1,h_{i-1}}^+) \cap \mathsf{O}^{\leq s}$	(D1.c)
		• $I_{i,1}^o = I_{i,1}^n = \Pi_\infty[E_{i-1}]$	(D1.d)
		• $\Delta_{i,1}^- = (\Pi_\infty[E_{i-1}]^{<s} \setminus \Pi_\infty[E_i]^{<s}) \cup ((E_i \setminus E_{i-1}) \cap \mathsf{O}^s)$	(D1.e)
		• $\Delta_{i,1}^+ = \Pi_\infty[E_i]^{<s} \setminus \Pi_\infty[E_{i-1}]^{<s}$	(D1.f)
D2	$J_{i,j} := \mathsf{Del}^M[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}^-, \emptyset \vdots \Delta_{i,j}^m]$ for $i \geq 1$ and $j > 1$	• $C_{i,j-1}$ is of the form $J_{i,j-1} := \mathsf{Del}^M[I_{i,j-1}^o, I_{i,j-1}^n \vdots \Delta_{i,j-1}^-, \Delta_{i,j-1}^+ \vdots \Delta_{i,j-1}^m]$	(D2.a)
		• $I_{i,j}^o = \Pi_\infty[E_{i-1}]$	(D2.b)
		• $I_{i,j}^n = (I_{i,j-1}^n \setminus \Delta_{i,j-1}^-) \cup \Delta_{i,j-1}^+$	(D2.c)
		• $\Delta_{i,j}^m = J_{i,j-1} \subseteq \Delta_{i,j}^- \subseteq \Pi_\infty[E_{i-1}]^s$	(D2.d)
R	$J_{i,j} := \mathsf{Red}^M[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}]$ for $i \geq 1$ and $j > 1$	• $C_{i,j-1}$ is of the form $J_{i,j-1} := \mathsf{Del}^M[I_{i,j-1}^o, I_{i,j-1}^n \vdots \Delta_{i,j-1}^-, \Delta_{i,j-1}^+ \vdots \Delta_{i,j-1}^m]$	(R.a)
		• $J_{i,j-1} = \emptyset$	(R.b)
		• $I_{i,j}^o = \Pi_\infty[E_{i-1}]$	(R.c)
		• $I_{i,j}^n = (I_{i,j-1}^n \setminus \Delta_{i,j-1}^-) \cup \Delta_{i,j-1}^+$	(R.d)
		• $I_{i,j}^n \cap \mathsf{O}^{<s} = \Pi_\infty[E_i]^{<s} \subseteq I_{i,j}^n \cap \mathsf{O}^{\leq s} \subseteq \Pi_\infty[E_i]^{\leq s}$	(R.e)
		• $\Delta_{i,j} = \Pi_\infty[E_{i-1}]^s \setminus (I_{i,j}^n \cup (E_{i-1} \cap E_i))$	(R.f)
A3	$J_{i,j} := \mathsf{Add}^M[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^m]$ for $i \geq 1$ and $j > 1$	• $C_{i,j-1}$ is of the form $J_{i,j-1} := \mathsf{Red}^M[I_{i,j-1}^o, I_{i,j-1}^n \vdots \Delta_{i,j-1}]$	(A3.a)
		• $I_{i,j} = I_{i,j-1}^o = \Pi_\infty[E_{i-1}]$	(A3.b)
		• $\Delta_{i,j}^- \cap \mathsf{O}^{<s} = \Pi_\infty[E_{i-1}]^{<s} \setminus \Pi_\infty[E_i]^{<s}$	(A3.c)
		• $\Delta_{i,j}^- = \Pi_\infty[E_{i-1}]^{\leq s} \setminus I_{i,j-1}^n$	(A3.d)
		• $\Delta_{i,j}^+ \cap \mathsf{O}^{<s} = \Pi_\infty[E_i]^{<s} \setminus \Pi_\infty[E_{i-1}]^{<s}$	(A3.e)
		• $\Delta_{i,j}^m = J_{i,j-1} \subseteq J_{i,j-1} \cup ((E_i \cap \mathsf{O}^s) \setminus I_{i,j-1}^n) \subseteq \Delta_{i,j}^+ \subseteq \Pi_\infty[E_i]^{\leq s}$	(A3.f)

Definition 13 can intuitively be understood as follows. First, program Π, stratification λ, stratum index s, and datasets $E_0, \ldots, E_m$ all provide the context for the function calls. Note that we need a sequence of explicitly given datasets: each E_i provides the context for the calls in run Q_i of H, so the number of runs in H must agree with the number of datasets. Moreover, stratification λ must assign all rules of M into a stratum with index s. Finally, Table 2 identifies the six types of calls that our algorithms can make. As we shall see, a call of type A3 follows a call of type R, so we place it at the end of Table 2 rather than with other calls of Add^M. We next discuss the possible types of calls.

- A call of type A1 is the first call of run Q_0, so it corresponds to the call made in line 12 of Algorithm 2 during initial materialisation. Condition (A1.a) captures the property established in line 10.
- By condition (A2.a), a call of type A2 follows another Add^M call in a run, so a call of type A2 is made in either line 18 of Algorithm 2, or line 48 of Algorithm 3. Condition (A2.b) describes how the 'new' partial materialisation is updated during addition. Finally, condition (A2.c) says that the output from the previous call is included into $\Delta_{i,j}^+$ for the current call, and the latter is contained in stratum s of the target materialisation $\Pi_\infty[E_i]$.
- A call of type D1 is the first call of a run of our incremental update algorithm, and it is made in line 28 of Algorithm 3. By condition (D1.a), this call follows the final Add^M call from the previous run; condition (D1.b) ensures that this previous call indicated completion of addition; and condition (D1.c) ensures all facts in strata with indices less than or equal to s were updated correctly in the previous run. Condition (D1.d) ensure that arguments $I_{i,1}^o$ and $I_{i,1}^n$ of the first Del^M call in overdeletion reflect the 'old' materialisation. Finally, conditions (D1.e) and (D1.f) capture the fact that updates to the facts from strata with indices less than s are contained in $\Delta_{i,1}^-$ and $\Delta_{i,1}^+$, and line 26 overdeletes the facts in $E^- \cap \mathsf{O}^s$.
- By condition (D2.a), a call of type D2 follows another Del^M call in a run, so a call of type D2 can be made only in line 48 of Algorithm 3. Condition (D2.b) says that $I_{i,j}^o$ is the 'old' materialisation, and condition (D2.c) describes how the 'new' partial materialisation is updated during overdeletion. Finally, condition (D2.d) says that the output from the previous call is included into $\Delta_{i,j}^-$ of the current call, and the latter is contained in stratum s of the 'old' materialisation $\Pi_\infty[E_{i-1}]$.

- Each run contains just one call of type R made in line 37 of Algorithm 3. Condition (R.a) ensures that such a call follows a Del^M call, which, by condition (R.b), must have indicated completion of overdeletion. Condition (R.c) ensures that $I^o_{i,j}$ is the 'old' materialisation, and condition (R.d) describes how the 'new' partial materialisation is updated during overdeletion. Condition (R.e) ensures that the facts in strata with indices less than s have been updated, and all facts in the stratum with index s that no longer hold have been overdeleted. Finally, condition (R.f) describes the argument to Red^M in line 37.
- By condition (A3.a), a call of type A3 may appear in a run only after a call to Red^M; hence, a call of type A3 can be made only in line 42 of Algorithm 3. Condition (A3.b) captures the fact that this call is passed the 'old' materialisation, and conditions (A3.c) and (A3.e) say that $\Delta^-_{i,j}$ and $\Delta^+_{i,j}$ contain all necessary updates to facts in strata with indices less than s. Finally, conditions (A3.d) and (A3.f) ensure that the facts in the stratum with index s belong to the 'new' materialisation.

We are now ready to formalise a key notion of correctness of module functions, which provides the foundation for arguing about the correctness of our algorithms.

Definition 14. 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 stratum index s of λ, each sequence of datasets $E_0, \ldots, E_m$, and each call history H for M compatible with Π, λ, s, and $E_0, \ldots, E_m$, each call $C_{i,j}$ with $0 \le i \le m$ and $1 \le j \le h_i$ in H is correct in the context of E_i, Π, λ, and s.

Thus, to prove correctness of functions of module M, we must consider each program Π, stratification λ of Π, stratum index s of λ, sequence of datasets $E_0, \ldots, E_m$, and call history H that is compatible with Π, λ, s, and $E_0, \ldots, E_m$. Then, we must show that each call $C_{i,j}$ in H satisfies the lower and upper bounds from Table 1. This will usually be done by double induction on the structure of H, where the 'outer' induction ranges over the runs of H, and the 'inner' induction ranges over the calls of one run. All correctness proofs for the module functions we present in Sections 6.2–6.5 follow this pattern. Crucially, such proofs need to consider only calls to functions of module M, and not functions of any other module from a partition of Π.

Theorem 15 shows that, if a program Π is partitioned into modules with correct module functions, then Algorithm 2 correctly computes the initial materialisation, and Algorithm 3 correctly realises incremental updates. The proof is given in Appendix A. This result shows that our framework is indeed modular: we can independently argue about the correctness of each module by considering only the functions of that module, and we can freely combine correct modules without any restrictions.

Theorem 15. *Let Π be a program, let λ be a stratification of Π with maximum stratum index S, let $M^{s,k}$ with $1 \le s \le S$ and $1 \le k \le n_s$ be the module partition of Π w.r.t. λ such that module functions for each $M^{s,k}$ are correct, let $E_0, \ldots, E_m$ be datasets, let I_0 be result of applying Algorithm 2 to Π, λ, and E_0, and, for $1 \le i \le m$, let I_i be the result of successively applying Algorithm 3 to Π, λ, $E^-_i = E_{i-1} \setminus E_i$, and $E^+_i = E_i \setminus E_{i-1}$. Then, $I_i = \Pi_\infty[E_i]$ for each $0 \le i \le m$.*

6. Implementing common modules

In this section, we show how different types of modules can be incorporated into our framework. Towards this goal, in Section 6.1 we first introduce an optimisation of overdeletion used by three of our modules. Then, in Sections 6.2–6.5 we consider modules for transitive closure, symmetric–transitive closure, chain rules, and sequencing totally ordered elements.

6.1. Optimising overdeletion using oracles

Consider a Datalog program is partitioned into modules M_1 and M_2, where module M_1 implements the DRed^c algorithm, and module M_2 realises the symmetric–transitive closure algorithm as outlined in Example 5. To implement DRed^c, module M_1 must track for each fact the number of nonrecursive derivations; but then, module M_2 can safely avoid overdeleting any fact that has at least one nonrecursive derivation via a rule in module M_1. In other words, an optimisation of overdeletion in M_2 may depend on the information maintained and provided by M_1. We capture such exchange of information among modules using the following abstract notion of an *oracle*.

Definition 16. An *oracle* is a Boolean function on all facts. Oracle isTrue is *correct in the context of* a dataset E and program Π if $\mathsf{isTrue}(F) = \mathsf{t}$ implies $F \in \Pi_\infty[E]$ for each fact F.

Intuitively, an oracle isTrue encapsulates an abstract mechanism that a module function can use to check whether a fact holds after an update. An oracle must be sound: $\mathsf{isTrue}(F) = \mathsf{t}$ informs the caller that F is known to hold in the 'new' materialisation. However, an oracle is unlikely to be complete: if $\mathsf{isTrue}(F) = \mathsf{f}$, one should not infer $F \notin \Pi_\infty[E]$; thus, an oracle that returns f on all facts is correct in all contexts. Module functions that we present in Sections 6.2–6.4 all use such

an oracle; moreover, whenever a call to an oracle is made, we implicitly assume that the oracle is correct in the relevant context.

Thus, if the partition of a Datalog program includes a module such as DRed^c or B/F^c that counts nonrecursive derivations, one can provide an oracle that returns t precisely for the facts with at least one nonrecursive derivation. The system we implemented to evaluate our algorithms (see Section 7) follows this approach, and we found this technique to be indispensable in practice. Other ways to construct an oracle are certainly possible, and we shall consider them in future.

6.2. Computing the transitive closure

The ability to axiomatise a binary predicate as transitive is one of the most widely used features of Datalog. For example, it is often used to determine connectivity in a network of objects, represent transitive part-of relations, or represent concept hierarchies. As we argue in Example 2 (see Section 4.2), evaluation of a transitivity rule can be worst-case cubic in the number of objects connected by the predicate. Therefore, optimising the evaluation of the transitivity rule has the potential to significantly improve the performance of reasoning in a range of practical applications. Definition 17 introduces notation for the module that we use in the rest of this paper.

Definition 17. For R a binary predicate, tc(R) is the module containing just the following rule:

$$R(x, y) \land R(y, z) \rightarrow R(x, z). \tag{18}$$

Our solution follows the general idea outlined in Example 4 of distinguishing the 'internal' facts produced by rule (8) from the 'external' facts produced by other rules. A tc(r) module keeps track of the latter in a global set X^R that is initially empty. The way this set is maintained ensures that the transitive closure of X^R coincides with the transitive closure of predicate R in the materialisation. Thus, set X^R can be understood as a 'backbone' from which the transitive closure of R can be recovered. This property is used in our module functions in a way that allows them to consider only instances of rule (18) where the first atom is matched to a fact in the 'backbone' X^R. This can reduce the number of rule instances considered in the same way as in Example 4. Please note, however, that our solution cannot be captured by simple predicate renaming as in Example 4. First, when reasoning incrementally, our technique can use an oracle to avoid overdeleting facts that obviously hold. As we show shortly, this requires managing the set X^R carefully in order to ensure completeness of the approach. Second, the correctness of our approach is not affected if set X^R is at any point replaced by another set X' provided that the transitive closures of X^R and X' coincide. In other words, set X^R can be periodically minimised, which can be needed to maintain the good performance of our technique after successive updates.

Based on these ideas, function Add$^{tc(R)}$, shown in Algorithm 4, essentially implements seminaïve evaluation for rule (18) where the first atom is evaluated in X^R: the loops in lines 52–53 and 54–57 handle the delta rules for the first and second atom of rule (18), respectively. Function Del$^{tc(R)}$, shown in Algorithm 5, is analogous to Add$^{tc(R)}$. The main difference between Add$^{tc(R)}$ and Del$^{tc(R)}$ is that the latter function uses an oracle to avoid deleting facts that obviously hold. This, however, introduces a complication: as Example 22 shows, facts that are not overdeleted due to the oracle must be added to the 'backbone' X_R after overdeletion. To achieve this, all such facts are added to an auxiliary set Y^R (line 73), which is added to X^R during rederivation (line 76) and then cleared. Finally, function Red$^{tc(R)}$, shown in Algorithm 6, identifies for each source vertex u all vertices reachable by the external facts in X^R.

Theorem 18 shows that our solution satisfies the correctness criterion from Definition 14. Moreover, Proposition 19 shows that set X^R can be replaced by another one as suggested earlier without affecting the correctness of the module functions. Both results are proved in Appendix B.

Theorem 18. *Functions* Add$^{tc(R)}$, Del$^{tc(R)}$, *and* Red$^{tc(R)}$ *are correct.*

Proposition 19. *The correctness of the* Add$^{tc(R)}$ *remains unaffected even if set* X^R *is replaced after a call with another set* X' *of R-facts such that* $tc(R)_\infty[X^R] = tc(R)_\infty[X']$.

We next present several examples showing a run of our algorithms that demonstrate several important points. Example 20 introduces the setting and shows how the addition function sets up a 'backbone' X^R.

Example 20. Let Π be the program containing rules (19) and (20), and let λ be the stratification that assigns all rules in Π to just one stratum. Furthermore, let M^1 and M^2 be the module partition of Π such that M^1 and M^2 contain rule (19) and rule (20), respectively. We assume that the module functions for M^1 simply return M_A^1, M_D^1, and M_R^1. Moreover, the only rule in M^1 is nonrecursive, so we also assume that the module maintains derivation counts for the rule and thus provides an oracle isTrue; that is, isTrue(α) = t for a fact α if and only if the number of derivations of α using rule (19) is not zero. Finally, we assume that the module functions for M^2 are implemented using Algorithms 4–6 and they use the oracle isTrue provided by M^1.

Algorithm 4 $\mathrm{Add}^{\mathrm{tc}(R)}[I \vdots \Delta^-, \Delta^+ \vdots \Delta^{\mathrm{m}}]$.

Global variables: X^R

49: $\Delta' := Q := \{R(u, v) \in \Delta^+ \mid R(u, v) \notin \Delta^{\mathrm{m}}\}$
50: $X^R := X^R \cup \Delta'$
51: $J := \emptyset$
52: **for each** $R(u, v) \in \Delta'$ and **each** $R(v, w) \in (I \setminus \Delta^-) \cup \Delta^{\mathrm{m}}$ **do**
53:     **if** $R(u, w) \notin (I \setminus \Delta^-) \cup \Delta^+ \cup J$ **then add** $R(u, w)$ to Q and J
54: **while** $Q \neq \emptyset$ **do**
55:     **remove** an arbitrarily chosen fact $R(v, w)$ from Q
56:     **for each** $R(u, v) \in X^R$ **do**
57:         **if** $R(u, w) \notin (I \setminus \Delta^-) \cup \Delta^+ \cup J$ **then add** $R(u, w)$ to Q and J
58: **return** J

Algorithm 5 $\mathrm{Del}^{\mathrm{tc}(R)}[I^{\mathrm{o}}, I^{\mathrm{n}} \vdots \Delta^-, \Delta^+ \vdots \Delta^{\mathrm{m}}]$.

Global variables: X^R, Y^R

59: $\Delta' := Q := \{R(u, v) \in \Delta^- \mid R(u, v) \notin \Delta^{\mathrm{m}} \cup Y^R\}$
60: $I' := I^{\mathrm{n}} \setminus (\Delta^- \cup Y^R)$
61: $J := \emptyset$
62: **for each** $R(u, v) \in \Delta'$ and **each** $R(v, w) \in I'$ **do**
63:     **if** $R(u, w) \in I^{\mathrm{n}} \setminus (\Delta^- \cup J \cup Y^R)$ **then**
64:         **add** $R(u, w)$ to Q
65:         **if** $\mathrm{isTrue}(R(u, w)) = \mathrm{f}$ **then add** $R(u, w)$ to J
66:         **else add** $R(u, w)$ to Y^R
67: **while** $Q \neq \emptyset$ **do**
68:     **remove** an arbitrarily chosen fact $R(v, w)$ from Q
69:     **for each** $R(u, v) \in X^R$ **do**
70:         **if** $R(u, w) \in I^{\mathrm{n}} \setminus (\Delta^- \cup J \cup Y^R)$ **then**
71:             **add** $R(u, w)$ to Q
72:             **if** $\mathrm{isTrue}(R(u, w)) = \mathrm{f}$ **then add** $R(u, w)$ to J
73:             **else add** $R(u, w)$ to Y^R
74: $X^R := X^R \setminus \Delta^-$
75: **return** J

Algorithm 6 $\mathrm{Red}^{\mathrm{tc}(R)}[I^{\mathrm{o}}, I^{\mathrm{n}} \vdots \Delta]$.

Global variables: X^R, Y^R

76: $X^R := X^R \cup Y^R$
77: $J := Y^R := \emptyset$
78: **for each** u such that there exist v with $R(u, v) \in \Delta$ **do**
79:     **for each** w reachable from u via R-facts in X^R **do**
80:         **if** $R(u, w) \in \Delta$ **then add** $R(u, w)$ to J
81: **return** J

$$S(x, y) \rightarrow R(x, y) \tag{19}$$

$$R(x, y) \land R(y, z) \rightarrow R(x, z) \tag{20}$$

Let E be as specified in equation (21), and consider applying Algorithm 2 to Π, λ, and E. This process derives facts shown in equation (22), where $I_1 = \Pi_\infty[E]$; moreover, the global set X_1^R of module M^2 contains facts shown in equation (23) after materialisation. We now discuss several important aspects of our algorithms.

$$E = \{S(b, c), R(c, d), R(d, e)\} \tag{21}$$

$$I_1 = \{R(b, c), R(b, d), R(b, e), R(c, e)\} \cup E \tag{22}$$

$$X_1^R = \{R(b, c), R(c, d), R(d, e)\} \tag{23}$$

First, note that the transitive closure of X_1^R is equal to I_1, but X_1^R is smaller than I_1. In other words, X_1^R is the 'backbone' of the transitive closure contained in I_1, and it consists of the 'external' facts (i.e., facts that are either explicitly given in the input or produced by other modules). Argument Δ^{m} is critical to our ability to distinguish the 'external' facts from the facts produced by the module itself. For example, module M_2 derives $R(c, e)$ in the first round of rule application so, in the second round, this fact is passed back to M_2 as part of Δ^+. Without any additional information, module M_2 would need to consider fact $R(c, e)$ as being 'external' and add it to X_1^R. To remedy this, our framework informs each module about the facts that the module derived in the preceding round using the Δ^{m} argument. Thus, $R(c, e) \in \Delta^{\mathrm{m}}$ holds on the second call to the addition function, so lines 49 and 50 ensure that fact $R(c, e)$ is not added to X^R. Analogous reasoning applies to the rest of $I_1 \setminus X_1^R$.

Second, by matching the first atom of rule (20) to X_1^R, Algorithm 4 skips certain instances of the rule; for example, the algorithm never considers instance $R(b,d) \wedge R(d,e) \rightarrow R(b,e)$. This, however, does not affect the correctness of materialisation because the transitive closure of X_1^R provides the required materialisation. This can boost performance as discussed in Example 4.

The following example shows how the 'backbone' is maintained in the presence of fact addition.

Example 21. Continuing Example 20, let E^+ be as specified in equation (24), and consider applying Algorithm 3 to Π, λ, $\emptyset$, and E^+. This process derives facts shown in equation (25), were $I_2 = \Pi_\infty[E \cup E^+]$, and it updates the global set of module M_1 to X_2^R as shown in equation (26). In particular, module M^1 uses E^+ to derive $R(a,c)$; fact $R(c,e)$ is not derived by M^1 because it is already contained in I_1. Fact $R(a,c)$ is then passed to module M^2, which incorporates it into the new 'backbone' X_2^R—that is, M^2 maintains the 'backbone' using the facts derived by other modules. Note that facts such as $R(a,d)$ and $R(a,e)$ are not included into X_2^R because they can be recovered by transitively closing X_2^R. Finally, note that facts $R(a,c)$, $R(b,c)$, and $R(c,e)$ all have one recursive derivation after the update, so the oracle isTrue returns t on all those facts.

$$E^+ = \{S(a,c), S(c,e)\} \tag{24}$$

$$I_2 = \{R(a,c), R(a,d), R(a,e)\} \cup I_1 \cup E^+ \tag{25}$$

$$X_2^R = \{R(a,c), R(b,c), R(c,d), R(d,e)\} \tag{26}$$

Finally, the following example demonstrates several difficulties that need to be handled in overdeletion. These arise largely due to an interaction between the oracle and the 'backbone'.

Example 22. Continuing Example 21, let E^- be as specified in equation (27), and consider applying Algorithm 3 to Π, λ, E^-, and $\emptyset$. Facts $S(b,c)$ and $S(c,e)$ are not affected by the update, so isTrue$(R(b,c)) =$ isTrue$(R(c,e)) =$ t holds both before and after the update. In contrast, the first call to Del$^{M^1}$ reduces the number of nonrecursive derivations of $R(a,c)$ to zero, which results in isTrue$(R(a,c)) =$ f. Since M^1 is the first module considered in stratum with index 1, the oracle is updated before it is used by module M^2, which ensures correctness. Deletion exhibits several intricacies, so we next discuss the process in detail.

$$E^- = \{R(d,e), S(a,c)\} \tag{27}$$

Consider now the first call of Algorithm 5 during overdeletion. This call is made with argument $\Delta^- = \{R(d,e), S(a,c)\}$: even though module M_1 identifies that fact $R(a,c)$ needs to be overdeleted, this fact will be passed to module M_2 only in the second call. Moreover, $(M_2)_D[I^o, I^n : \Delta^-, \emptyset]$ contains fact $R(a,e)$: this fact is derived by the instance $R(a,d) \wedge R(d,e) \rightarrow R(a,e)$ of rule (20) where $R(d,e) \in \Delta^-$. Finally, fact $R(a,e)$ does not hold in the 'new' materialisation because $R(a,c)$ will be deleted eventually. Therefore, Algorithm 5 must return $R(a,e)$ to satisfy the lower bound.

Now consider the execution of Algorithm 5. Fact $R(d,e)$ is added to the set Q in line 59; as a result, fact $R(c,e)$ is considered in line 70 and subsequently in line 72. Since isTrue$(R(c,e)) =$ t holds, the fact is known to hold after the update, so Algorithm 5 does not overdelete $R(c,e)$. One might expect that we do not need to consider further consequences of $R(c,e)$; but then, the algorithm would never examine and overdelete $R(a,e)$ (as is needed for the lower bound). Intuitively, this problem arises because our algorithm never considers the rule instance from the previous paragraph. Instead, the algorithm matches the first body atom in X^R and only considers instance $R(a,c) \wedge R(c,e) \rightarrow R(a,e)$; now the second atom of this rule instance is true after the update, and the first atom will be deleted eventually, but the algorithm is not yet informed of this.

To remedy this, our algorithm considers the consequences of $R(c,e)$ even though the fact is not overdeleted. This eventually ensures that the algorithm considers and overdeletes $R(a,e)$. In a similar vein, fact $R(b,e)$ depends on $R(d,e)$, and there is no evidence that the fact holds after the update; hence, fact $R(b,e)$ is overdeleted too. In the second iteration of overdeletion, fact $S(a,c)$ is passed to function Del$^{M^1}$, which overdeletes $R(a,c)$. After overdeletion, sets X^R and Y^R are as follows.

$$X_3^R = \{R(b,c), R(c,d)\} \tag{28}$$

$$Y^R = \{R(c,e)\} \tag{29}$$

In the rederivation phase, fact $R(b,e)$ must be rederived. However, overdeletion changed the 'backbone' so that $R(b,e)$ does not follow from the transitive closure of X_3^R, so Algorithm 6 must repair the 'backbone' in a way that restores this property. This is the role of set Y^R: this set records the facts encountered during overdeletion that are known to hold after the update, so that they can be added back to X^R in line 76. In our example, this allows Algorithm 6 to correctly rederive $R(b,e)$.

6.3. Computing the symmetric–transitive closure

While Section 6.2 deals with reachability in directed relations, applications often need to deal with undirected relations as well. Reachability in undirected relations can be axiomatised as follows.

Definition 23. For R a binary predicate, stc(R) is the module containing the following two rules:

$$R(x, y) \rightarrow R(y, x) \tag{30}$$

$$R(x, y) \wedge R(y, z) \rightarrow R(x, z). \tag{31}$$

Rules of the form (30)–(31) abound in practice. For example, many ontologies we discuss in Section 7 contain symmetric and transitive predicates: the FHKB ontology of genealogical relations contains predicates such as 'is sibling of', and the Relations life sciences ontology contains predicates such as 'shares ancestor with' and 'is homologous to'. Thus, optimising reasoning with stc(R) has the potential to improve reasoning performance in many practical scenarios.

Our approach to dealing with module stc(R) has already been suggested in Example 5. In particular, the rules of stc(R) ensure that constants occurring in facts with predicate R are partitioned into connected components—that is, sets of mutually reachable constants. Once we have identified connected components, we can simply connect all pairs of constants in each component. Our main challenge is to maintain the list of connected components incrementally, rather than recompute it after each update.

Module functions for stc(R), shown in Algorithms 7–9, show how to put these ideas into practice. They depend on two global variables. Variable X^R contains all connected components—that is, each member of X^R is a set of constants that are mutually connected in the current materialisation. This set is maintained incrementally during additions, deletions, and rederivation. Global variable Y^R is used during deletion and rederivation only, and it keeps track of facts of the form $R(u', v')$ whose overdeletion was attempted, but which were identified by the oracle as holding after the update. Both sets are initially empty.

Addition and rederivation both use an auxiliary function CLOSEEDGES(Δ), which computes the effects of adding the facts in Δ to the connected components represented by the current value of X^R. The function updates X^R to reflect the connected components after Δ has been added, and it returns the facts that should be added to the materialisation due to adding Δ. To this end, the function considers in lines 85–90 each fact $R(u, v) \in \Delta$. If constant u does not occur in a connected component in X^R, then X^R is updated by creating a new component containing just u, and $R(u, u)$ is added to the function's result (line 86). Constant v is processed analogously (line 87). Finally, if u and v occur in distinct connected components in X^R, then the components are merged (line 89), and facts connecting pairs of constants from the two components are added to the function's result (line 90).

Function Add$^{stc(R)}$, shown in Algorithm 7, simply calls this function on $\Delta^+ \setminus \Delta^m$ (i.e., the facts produced by other modules in the last round of rule application), and it returns the facts that are not already contained in the materialisation. Function Del$^{stc(R)}$, shown in Algorithm 8, processes each fact $R(u, v)$ in $\Delta^- \setminus \Delta^m$ (i.e., each fact overdeleted by other modules in the last round of rule application) such that u and v belong to the same connected component U (lines 93–97), it overdeletes each fact $R(u', v')$ with $\{u', v'\} \subseteq U$ that is not known to hold after the update (line 95), and it removes U from X^R (line 97). Each fact $R(u', v')$ that is known to hold after the update is added to Y^R in order to facilitate rederivation (line 96). Finally, function Red$^{stc(R)}$, shown in Algorithm 9, simply closes the set Y^R as during addition (line 99) and empties it (line 100).

It is straightforward to see that the worst-case running times of our functions are determined by the loops in line 90 and 94–96, which require $O(n^2)$ time where n is the maximum size of a connected component. In contrast, evaluating rules (30) and (31) using seminaïve evaluation requires $O(n^3)$ time in the worst case. This improvement in worst-case complexity is directly due to the fact that our optimised algorithms do not need to consider all instances of rule (31). Theorem 24 shows that our solution satisfies the correctness criterion from Definition 14, and it is proved in Appendix C.

Theorem 24. *Functions* Add$^{stc(R)}$, Del$^{stc(R)}$, *and* Red$^{stc(R)}$ *are correct.*

6.4. Reasoning with regular chain rules

In this section, we generalise our technique from Section 6.2 to a class of programs that can express very general connectivity properties. In particular, we shall consider a specific subclass of *chain programs* [2], whose structure is captured by the following definition.

Definition 25. A *chain program* is a Datalog program that contains only rules of the form

$$S_1(x_0, x_1) \wedge \cdots \wedge S_n(x_{n-1}, x_n) \rightarrow R(x_0, x_n). \tag{32}$$

We consider a restriction of chain programs where the consequences of the program with a specific predicate can be described using a regular language. Such programs are commonly called *regular chain programs*, and they have been studied extensively in numerous contexts. For example, Horrocks and Sattler [41] showed that regular chain programs can be

---

Algorithm 7 $\text{Add}^{\text{stc}(R)}[I \mathop{:} \Delta^-, \Delta^+ \mathop{:} \Delta^{\mathsf{m}}]$.

Global variables: X^R

82: **return** $\text{CloseEdges}(\Delta^+ \setminus \Delta^{\mathsf{m}}) \setminus ((I \setminus \Delta^-) \cup \Delta^+)$

83: **function** $\text{CloseEdges}(\Delta)$
84:    $J := \emptyset$
85:    **for each** $R(u, v) \in \Delta$ **do**
86:       **if** no $U \in X^R$ exists such that $u \in U$ **then add** $\{u\}$ to X^R, and $R(u, u)$ to J
87:       **if** no $V \in X^R$ exists such that $v \in V$ **then add** $\{v\}$ to X^R, and $R(v, v)$ to J
88:       **if** u and v belong to distinct $U \in X^R$ and $V \in X^R$, respectively **then**
89:          **remove** U and V from X^R, and **add** $U \cup V$ to X^R
90:          **for each** $u' \in U$ and **each** $v' \in V$ **do add** $R(u', v')$ and $R(v', u')$ to J
91:    **return** J

Algorithm 8 $\text{Del}^{\text{stc}(R)}[I^{\mathsf{o}}, I^{\mathsf{n}} \mathop{:} \Delta^-, \Delta^+ \mathop{:} \Delta^{\mathsf{m}}]$.

Global variables: X^R, Y^R

92: $J := \emptyset$
93: **for each** $U \in X^R$ such that there exists $R(u, v) \in \Delta^- \setminus \Delta^{\mathsf{m}}$ with $\{u, v\} \subseteq U$ **do**
94:     **for each** $u' \in U$ and **each** $v' \in U$ **do**
95:        **if** $\text{isTrue}(R(u', v')) = \mathsf{f}$ **then add** $R(u', v')$ to J
96:        **else add** $R(u', v')$ to Y^R
97:     **remove** U from X^R
98: **return** $J \cap (I^{\mathsf{n}} \setminus \Delta^-)$

Algorithm 9 $\text{Red}^{\text{stc}(R)}[I^{\mathsf{o}}, I^{\mathsf{n}} \mathop{:} \Delta]$.

Global variables: Y^R

 99: $J := \text{CloseEdges}(Y^R) \cap \Delta$
100: $Y^R := \emptyset$
101: **return** J

---

combined with description logics in a way that preserves decidability of reasoning. This work provided the foundation for *complex property inclusions* in the Web Ontology Language (OWL) [60], which are extensively used in practical applications. Consequently, numerous applications can be expected to benefit from optimisations of reasoning with regular chain programs.

To define regular chain programs formally, we first simplify the notions of rule unfolding presented in Section 3 to chain programs. Let Σ_b be the set of all binary predicates in the signature. Note that the body of each rule of the form (32) naturally corresponds to the finite word $S_1, \dots, S_n \in \Sigma_b^*$ of predicates from the rule body: for each word, we just need to 'plug in' the relevant variables. Moreover, unfolding of rules of the form (32) produces rules of the same form. This motivates the following definition, which allows us to represent an unfolding of a binary predicate by a possibly infinite set of finite words over Σ_b.

Definition 26. Given a chain program Π, relation $\rightsquigarrow_\Pi \subseteq \Sigma_b \times \Sigma_b$ is the smallest relation on finite words over Σ_b such that $P_1, \dots, P_m \rightsquigarrow_\Pi P_1, \dots, P_{i-1}, S_1, \dots, S_n, P_{i+1}, \dots, P_m$ holds for all binary predicates $P_1, \dots, P_n$ and each rule in Π of the form (32) where $R = P_i$. Let $\rightsquigarrow_\Pi^*$ be the reflexive–transitive closure of $\rightsquigarrow_\Pi$. Then, a word $P_1, \dots, P_n$ is an *unfolding* of a binary predicate R w.r.t. Π if $R \rightsquigarrow_\Pi^* P_1, \dots, P_n$.

Since each rule of the form (32) contains at least one body atom, each unfolding of a predicate is nonempty. Proposition 27 reformulates the characterisation of predicate unfolding from Section 3: each derivation of a binary fact in a chain program corresponds to a path in the set of explicitly given facts matching one unfolding of the fact's predicate.

Proposition 27. *For Π a chain program, R a binary predicate, E a set of explicitly given facts, and a and b constants, $R(a, b) \in \Pi_\infty[E]$ if and only if there exist an unfolding $P_1, \dots, P_n$ of R w.r.t. Π and constants $c_0, c_1, \dots, c_n$ such that $c_0 = a$, $c_n = b$, and $\{P_1(c_0, c_1), \dots, P_n(c_{n-1}, c_n)\} \subseteq E$.*

The set of unfoldings of each binary predicate in a chain program can thus be viewed as a language over an alphabet Σ_b, which motivates the following definition.

Definition 28. A chain program Π is *regular* if, for each binary predicate R, the language of all unfoldings of R w.r.t. Π is regular.

We write Π_{rch} to stress that Π is a regular chain program. For each binary predicate R from the head of a rule in Π_{rch}, we fix a nondeterministic finite automaton (NFA) $N^R = \langle Q^R, \Sigma_b, \delta^R, q_s^R, F^R \rangle$ that accepts precisely all unfoldings of R w.r.t.

Fig. 1. NFAs for predicates R (left) and S (right) from Example 29.

Π; here, Q^R is the finite set of states, $\delta^R : Q^R \times \Sigma_b \to 2^{Q^R}$ is the transition function, q_s^R is the start state, and $F^R \subseteq Q^R$ is the set of final states. We assume that the states of each NFA are private to the NFA—that is, $Q^{R_1} \cap Q^{R_2} = \emptyset$ for all distinct R_1 and R_2. We also assume that each NFA is free of ε-transitions; this is without loss of generality as such transitions can always be eliminated. Finally, since unfoldings are not empty, $s_s^R \notin F^R$ holds—that is, the start state is never also a final state.

While chain programs can be easily recognised syntactically, determining whether a chain program is regular involves checking a semantic property. Note that the rules of a chain program straightforwardly correspond to productions of a *pure context-free grammar* (i.e., a context-free grammar that does not distinguish between terminal and nonterminal symbols). Maurer et al. [57] conjectured that checking whether a pure context-free grammar generates a regular language is undecidable, and we are unaware of a result that settles this question. Several sufficient and practically verifiable restrictions of chain programs have been developed [41,48], and all of them provide ways to construct the relevant NFAs. Moreover, the restrictions by Horrocks and Sattler [41] have been incorporated into the OWL 2 DL ontology language, and they are widely used in practice; thus, the results from this section are applicable to the chain programs obtained by translating OWL 2 DL ontologies.

We next outline a well-known technique for reasoning with regular chain rules by NFA traversal, instead of seminaïve evaluation. Variants of this technique have been considered in numerous different contexts, such as description logic reasoning [41] and incremental Datalog evaluation [24]. Our approach manipulates *q-facts*—expressions of the form $q(u, v)$ where u and v are constants and q is an NFA state; a finite set of q-facts is called a *q-dataset*. Now let E be a set of explicitly given facts, and let N^P be an NFA that accepts precisely all unfoldings of a predicate P w.r.t. a regular chain program Π. We can compute all facts with the P predicate in $\Pi_\infty[E]$ using Proposition 27; the so-called *product construction* [9] describes this from a conceptual point of view, but we next outline a procedure that is more amenable to practical implementation. First, we initialise a q-dataset with a q-fact of the form $q_s^P(a, a)$ for each constant a in E, where q_s^P is the start state of N^P. Next, we exhaustively apply the following step: for each $q(a, b)$ that has already been derived, each $R(b, c) \in E$, and each R-transition from q to q' in N^P, we derive $q'(a, c)$. Once this process reaches the least fixpoint, for each $q(a, b)$ that has been derived where q is a final state of N^P, we derive $P(a, b)$. Example 29 shows that, in a way similar to Section 6.2, such a technique can reduce the complexity of rule matching.

Example 29. Let Π contain rules (33) and (34). For E as specified in equation (35), the materialisation I of Π w.r.t. E is shown in equation (36).

$$R(x, y) \wedge R(y, z) \to R(x, z) \tag{33}$$

$$S(x, y) \wedge R(y, z) \to S(x, z) \tag{34}$$

$$E = \{R(c_i, c_{i+1}) \mid 1 \le i < n\} \cup \{S(d_k, c_1) \mid 1 \le k \le m\} \tag{35}$$

$$I = \Pi_\infty[E] = \{R(c_i, c_j) \mid 1 \le i < j \le n\} \cup \{S(d_k, c_j) \mid 1 \le k \le m, 1 \le j \le n\} \tag{36}$$

Note that $O(n^3)$ rule instances of the form (37) and $O(mn^2)$ rule instances of the form (38) are applicable to I; thus, seminaïve evaluation requires $O(n^3 + mn^2)$ steps.

$$R(c_i, c_j) \wedge R(c_j, c_\ell) \to R(c_i, c_\ell) \qquad \text{with } 1 \le i < j < \ell \le n \tag{37}$$

$$S(d_k, c_i) \wedge R(c_i, c_j) \to S(d_k, c_j) \qquad \text{with } 1 \le k \le m \text{ and } 1 \le i < j \le n \tag{38}$$

To apply the automata-based approach, first note that NFAs N^R and N^S from Fig. 1 accept precisely the unfoldings of R and S, respectively. To apply N^R to E, we derive $q_0(d_k, d_k)$ for $1 \le k \le m$ and $q_0(c_i, c_i)$ for $1 \le i \le n$; then, using $q_0(c_i, c_i)$ and $R(c_i, c_{i+1})$ we derive $q_1(c_i, c_{i+1})$ for $1 \le i < n$; finally, using $q_1(c_i, c_j)$ and $R(c_j, c_{j+1})$ we derive $q_1(c_i, c_{j+1})$ for $1 \le i < j \le n$. This process clearly takes $O(n^2)$ steps. To apply N^S to E, we derive $q_2(d_k, d_k)$ for $1 \le k \le m$ and $q_2(c_i, c_i)$ for $1 \le i \le n$; then, using $q_2(d_k, d_k)$ and $S(d_k, c_1)$ we derive $q_3(d_k, c_1)$ for $1 \le k \le m$; finally, using $q_3(d_k, c_i)$ and $R(c_i, c_{i+1})$ we derive $q_3(d_k, c_{i+1})$ for $1 \le i < n$. This process clearly takes $O(mn)$ steps. Hence, the complexity of reasoning with Π is quadratic, rather than cubic. As in Section 6.2, this improvement arises because NFA traversal considers only the facts in E (i.e., the 'backbone'), and not the facts derived by the traversal itself. The algorithms presented in this section generalise the ones from Section 6.2; however, the algorithms for transitivity are nevertheless useful in practice because they do not incur the overhead of maintaining q-facts.

Note that the complexity improvements disappear if we consider the two rules in isolation. For example, we can apply rule (33) in $O(n^2)$ steps using either the NFA approach or as in Section 6.2. However, if we then apply rule (33) to all consequences of rule (33), both the seminaïve evaluation and the automata-based approach require $O(mn^2)$ steps. Intuitively, an NFA for a predicate encodes all consequences of all rules, so the rules in a regular chain program do not need to exchange information; thus, each NFA can be matched against the same 'backbone' and thus benefit from performance improvements.

Another benefit of our approach is that NFAs can be converted into DFAs and then minimised. Although the first of these steps is worst-case exponential, an exponential blowup rarely occurs in practice in our experience; moreover, NFAs are typically quite small so, even if an exponential blowup occurs, the resulting DFAs are still of manageable sizes. Finally, DFA minimisation can be seen as elimination of redundancy from a program, which is beneficial for performance.

Based on these ideas, Algorithms 10–12 implement functions $\mathsf{Add}^{\Pi_\mathsf{rch}}$, $\mathsf{Del}^{\Pi_\mathsf{rch}}$, and $\mathsf{Red}^{\Pi_\mathsf{rch}}$ for a module Π_rch consisting of regular chain rules. The algorithms maintain several global variables, all of which are initially empty. In particular, sets X and Y play the same role as sets X^R and Y^R in Section 6.2: the former contains the 'backbone' facts produced by other modules, and the latter collects facts that are not overdeleted due to the oracle. Moreover, our algorithms use q-facts to represent partial matches of NFAs, but these cannot be stored in the materialisation or passed in sets Δ^+ and Δ^-. Therefore, a global q-dataset I^q collects the q-facts derived thus far and thus 'shadows' the materialisation, and another global q-dataset Δ^q collects all q-facts considered during deletion. Since NFAs do not share states, we need just one I^q and one Δ^q. Finally, for each predicate R_i occurring in the head of a rule in Π_rch, our algorithms use a global set of constants Z^{R_i} whose role will be clarified shortly.

The addition and the rederivation functions use an auxiliary function $\textsc{AddEdges}(\Delta, Q, B)$, whose task is to return the q-facts that should be added to I^q as a result of extending the materialisation with 'ordinary' facts in Δ. Set B contains 'blocked' q-facts—that is, facts that have already been derived and should not be returned. Finally, set Q contains additional q-facts that should be considered. The function is similar to function $\mathsf{Add}^{\mathsf{tc}(R)}$ from Section 6.2, and the main difference is that it needs to maintain both q-facts and 'ordinary' facts. The function keeps track of the q-facts to be processed in set Q, and it collects the resulting q-facts in an auxiliary set J^q. Processing is split into two parts. In lines 107–112, each fact $R(v, w)$ produced by other modules (and thus to be added to the materialisation) is considered and added to the 'backbone' X (line 108). Next, each R-transition from state q_1 to state q_2 in each NFA N^S is considered (line 109). If q_1 is the start state q_s^S of N^S, $q_2(v, w)$ is derived to record the move from the start state (line 110). Thus, unlike the approach outlined in Example 29, $q_s^S(v, v)$ is not derived; however, this is not a problem because q_s^S cannot be a final state of N^S and so it does not need to be considered in line 104 or 142. Moreover, each q-state $q_1(u, v)$ in I^q shows that there is a word connecting u and v that moves the NFA into state q_1, and fact $R(v, w)$ allows this word to be extended and move the NFA to q_2, so $q_2(u, w)$ is derived (lines 111–112). After processing all facts in Δ, the set of q-facts is closed to a fixpoint (lines 113–116) by joining each q-fact $q_1(u, v)$ from Q (line 113) with each fact $R(v, w)$ from the 'backbone' that corresponds to a transition of some NFA N^S from q_1 (line 115).

Function $\mathsf{Add}^{\Pi_\mathsf{rch}}$ then simply calls $\textsc{AddEdges}$ on the set of facts $\Delta^- \setminus \Delta^m$—that is, facts produced by other modules in the preceding round of rule applications. Argument I^q ensures that the function derives q-facts only for transitions not considered previously. Once set J^q of newly derived q-facts is produced, it is added to I^q (line 103), and each q-fact corresponding to a finishing state of an NFA is converted into an ordinary fact (line 104).

Function $\mathsf{Del}^{\Pi_\mathsf{rch}}$ computes in lines 119–128 the q-facts that should be deleted from I^q as a result of deleting the facts in Δ^-. This process is analogous to $\textsc{AddEdges}$: lines 119–123 compute the direct consequences of Δ^-, and lines 124–128 compute the fixpoint. Set Δ^q keeps track of all q-facts encountered during overdeletion so we can overdelete each q-fact just once. To convert J^q to ordinary facts, each overdeleted q-fact $q(u, v) \in J^q$ is considered (lines 129–134), and the NFA N^R that q belongs to is identified (line 130). Note that $q(u, v)$ may need to be rederived, but neither u nor v will necessarily occur in the arguments passed to the rederivation functions; thus, to keep track of constants that should be revisited during rederivation, constant u is added to the global set Z^R (line 131). Note that Z^R is associated with NFA N^R—that is, a separate set is maintained for each NFA. Moreover, if q is a final state of N^R, then $q(u, v)$ corresponds to the ordinary fact $R(u, v)$. As in Section 6.2, an oracle is consulted to see whether overdeleting $R(u, v)$ can be avoided; if so, $R(u, v)$ is added to the global set Y for the same reasons as in Section 6.2. Finally, I^q and Δ^q are updated and Δ^- is removed from the backbone (line 135).

Function $\mathsf{Red}^{\Pi_\mathsf{rch}}$ deals with rederivation. As mentioned already, argument Δ provides no information about which transitions have been overdeleted. Therefore, each constant $u \in Z^S$ from the set associated with NFA N^S is considered; if the NFA contains a transition from u that can be matched to a fact $R(u, v) \in X$ in the 'backbone', $q(u, v)$ is added to the set Q (lines 138–139). Function $\textsc{AddEdges}(Y, Q, \emptyset)$ then computes the q-facts needed to close set Q and also adds Y into the 'backbone'; the need for the latter is the same as in Section 6.2. Finally, the q-facts are converted into ordinary facts (line 142) as in addition.

Theorem 30 shows that our solution satisfies the correctness criterion from Definition 14, and it is proved in Appendix D.

Theorem 30. *Functions* $\mathsf{Add}^{\Pi_\mathsf{rch}}$, $\mathsf{Del}^{\Pi_\mathsf{rch}}$, *and* $\mathsf{Red}^{\Pi_\mathsf{rch}}$ *are correct.*

Algorithm 10 $\mathsf{Add}^{\Pi_{\mathsf{rch}}}[I : \Delta^-, \Delta^+ : \Delta^\mathsf{m}]$.

Global variables: I^q, X

102: $J^q := \textsc{AddEdges}(\Delta^+ \setminus \Delta^\mathsf{m}, \emptyset, I^q)$
103: $I^q := I^q \cup J^q$
104: **return** $\{R(u, v) \mid \exists q \in F^R \text{ such that } q(u, v) \in J^q \text{ and } R(u, v) \notin (I \setminus \Delta^-) \cup \Delta^+\}$

105: **function** $\textsc{AddEdges}(\Delta, Q, B)$
106:     $J^q := Q$
107:     **for each** $R(v, w) \in \Delta$ such that R appears in Π_{rch} **do**
108:         $X := X \cup \{R(v, w)\}$
109:         **for each** predicate S occurring in the head of a rule in Π_{rch} and **all** $q_1, q_2 \in Q^S$ such that $q_2 \in \delta^S(q_1, R)$ **do**
110:             **if** $q_1 = q_s^S$ and $q_2(v, w) \notin B \cup J^q$ **then add** $q_2(v, w)$ to Q and J^q
111:             **for each** $q_1(u, v) \in I^q$ **do**
112:                 **if** $q_2(u, w) \notin B \cup J^q$ **then add** $q_2(u, w)$ to Q and J^q
113:     **while** $Q \neq \emptyset$ **do**
114:         **remove** an arbitrarily chosen fact $q_1(u, v)$ from Q
115:         **for each** predicate S occurring in the head of a rule in Π_{rch}, **each** $R(v, w) \in X$, and **each** $q_2 \in \delta^S(q_1, R)$ **do**
116:             **if** $q_2(u, w) \notin B \cup J^q$ **then add** $q_2(u, w)$ to Q and J^q
117:     **return** J^q

Algorithm 11 $\mathsf{Del}^{\Pi_{\mathsf{rch}}}[I^\mathsf{o}, I^\mathsf{n} : \Delta^-, \Delta^+ : \Delta^\mathsf{m}]$.

Global variables: I^q, Δ^q, X, Y, $Z^{R_1}, \ldots, Z^{R_n}$

118: $Q := J := J^q := \emptyset$
119: **for each** $R(v, w) \in \Delta^-$ such that R appears in Π_{rch} **do**
120:     **for each** predicate S occurring in the head of a rule in Π_{rch} and **all** $q_1, q_2 \in Q^S$ such that $q_2 \in \delta^S(q_1, R)$ **do**
121:         **if** $q_1 = q_s^S$ and $q_2(v, w) \notin \Delta^q \cup J^q$ **then add** $q_2(v, w)$ to Q and J^q
122:         **for each** $q_1(u, v) \in I^q$ **do**
123:             **if** $q_2(u, w) \notin \Delta^q \cup J^q$ **then add** $q_2(u, w)$ to Q and J^q
124: **while** $Q \neq \emptyset$ **do**
125:     **remove** an arbitrarily chosen fact $q_1(u, v)$ from Q
126:     Let S be the predicate occurring in the head of a rule in Π_{rch} such that $q_1 \in Q^S$
127:     **for each** $R(v, w) \in X$ and **each** $q_2 \in \delta^S(q_1, R)$ **do**
128:         **if** $q_2(u, w) \notin \Delta^q \cup J^q$ **then add** $q_2(u, w)$ to Q and J^q
129: **for each** $q(u, v) \in J^q$ **do**
130:     Let R be the predicate occurring in the head of a rule in Π_{rch} such that $q \in Q^R$
131:     $Z^R := Z^R \cup \{u\}$
132:     **if** $q \in F^R$ and $R(u, v) \in I^\mathsf{n} \setminus (\Delta^- \cup J \cup Y)$ **then**
133:         **if** isTrue$(R(u, v)) = \mathsf{t}$ **then** $Y := Y \cup \{R(u, v)\}$
134:         **else** $J := J \cup \{R(u, v)\}$
135: $I^q := I^q \setminus J^q$, $\Delta^q := \Delta^q \cup J^q$, $X := X \setminus \Delta^-$
136: **return** J

Algorithm 12 $\mathsf{Red}^{\Pi_{\mathsf{rch}}}[I^\mathsf{o}, I^\mathsf{n} : \Delta]$.

Global variables: I^q, Δ^q, X, Y, $Z^{R_1}, \ldots, Z^{R_n}$

137: $Q := \emptyset$
138: **for each** predicate S occurring in the head of a rule in Π_{rch}, **each** $u \in Z^S$, **each** $R(u, v) \in X$, and **each** $q \in \delta^S(q_s^S, R)$ **do**
139:     **if** $q(u, v) \notin Q$ **then add** $q(u, v)$ to Q
140: $J^q := \textsc{AddEdges}(Y, Q, \emptyset)$
141: $I^q := I^q \cup J^q$, $\Delta^q := Y := Z^{R_1} := \cdots := Z^{R_n} := \emptyset$
142: **return** $\{S(u, v) \in \Delta \mid \exists q \in F^S \text{ such that } q(u, v) \in I^q\}$

Algorithms 10–12 correspond closely to Algorithms 4–6 from Section 6.2, but there is a notable difference: $\mathsf{Del}^{\mathsf{tc}(R)}$ skips facts in $\Delta^\mathsf{m} \cup Y^R$ in line 59, whereas $\mathsf{Del}^{\Pi_{\mathsf{rch}}}$ does not do the same in line 119. The following example illustrates the rationale for this.

Example 31. Consider a program Π partitioned into two modules assigned to one stratum: M^1 contains rules (39)–(41) and M^2 contains rules (42)–(43). Moreover, we assume that the functions for M^1 return $M_\infty^1[I] \setminus ((I \setminus \Delta^-) \cup \Delta^+)$, M_D^1, and M_R^1, and that M^2 is handled by Algorithms 10–12 that use NFAs N^R and N^T from Fig. 2 and an oracle that always returns f.

$$U(x, y) \rightarrow S(x, y) \tag{39}$$

$$S(x, y) \rightarrow T(x, y) \tag{40}$$

$$U(x, y) \rightarrow W(x, y) \tag{41}$$

$$W(x, y) \rightarrow T(x, y) \tag{42}$$

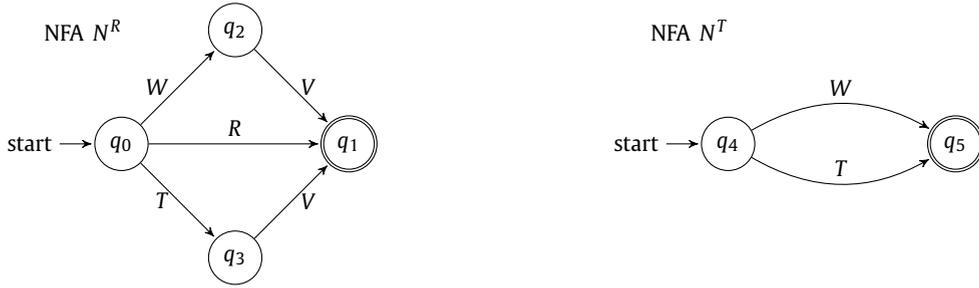$$T(x, y) \wedge V(y, z) \rightarrow R(x, z) \tag{43}$$

NFA N^R

NFA N^T

Fig. 2. NFAs for predicates R (left) and T (right) from Example 31.

We apply Algorithm 2 to Π and E from equation (44). Since Add^{M_1} essentially returns the upper bound, $S(a,b)$, $T(a,b)$, and $W(a,b)$ are all derived in line 12. In the same line, Add^{M^2} adds $V(b,c)$ to X, but keeps I^q empty. Then, in line 18, Add^{M^2} adds $T(a,b)$ and $W(a,b)$ to X, updates I^q, and derives $R(a,c)$. Materialisation thus derives the facts shown in equation (45) where $I_1 = \Pi_\infty[E]$, and the global sets X and I^q of module M^2 are shown in equations (46) and (47).

$$E = \{U(a,b), V(b,c)\} \tag{44}$$

$$I = \{S(a,b), T(a,b), W(a,b), R(a,c)\} \cup E \tag{45}$$

$$X = \{T(a,b), W(a,b), V(b,c)\} \tag{46}$$

$$I^q = \{q_1(a,c), q_2(a,b), q_3(a,b), q_5(a,b)\} \tag{47}$$

$$E^- = \{U(a,b)\} \tag{48}$$

$$E^+ = \{V(b,d)\} \tag{49}$$

We next apply Algorithm 3 to Π and sets E^- and E^+ from equations (48) and (49). The algorithm makes the first call to the module functions during overdeletion in line 28. Function Del^{M^1} implements one-step rule application; thus, due to rules (39) and (41) and the deletion of $U(a,b)$, it overdeletes $S(a,b)$ and $W(a,b)$. Function Del^{M^2} does not update X or I^q at this point since predicate U does not appear in M^2. The algorithm next proceeds to the first iteration of lines 29–34. Function Del^{M^1} overdeletes $T(a,b)$ using $S(a,b)$ and rule (40). Function Del^{M^2} removes $W(a,b)$ from X, removes $q_5(a,b)$, $q_2(a,b)$, and $q_1(a,c)$ from I^q, and finally overdeletes $T(a,b)$ and $R(a,c)$ in line 134. Thus, after the first iteration of lines 29–34, we have $I^q = \{q_3(a,b)\}$. In the second iteration of lines 29–34, function Del^{M^2} is called with $\Delta^- = \{T(a,b), R(a,c)\}$ and $\Delta^m = \{T(a,b), R(a,c)\}$. If at this point we were to skip the facts in Δ^m in line 119, then the q-fact $q_3(a,b)$ would be kept after overdeletion, which would be incorrect: $T(a,b)$ is overdeleted so there does not exist a word that connects a and b and that moves NFA N^R from q_0 to q_3. Therefore, in the addition phase, function Add^{M^2} would join the added fact $V(b,d)$ with $q_3(a,b)$ in line 42; thus, $q_1(a,d)$ would be added to I^q and $R(a,d)$ would be added to I. The latter, however, is incorrect: fact $R(a,d)$ does not hold after update.

Intuitively, facts in Δ^- cannot be skipped in line 119 because they can give rise to partial matches of NFAs. In contrast, Algorithm 5 is conceptually applied to a simpler automaton whose structure allows for such an optimisation.

6.5. Sequencing totally ordered elements

The modules considered in Sections 6.2–6.4 are similar in that they involve negation-free recursive rules that express various forms of connectivity over binary predicates, which raises the question of whether our framework is beneficial only in such scenarios. To show that this is not the case, in this section we present a module that involves just one nonrecursive rule with negation. The motivation for this module arose while working with a well-known financial institution on an application of Datalog in the financial services domain. The goal of our application was to identify fraudulent transactions by describing suspicious transactions declaratively using Datalog and then using a highly optimised Datalog reasoner to analyse the data. A key requirement was to order a set of transactions into a sequence according to the transactions' timestamps. For example, given transactions t_1, t_2, and t_3 with timestamps 2, 5, and 9, respectively, our program had to derive a 'follows' relationship between t_1 and t_2, and t_2 and t_3. This relationship was used in patterns such as 'three consecutive transactions between the same accounts within a time period'. To discuss the technical challenges in doing so, we first reformulate the problem in more abstract terms.

To compare timestamps, we must extend Datalog with *builtin predicates*. To this end, we assume that the signature contains a special binary predicate $<$, which is usually written using the infix notation: instead of $<(t_1, t_2)$, we write $(t_1 < t_2)$. A rule is allowed to contain such atoms only in the body, and each variable occurring in such an atom must also occur in an 'ordinary' positive body atom. Finally, predicate $<$ is interpreted as a total order over all constants in the signature—that is, we extend each dataset I_i^s from equation (5) in Section 3 with the infinite set of facts involving the $<$

predicate. Since $<$ is not allowed to occur in rule heads, the facts involving the $<$ predicate are the same in every I_i^s, so we do not need to store them explicitly; moreover, the restriction on variable use ensures that no rule derives infinitely many facts. Thus, builtin predicates do not change the nature of Datalog in any significant way [18], and it is straightforward to see that our framework can use them too.

With this extension in place, we can now formulate our problem as follows. We assume facts of the form $P(a_i)$ enumerate all constants that need to be arranged into a sequence—that is, unary predicate P identifies the domain of our sequence. Our objective is to derive facts of the form $R(a_i, a_j)$ for all pairs of a_i and a_j such that $a_i < a_j$ holds, and for which no constant a_z exists that satisfies $a_i < a_z < a_j$. This condition is captured directly by the following definition.

Definition 32. For P a unary predicate, R a binary predicate, and $<$ a total order over all constants, $\text{seq}(P, R, <)$ is the module containing the following rule:

$$P(x) \wedge P(y) \wedge (x < y) \wedge \text{not} \exists z.(P(z) \wedge (x < z) \wedge (z < y)) \rightarrow R(x, y). \tag{50}$$

To apply Definition 32 to our example, we would use P to enumerate all relevant time stamps and let R be the 'follows' relation. In practice, it might be more convenient to modify rule (50) so that it derives the 'follows' relationship between transactions, rather than transaction timestamps. Such modifications are straightforward and they do not significantly change the nature of our algorithms, so we do not consider them further for the sake of simplicity.

Now let n be the number of facts of the form $P(a_i)$, and let us assume that atoms of the form $(a_i < a_j)$ can be evaluated in constant time. To evaluate rule (50) using seminaïve evaluation, we need to consider $O(n^2)$ rule instances of the form

$$P(a_i) \wedge P(a_j) \wedge (a_i < a_j) \wedge \text{not} \exists z.(P(z) \wedge (a_i < z) \wedge (z < a_j)) \rightarrow R(a_i, a_j).$$

To evaluate the negative body literal, we need to consider $O(n)$ possible bindings for z: evaluating $(a_i < z)$ may produce infinitely many candidates for z, so we need to evaluate atom $P(z)$ first. Thus, the worst-case complexity of rule evaluation is $O(n^3)$. Due to this, the application we discussed above could initially process only very small inputs.

However, rule (50) can be applied more efficiently: we retrieve all constants $a_1, \ldots, a_n$ occurring in facts with the P predicate, sort these constants according to the $<$ predicate, and derive $R(a_i, a_j)$ for all pairs of adjacent constants in the sorted list. The worst-case complexity of such a solution is clearly determined by the sorting step, which can be realised in $O(n \log n)$ steps using, for example, the heap sort algorithm. Even with an algorithm such as quicksort whose worst-case running time is $O(n^2)$, such a solution is still considerably more efficient than seminaïve evaluation.

Based on this idea, Algorithms 13–15 present functions for the $\text{seq}(P, R, <)$ module. Our main challenge is to adapt our idea to the incremental setting. To avoid recomputing the sorted list of constants in each module function call, our module maintains a global list of sorted constants S^P. The addition function then extends S^P with the constants from facts with the P predicate (line 144); an incremental algorithm such as heap sort can be used to keep S^P sorted. Then, in lines 145–147 the algorithm identifies instances of rule (50) where $P(b) \in \Delta^+$ is matched to $P(x)$ or $P(y)$; and in lines 148–149 the algorithm identifies instances of rule (50) where $P(b) \in \Delta^-$ is matched to $P(z)$. The deletion function is analogous and it essentially just reverses this process. Finally, the rederivation function simply returns each fact $R(a, b) \in \Delta$ where b comes immediately after a in the sorted list S^P. Theorem 33 shows that our solution satisfies the correctness criterion from Definition 14, and it is proved in Appendix E.

Theorem 33. *Functions* $\text{Add}^{\text{seq}(P,R,<)}$, $\text{Del}^{\text{seq}(P,R,<)}$, *and* $\text{Red}^{\text{seq}(P,R,<)}$ *are correct.*

7. Empirical evaluation

To evaluate our work, we have implemented our algorithms in a new research prototype. We then conducted several experiments in order to investigate how our approaches perform under a variety of workloads. In this section, we discuss our test systems and benchmarks, then we present our experimental setup, and finally we discuss the results. Our test systems, test data, and results are available online.[2]

7.1. Test systems

In order to simplify the development effort, we implemented our system on top of data storage and indexing mechanisms from RDFox [61]—a state-of-the-art system for Datalog reasoning over RDF data. Our system thus stores RDF triples, which are facts of the form $\langle s, p, o \rangle$ where s, p, and o are constants called *subject*, *predicate*, and *object*. All modules considered in this paper deal only with unary and binary facts, which we transformed into triples by converting each fact of the form $C(a)$ or $R(a, b)$ into a triple $\langle a, rdf{:}type, C \rangle$ or $\langle a, R, b \rangle$, respectively. RDFox stores all of its triples in RAM, and it provides an interface for retrieving triples matching a single atom; we implemented our algorithms on top of this interface. Our system

---

[2] https://krr-nas.cs.ox.ac.uk/2021/modular-reasoning/.

Algorithm 13 $\text{Add}^{\text{seq}(P,R,<)}[I:\Delta^-, \Delta^+:\Delta^{\text{m}}]$.

Global variable: S^P

143: $J := \emptyset$
144: **add each** b with $P(b) \in \Delta^+$ to S^P while keeping S^P sorted according to $<$
145: **for each** $P(b) \in \Delta^+$ **do**
146:     **if** b has an immediate predecessor a in S^P **then** $J := J \cup \{R(a,b)\}$
147:     **if** b has an immediate successor c in S^P **then** $J := J \cup \{R(b,c)\}$
148: **for each** $P(b) \in \Delta^-$ **do**
149:     **if** b has an immediate predecessor a in S^P and an immediate successor c in S^P **then** $J := J \cup \{R(a,c)\}$
150: **return** $J \setminus ((I \setminus \Delta^-) \cup \Delta^+)$

Algorithm 14 $\text{Del}^{\text{seq}(P,R,<)}[I^{\text{o}}, I^{\text{n}}:\Delta^-, \Delta^+:\Delta^{\text{m}}]$.

Global variable: S^P

151: $J := \emptyset$
152: **for each** $P(b) \in \Delta^-$ **do**
153:     **if** b has an immediate predecessor a in S^P **then** $J := J \cup \{R(a,b)\}$
154:     **if** b has an immediate successor c in S^P **then** $J := J \cup \{R(b,c)\}$
155: **for each** $P(b) \in \Delta^+$ **do**
156:     **if** b has an immediate predecessor a in S^P and an immediate successor c in S^P **then** $J := J \cup \{R(a,c)\}$
157: **remove each** b with $P(b) \in \Delta^-$ from S^P and keep S^P sorted according to $<$
158: **return** $J \cap (I^{\text{n}} \setminus \Delta^-)$

Algorithm 15 $\text{Red}^{\text{seq}(P,R,<)}[I^{\text{o}}, I^{\text{n}}:\Delta]$.

Global variable: S^P

159: $J := \emptyset$
160: **for each** $R(a,b) \in \Delta$ **do**
161:     **if** b is the immediate successor of a in S^P **then** $J := J \cup \{R(a,b)\}$
162: **return** J

Table 3
Benchmark statistics.

Benchmark	$|E|$ (k)	$|I|$ (k)	$|\Pi_{\text{nr}}|$	$|\Pi_{\text{r}}|$	S	tc	stc	rch	seq
Claros-LE	18793.3	533348.9	1031	306	11	27	2	0	0
LUBM-LE	133573.9	332615.4	85	22	5	1	2	0	0
DBpedia-SKOS	5000.0	96992.9	26	15	5	2	1	0	0
DAG-R	100.0	22886.1	1	1	1	1	0	0	0
Family	1000.0	51252.3	202	162	10	5	3	0	0
Relations-STD	854.6	212220.7	985	520	18	42	3	0	0
Relations-ALL						42	3	8	0
Relations-CUS						40	3	1	0
Seq	2001.0	2003.0	1	0	1	0	0	0	1

creates one module per stratum that contains all rules in the stratum for which no specialised algorithms exist. Such rules are handled as in DRed^c: as outlined in Section 2.3, each fact is associated with counters that keep track of nonrecursive and recursive derivations; the former is used to prevent unnecessary overdeletion, and the latter is used to rederive facts without any 'backward' rule evaluation. The nonrecursive counter is also used as a global oracle that is available to all other modules, as described in Section 6.1. Our system thus combines DRed^c with modular reasoning, so we call it M-Dred^c. The system is written in C++ and it runs on Linux.

DRed^c and B/F^c were shown to outperform standard DRed and B/F on a wide range of inputs, so they provide a natural baseline for comparison with M-DRed^c.

7.2. Test benchmarks

We compared our test systems on a range of real-world and synthetic benchmarks. For each benchmark, Table 3 shows the numbers of explicit ($|E|$) and derived ($|I|$) facts, nonrecursive ($|\Pi_{\text{nr}}|$) and recursive ($|\Pi_{\text{r}}|$) rules, and strata (S), as well as the numbers of modules in the partition of the benchmark program (but we do not count the generic modules based on standard rule matching). We next briefly describe each benchmark.

Claros[3] is a dataset about archaeological artefacts, and its structure is described by an OWL ontology. We used the *lower bound extended* (Claros-LE) program by Motik et al. [61], which was obtained from the Claros ontology by removing axioms

---

with features such as existential quantification and disjunction that cannot be captured in Datalog, converting the remaining axioms into rules, and manually adding several 'difficult' rules.

LUBM [37] is a well-known benchmark that models individuals and organisations in a university domain. We used the lower bound extended program by Motik et al. [61], which was obtained analogously to Claros-LE.

DBpedia [52] contains structured information extracted from Wikipedia. Various Wikipedia categories are represented using the SKOS vocabulary [58], which defines several transitive properties. We used the Datalog subset of the SKOS RDF schema. The materialisation of DBpedia-SKOS is too large to fit into the memory of our test server, so we selected a subset of five million facts using uniform sampling.

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 'connected' predicate as transitive.

The Family benchmark was derived from the Family History Knowledge Base (FHKB) [79] that captures genealogical data. The FHKB ontology is interesting because it contains several transitive and symmetric–transitive properties, and we extracted the Datalog subset of the ontology in the usual way. The dataset of FHKB is very small so, to obtain a more substantial dataset, we extracted a subset of the publicly available Familinx[4] genealogical dataset and retrofitted it to the FHKB ontology.

The Relations benchmark is obtained from the Relations Ontology [77], a widely used component of numerous biomedical ontologies. The ontology does not contain any data, so we created a synthetic dataset using the Watdiv Data Generator [4]. This benchmark provided us with many nontrivial regular chain rules, which allowed us to evaluate our algorithms from Section 6.4. However, the Datalog program also contains many transitive and symmetric–transitive rules so, since transitivity axioms are chain rules, it is not obvious how to partition the program into modules. To study how partition choices affect the reasoning performance, we partitioned the Datalog program in three different ways. The STD version provides us with a baseline: all transitive and symmetric–transitive rules were assigned to tc and stc modules, and all remaining chain rules were handled using standard seminaïve evaluation. The ALL version naturally refines the STD version: in each stratum, we selected a maximum subset of the chain rules not already assigned to tc and stc modules that satisfies the regularity condition by Horrocks and Sattler [41], and, if the result was not empty, we created one rch module for the stratum. We thus obtained eight rch modules containing a total of 119 rules; the largest such module contained 73 rules. Finally, as we discuss shortly, our experiments showed that our algorithms for regular chain rules can incur a nontrivial overhead. Therefore, we created a custom (CUS) version with just one rch module that consists of two transitivity rules and several chain rules that we noticed were difficult for seminaïve algorithm.

Seq is a synthetic benchmark designed to test the object sequencing module. The program contains just one rule of the form (50). The dataset contains 2,000 facts of the form $P(a)$, and 1,999,000 facts axiomatising the ordering predicate $<$.

7.3. Test setup and results

To test incremental reasoning, we needed sets of facts to be removed or added. Thus, for each benchmark dataset E, we selected subsets E_1 and E_2 using uniform sampling. For all benchmarks other than Seq, E_1 contained 1,000 facts, and E_2 contained 25% of E. For Seq, E_1 contained 50 facts of the form $P(a)$, and E_2 contained 500 facts of that form—that is, 25% of all the facts in E that are of the form $P(a)$.

We conducted all experiments on a Dell PowerEdge R730 server with 128 GB RAM and two Intel Xeon E5-2660 2.6 GHz processors running Fedora 33, kernel version 5.8.15. For each benchmark, we loaded the benchmark dataset E into our system and then measured the wall-clock time needed to accomplish the following reasoning tasks. First, we computed the materialisation of E. Next, to see how various incremental algorithms deal with small changes, we deleted and then reintroduced subset E_1. Finally, to see how incremental algorithms deal with large changes, we deleted subset E_2. Since materialisation and incremental reasoning is achieved in the same way in all of our algorithms, we did not measure the time for reintroducing E_2. The results of all of our tests are shown in Table 4. Since addition is performed in exactly the same way in DRed^c and B/F^c, we report the times only for the former. Similarly, the partitioning of the program into modules is irrelevant for the DRed^c and B/F^c algorithms, so we report the times of DRed^c and B/F^c just for the STD version. Finally, M-Mat was an order of magnitude slower on Relations-ALL than on Relations-STD, which suggests that the algorithms for regular chain rules from Section 6.4 can incur a nontrivial cost. We believe that these overheads would also dominate the incremental reasoning tests so, in order to run our experiments more quickly, we omitted the incremental reasoning tests for Relations-ALL.

7.4. Discussion

As one can see from our results, M-Mat significantly outperformed the standard seminaïve evaluation on all benchmarks apart from Relations-ALL. Performance improvements range from 3.7 times on LUBM-LE and 5.2 times on Claros-LE, to around 100 times on DAG-R and Family and almost four orders of magnitude on Seq. Most benchmarks contain transitive rules, and many also contain symmetric–transitive rules; moreover, the datasets contain patterns that make evaluation of

---

Table 4

Materialisation and incremental reasoning times in seconds.

Benchmark	Materialisation		Small deletions			Small additions		Large deletions		
	Mat	M-Mat	DRed^c	B/F^c	M-DRed^c	DRed^c	M-DRed^c	DRed^c	B/F^c	M-DRed^c
Claros-LE	4996	962	1216	585	1	1	0	4733	5132	365
LUBM-LE	1268	339	4	0	0	0	0	1568	844	210
DBpedia-SKOS	4191	120	759	515	31	3	0	4026	87287	206
DAG-R	3799	35	3262	1302	104	137	17	4595	10580	139
Family	12052	102	6781	7375	12	432	1	9750	20274	47
Relations-STD	13714	1613	16754	50610	637	568	26	12488	262417	1640
Relations-ALL		20345			–		–			–
Relations-CUS		680			414		19			842
Seq	8393	0	27	27	0	27	0	254	251	0

these rules difficult. For example, the program of Claros-LE contains a symmetric–transitive predicate *relatedPlaces*, and the materialisation contains large cliques of constants connected to each other via this predicate. DBpedia contains long chains/cycles over the transitive *skos:broader* predicate [13]. Similarly, the Family benchmark contains the *isBloodRelationOf* symmetric–transitive predicate. The cost of exploring all rule instances dominates the performance of reasoning, and our techniques seem to be very effective at reducing this cost by exploring only a subset of these rule instances as we discussed in Sections 6.2–6.5. In many cases, the improvement is actually in worst-case complexity, which drops from $O(n^3)$ to $O(n^2)$ and $O(n \log n)$ for stc and seq, respectively.

M-Mat was slower than Mat on Relations-ALL, but much faster on Relations-STD and Relations-CUS. Our investigation revealed that the slowdown is due to two issues: the overhead of maintaining global sets I^q, Δ^q, and X can outweigh the benefits of distinguishing 'external' facts from 'internal' ones; moreover, when there are many interconnected chain rules, matching the 'external' facts in many NFAs can be costly. Nevertheless, results for Relations-STD and Relations-CUS show that our algorithms can still be very useful when they are applied to regular chain rules that generate a large number of 'internal' facts: materialisation and large deletions were twice as fast in the latter case, and the performance of small deletions and additions was improved too. Thus, an important question for our future work is to develop techniques that can automatically identify cases in which our optimisation for regular chain rules is likely to lead to performance improvements.

On incremental reasoning tasks, M-DRed^c is faster in most cases by at least an order of magnitude than DRed^c and B/F^c; moreover, updates are instantaneous for Seq. Again, the improvement is often due to the reduction in the worst-case complexity. For example, when a fact of the form *relatedPlaces*(a, b) is deleted in Claros-LE, 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, the stc module enumerates all elements of the clique, allowing an update to be completed in $O(n^2)$ steps. The worst-case complexity analysis also explains the difference between Mat and DRed^c on Seq: Mat handles four times as much data as DRec^c on large deletion so, since rule matching is cubic, one could expect a slowdown of a factor of around $4^3 = 64$; this is not far from the actual slowdown factor of 33 (i.e., 8,393 vs 254). In contrast, by using well-known and highly optimised sorting algorithms, the seq module eliminates virtually all overheads of rule matching, which renders the difference between M-Mat and M-DRed^c negligible.

The B/F^c algorithm performs significantly worse than DRed^c on all but two benchmarks for large deletions. This is because B/F^c uses 'backward' rule evaluation to search for alternative derivations, which can be very inefficient when rules are complex. As more facts are deleted in the input, more 'backward' rule evaluation is needed; thus, B/F^c can be more efficient than DRed^c on small deletions, but on large deletions DRed^c is more efficient because it completely eliminates 'backward' rule evaluation. Please note, however, that our framework is sufficiently general to capture both algorithms, so the optimised algorithms from Sections 6.2–6.5 can be used with B/F^c, DRed^c, or indeed most related algorithms proposed in the literature.

We point out an apparent oddity in the results for DAG-R: for both DRed^c and M-DRed^c, update times for small deletion were larger than for the initial materialisation, and they were close to the times for large deletions. This is because deleting 1,000 edges from the graph caused a large part of the materialisation to be overdeleted and rederived. Doing so was much more costly than initial materialisation; moreover, the number of deleted edges was only slightly larger in the case of large deletions, which is why the times for small and large deletions were close. Nevertheless, M-DRed^c was faster than DRed^c by at least two orders of magnitude on both types of deletion.

Incremental additions are in general easier to handle than deletions. Intuitively, when an incremental addition algorithm derives a fact that is already contained in the materialisation, the derived fact can simply be ignored without adding any further work. In contrast, when any of the incremental algorithms identifies a fact during overdeletion, this fact can be ignored only if the nonrecursive counter for the fact show that the fact actually holds. Therefore, overdeletion tends to involve much more work and is generally much harder than fact addition. This is clearly reflected in Table 4. Nevertheless, M-DRed^c was several times faster than DRed^c on small additions in all cases apart from LUBM-LE, where both algorithms computed the update instantaneously.

Finally, the results for the two versions of the Relations benchmark show that specialised algorithms for regular chains can lead to significant improvements: the CUS version is around two times faster than the STD version on materialisation

and large deletions. We take this as confirmation that, despite the complexity of implementation, our algorithms for regular chains may bring important benefits to performance-sensitive applications.

8. Conclusion

We have proposed a modular framework for the computation and maintenance of Datalog materialisations. The framework supports the integration of custom algorithms for specific types of rules with standard Datalog reasoning methods. Moreover, we have presented such custom algorithms for programs axiomatising the transitive and the symmetric–transitive closure of a binary predicate, dealing with regular chain rules, and sequencing totally ordered elements. Finally, we have shown empirically that our algorithms consistently outperform the existing ones on a wide range of benchmarks, and that they are faster by several orders of magnitude in many cases. Thus, our algorithms can play a critical role in performance-sensitive applications that need to deal with complex rules. In our future work, we shall identify other modules that can be implemented more efficiently using custom algorithms. Moreover, we shall also consider the problem of automating the process of partitioning a Datalog program into modules. At present, our system expects a module partition to be produced manually; moreover, our evaluation results have shown that the performance of our approach can depend on the exact partition used, particularly when using complex chain rules. To identify a partition that promises optimal performance, we shall consider developing a model for estimating the cost of a given partition, which would allow us to consider the partitioning problem as a kind of optimisation problem.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Appendix A. Proof of Theorem 15

Theorem 15. *Let Π be a program, let λ be a stratification of Π with maximum stratum index S, let $M^{s,k}$ with $1 \leq s \leq S$ and $1 \leq k \leq n_s$ be the module partition of Π w.r.t. λ such that module functions for each $M^{s,k}$ are correct, let $E_0, \ldots, E_m$ be datasets, let I_0 be result of applying Algorithm 2 to Π, λ, and E_0, and, for $1 \leq i \leq m$, let I_i be the result of successively applying Algorithm 3 to Π, λ, $E_i^- = E_{i-1} \setminus E_i$, and $E_i^+ = E_i \setminus E_{i-1}$. Then, $I_i = \Pi_\infty[E_i]$ for each $0 \leq i \leq m$.*

For the rest of this section, we fix an arbitrary program Π, an arbitrary stratification λ of Π, and arbitrary modules $M^{s,k}$ with $1 \leq s \leq S$ and $1 \leq k \leq n_s$ that constitute a module partition of Π w.r.t. λ such that module functions for each $M^{s,k}$ are correct. Moreover, we also fix an arbitrary sequence of datasets $E_0, \ldots, E_m$, we let I_0 be result of applying Algorithm 2 to Π, λ, and E_0, and, for $1 \leq i \leq m$, we let I_i be the result of successively applying Algorithm 3 to Π, λ, $E_i^- = E_{i-1} \setminus E_i$, and $E_i^+ = E_i \setminus E_{i-1}$. Finally, let $H_m^{s,k}$ be the call history for module $M^{s,k}$ obtained by such invocation of Algorithms 2 and 3; and for $0 \leq i \leq m$ and $1 \leq j \leq h_i$, let $H_{i,j}^{s,k}$ be the subset of $H_m^{s,k}$ up to and including call $C_{i,j}$. Note that our algorithms always call the functions of all modules in some stratum in parallel. Thus, for any two modules $M^{s,k}$ and $M^{s,k'}$ belonging to a stratum with the same index s, histories $H_m^{s,k}$ and $H_m^{s,k'}$ consist of the same number of calls, and, for each $0 \leq i \leq m$ and $1 \leq j \leq h_m$, calls $C_{i,j}$ and $C'_{i,j}$ from these two histories are of the same type and have the same arguments (but, clearly, their results can differ). Specifically, the numbers $h_0, \ldots, h_m$ in these histories depend only on the stratum index s, so we often refer to $h_0, \ldots, h_m$ without specifying a history. We now prove by induction on m that the claim of Theorem 15 and the following properties hold for each module $M^{s,k}$:

(H1) call history $H_m^{s,k}$ is compatible with Π, λ, s, and $E_0, \ldots, E_m$, and
(H2) $H_m^{s,k}$ ends with $J_{m,h_m} := \mathsf{Add}^{M^{s,k}}[I_{m,h_m} \stackrel{.}{:} \Delta_{m,h_m}^-, \Delta_{m,h_m}^+ \stackrel{.}{:} \Delta_{m,h_m}^\mathsf{m}]$ where $J_{m,h_m} = \emptyset$ and $\Pi_\infty[E_m]^{\leq s} = ((I_{m,h_m} \setminus \Delta_{m,h_m}^-) \cup \Delta_{m,h_m}^+) \cap \mathsf{O}^{\leq s}$.

For the induction base, in Claim 34 we consider applying Algorithm 2 to E_0. For the inductive step, in Claim 35 we assume that Theorem 15 and properties (H1) and (H2) hold for $m-1$, and we show that they also hold after applying Algorithm 3 to E_m. In all cases, it is obvious that the arguments of each call satisfy the conditions specified in Table 1, so we do not discuss this point any further for the sake of brevity.

Claim 34. *Theorem 15 and properties (H1) and (H2) hold for $m = 0$.*

Proof. We consider the run of Algorithm 2 on E_0. For each stratum index s and each module $M^{s,k}$, the last call in $H_{s,k}$ is clearly of the form $J_{0,h_0} := \mathsf{Add}^{M^{s,k}}[I_{0,h_0} \,\vdots\, \Delta^-_{0,h_0}, \Delta^+_{0,h_0} \,\vdots\, \Delta^{\mathsf{m}}_{0,h_0}]$, the loop in lines 13–18 for stratum with index s terminates only when $J_{0,h_0} \subseteq \Delta = \emptyset$, and applying property (M1) below for m ensures that property (H2) holds.

Now let $I^0 = \emptyset$ and let I^s be the content of I after stratum with index s is processed. By induction on s, we prove that property (H1) holds each $s > 0$ and that $I^s = \Pi_\infty[E_0]^{\leq s}$ holds as well; for $s = S$, these clearly imply Claim 34. The base case holds trivially for $s = 0$. For the induction step, we assume that $I^{s-1} = \Pi_\infty[E_0]^{<s}$ holds, and we prove properties (M1)–(M3) shown below, which imply our claim.

Soundness and property (H1). Let $I^s_1, I^s_2, \ldots$ and $\Delta^s_1, \Delta^s_2, \ldots$ be the contents of I and Δ, respectively, in successive iterations just after line 15 for the stratum with index s. We now show by induction on the steps of Algorithm 2 that, for each $1 \leq j \leq h_0$,

(M1) for each $M^{s,k}$, call $J_{0,j} := \mathsf{Add}^{M^{s,k}}[I_{0,j} \,\vdots\, \emptyset, \Delta^+_{0,j} \,\vdots\, \Delta^{\mathsf{m}}_{0,j}]$ satisfies conditions in Table 2, and $I^s_j = I_{0,j} \cup \Delta^+_{0,j}$, and
(M2) $I^s_j \subseteq \Pi_\infty[E_0]^{\leq s}$ and $\Delta^s_j \subseteq \Pi_\infty[E_0]^s$.

For the base case $j = 1$, consider line 12. For an arbitrary module $M^{s,k}$, the call is of the form $J^{s,k}_{0,1} := \mathsf{Add}^{M^{s,k}}[\emptyset \,\vdots\, \emptyset, \Delta^+_{0,1} \,\vdots\, \emptyset]$, where $\Delta^+_{0,1} = I^{s-1} \cup (E_0 \cap \mathsf{O}^s) = I^s_1 \subseteq \Pi_\infty[E_0]^{\leq s}$ and the first equality holds by line 10; thus, properties (A1.a) and (M1) hold. This call is thus correct in the context of E_0, Π, λ, and s by the assumption of Theorem 15, so $J^{s,k}_{0,1} \subseteq \Pi_\infty[E_0] \cap \mathsf{O}^s = \Pi_\infty[E_0]^s$. By combining the outputs of all modules in line 15, we have $\Delta^s_1 \subseteq \Pi_\infty[E_0]^s$, so property (M2) holds.

For the inductive step, assume that properties (M1) and (M2) hold for some $j - 1$ and consider line 18. For an arbitrary module $M^{s,k}$, the call is of the form $J^{s,k}_{0,j} := \mathsf{Add}^{M^{s,k}}[I_{0,j} \,\vdots\, \emptyset, \Delta^+_{0,j} \,\vdots\, \Delta^{\mathsf{m}}_{0,j}]$, where $I_{0,j} = I^s_{j-1}$ and $\Delta^+_{0,j} = \Delta^s_{j-1}$. The previous call is clearly of the form $J^{s,k}_{0,j-1} := \mathsf{Add}^{M^{s,k}}[I_{0,j-1} \,\vdots\, \emptyset, \Delta^+_{0,j-1} \,\vdots\, \Delta^{\mathsf{m}}_{0,j-1}]$, so property (A2.a) holds. Moreover, property (M1) holds for $j-1$ by induction assumption, so we have $I_{0,j} = I^s_{j-1} = I_{0,j-1} \cup \Delta^+_{0,j-1}$, as required for property (A2.b). In addition, property (M2) holds for $j-1$ by induction assumption, so we have $\Delta^+_{0,j} = \Delta^s_{j-1} \subseteq \Pi_\infty[E_0]^s$. Finally, $\Delta^{\mathsf{m}}_{0,j} = J^{s,k}_{0,i-1} \subseteq \Delta^+_{0,j} = \Delta^s_{j-1}$ holds obviously by the structure of the algorithm, as required for property (A2.c). Thus, property (M1) holds for j. This call is thus correct in the context of E_0, Π, λ, and s by the assumption of Theorem 15, so $J^{s,k}_{0,j} \subseteq \Pi_\infty[E_0] \cap \mathsf{O}^s = \Pi_\infty[E_0]^s$. Hence, as in the previous paragraph, we conclude that property (M2) holds for j.

Completeness. Let $\mathcal{I}^s_0, \mathcal{I}^s_1, \ldots$ be the sequence of datasets and let $\mathcal{I}^s_\infty$ be the dataset from the construction of $\Pi_\infty[E_0]$ for stratum with index s from equation (5). By induction on this sequence, we next show the following property:

(M3) $\mathcal{I}^s_i \subseteq I^s$ holds for each $i \geq 0$.

By the inductive assumption on s, we have $I^{s-1} = \mathcal{I}^{s-1}_\infty$. For the base case, $\mathcal{I}^s_0 = \mathcal{I}^{s-1}_\infty \cup (E \cap \mathsf{O}^s) \subseteq I^s$, where the inclusion holds due to line 10. For the inductive step, we assume that property (M3) holds for $j - 1$, and we show that it holds for j as well. To this end, we consider an arbitrary fact $F \in \mathcal{I}^s_i \setminus \mathcal{I}^s_{i-1}$ derived by an instance $r\sigma$ of a rule $r \in \Pi^s$; thus, $F = \mathsf{h}(r\sigma)$ and $\mathcal{I}^s_{i-1} \models \mathsf{b}(r\sigma)$. Since Π is stratified, all atoms occurring in $\mathsf{b}^-(r\sigma)$ belong to a stratum with index less than s; moreover, $I^s \cap \mathsf{O}^{<s} = I^{s-1} = \mathcal{I}^s_{i-1} \cap \mathsf{O}^{<s}$ clearly holds; thus, $\mathcal{I}^s_{i-1} \models \mathsf{b}^-(r\sigma)$ implies $I^s \models \mathsf{b}^-(r\sigma)$. Moreover, $\mathcal{I}^s_{i-1} \models \mathsf{b}^+(r\sigma)$ implies $\mathsf{b}^+(r\sigma) \subseteq \mathcal{I}^s_{i-1}$, and $\mathcal{I}^s_{i-1}$ satisfies property (M3) by the induction assumption, so we have $\mathsf{b}^+(r\sigma) \subseteq I^s$. This means that, either $\mathsf{b}^+(r\sigma) \subseteq I$ holds in line 11, or $\mathsf{b}^+(r\sigma) \subseteq I \cup \Delta$ holds in line 18 at some point during the algorithm's execution. By the definition of module partition, there exists a module $M^{s,k}$ such that $r \in M^{s,k}$. The corresponding call to $\mathsf{Add}^{M^{s,k}}$ in line 11 or 18 is correct in the context of E_0, Π, λ, and s by property (M1), so $F \in M_{\mathsf{A}}[\emptyset \,\vdots\, \emptyset, I] \subseteq \Delta_k$ or $F \in M_{\mathsf{A}}[I \,\vdots\, \emptyset, \Delta] \subseteq \Delta_k$ holds, where the membership of F is ensured by Definition 8. Thus, $F \in I^s$, and property (M3) holds. □

Claim 35. *If Theorem 15 and properties (H1) and (H2) hold for $m - 1$, they also hold for m.*

Proof. We consider the run of Algorithm 2 on E_m for $m \geq 1$. The induction assumption for $m-1$ ensures that $I = \Pi_\infty[E_{m-1}]$ holds during the entire run up to line 24.

Let $I^0 = \emptyset$, let I^s be the content of $(I \setminus D) \cup A$ after stratum with index s is processed, and let I^s_D be the content of $(I \setminus D) \cup A$ after function $\mathsf{Overdeletion}$ processes stratum with index s. By induction on s, we prove (H1) and the following property:

(U1) $I^s \cap \mathsf{O}^{\leq s} = \Pi_\infty[E_m]^{\leq s}$.

For each module $M^{s,k}$, the last call of the algorithm is clearly of type $\mathsf{Add}^{M^{s,k}}$, and property (H2) holds in the same way as in the proof of Claim 34; for $s = S$, these clearly imply Claim 35. The base case holds trivially for $s = 0$. For the inductive step, we assume that property (U1) holds for $s - 1$, and we consider the three phases for stratum with index s.

Overdeletion maintains property (H1). Let d be the index of the last $\mathsf{Del}^{M^{s,k}}$ call. Moreover, let $I_1^s, I_2^s, \ldots$ and $\Delta_1^s, \Delta_2^s, \ldots$ be the contents of $(I \setminus D) \cup A$ and Δ, respectively, in successive passes through line 34 for the stratum with index s. We now show by induction on the steps of function OVERDELETE that, for each $1 \le j \le d$,

(U2) for each $M^{s,k}$, call $J_{m,j} := \mathsf{Del}^{M^{s,k}}[I, I_{m,j}^{\mathsf{n}} \vdots \Delta_{m,j}^-, \Delta_{m,j}^+ \vdots \Delta_{m,j}^{\mathsf{m}}]$ satisfies conditions in Table 2, and $I_j^s = (I_{m,j}^{\mathsf{n}} \setminus \Delta_{m,j}^-) \cup \Delta_{m,j}^+$.
(U3) $\Delta_j^s \subseteq \Pi_\infty[E_{m-1}]^s$.

For the base case $j = 1$, consider line 28. For any module $M^{s,k}$, the call is of the form $J_{m,1}^{s,k} := \mathsf{Del}^{M^{s,k}}[I, I_{m,1}^{\mathsf{n}} \vdots \Delta_{m,1}^-, \Delta_{m,1}^+ \vdots \emptyset]$, where $I_{m,1}^{\mathsf{n}} = I$. The previous call is clearly of type $\mathsf{Add}^{M^{s,k}}$, and is clearly satisfies properties (D1.a) and (D1.b). Property (D1.c) follows from property (H2) for $m - 1$. Property (D1.d) holds trivially. Moreover, property (U1) holds for $s - 1$, so $I^{s-1} \cap \mathsf{O}^{<s} = \Pi_\infty[E_m]^{<s}$; thus, in line 28 we have $D \setminus A = \Pi_\infty[E_{m-1}]^{<s} \setminus \Pi_\infty[E_m]^{<s}$ and $A \setminus D = \Pi_\infty[E_m]^{<s} \setminus \Pi_\infty[E_{m-1}]^{<s}$; finally, line 26 ensures $\Delta = (E_i \setminus E_{i-1}) \cap \mathsf{O}^s$; consequently, properties (D1.e) and (D1.f) hold. In addition, $I_1^s = (I_{m,1}^{\mathsf{n}} \setminus \Delta_{m,1}^-) \cup \Delta_{m,1}^+$ clearly holds, as required for property (U2). This call is correct in the context of E_m, Π, λ, and s by the assumption of Theorem 15, so we have $J_{m,1}^{s,k} \subseteq I \cap \mathsf{O}^s \subseteq \Pi_\infty[E_{m-1}]^{\le s}$. By combining the outputs of all modules in line 31, we have $\Delta_1^s \subseteq \Pi_\infty[E_{m-1}]^s$, so property (U3) holds.

For the inductive step, assume that properties (U2) and (U3) hold for some $j - 1$ and consider line 34 For an arbitrary module $M^{s,k}$, the call is of the form $J_{m,j}^{s,k} := \mathsf{Del}^{M^{s,k}}[I, I_{m,j}^{\mathsf{n}} \vdots \Delta_{m,j}^-, \emptyset \vdots \Delta_{m,j}^{\mathsf{m}}]$, where $I_{m,j}^{\mathsf{n}} = I_{j-1}^s$ and $\Delta_{m,j}^- = \Delta_{j-1}^s$. The previous call is clearly of the form $J_{m,j-1}^{s,k} := \mathsf{Del}^{M^{s,k}}[I, I_{m,j-1}^{\mathsf{n}} \vdots \Delta_{m,j-1}^-, \Delta_{m,j-1}^+ \vdots \Delta_{m,j-1}^{\mathsf{m}}]$, so property (D2.a) holds. Property (D2.b) holds trivially. Moreover, property (U2) holds for $j - 1$ by induction assumption, so we have $I_{m,j}^{\mathsf{n}} = I_{j-1}^s = (I_{m,j-1}^{\mathsf{n}} \setminus \Delta_{m,j-1}^-) \cup \Delta_{m,j-1}^+$, as required for property (D2.c). In addition, property (U3) holds for $j - 1$ by the induction assumption, so we have $\Delta_{m,j}^- = \Delta_{j-1}^s \subseteq \Pi_\infty[E_{m-1}]^s$. Finally. $\Delta_{m,j}^- = J_{m,j-1}^{s,k} \subseteq \Delta_{m,j}^- \subseteq \Pi_\infty[E_{i-1}]^s$ holds obviously by the structure of the algorithm. Thus, property (U2) holds for j. This call is thus correct in the context of E_m, Π, λ, and s by the assumption of Theorem 15, so $J_{m,j}^{s,k} \subseteq I \cap \mathsf{O}^s \subseteq \Pi_\infty[E_{m-1}]^s$. Hence, as in the previous paragraph, we conclude that property (U3) holds for j.

Overdeletion is complete. We now prove that the overdeletion phase indeed deletes all facts of O^s that no longer hold due to the update. Towards this goal, let $\mathcal{I}_0^s, \mathcal{I}_1^s, \ldots$ be the sequence of datasets and let $\mathcal{I}_\infty^s$ be the dataset from the construction of $\Pi_\infty[E_{m-1}]$ for stratum with index s from equation (5). By induction on this sequence, we next show the following property:

(U4) $(\mathcal{I}_i^s \setminus \Pi_\infty[E_m]^{\le s}) \cap I_D^s = \emptyset$.

For the induction base, property (U1) ensures $I^s \cap \mathsf{O}^{<s} = \Pi_\infty[E_m]^{<s}$; moreover, all facts added to set D in the overdeletion phase are from O^s, so $I_D^s \cap \mathsf{O}^{<s} = I^s \cap \mathsf{O}^{<s} = \Pi_\infty[E_m]^{<s}$. Since $\mathcal{I}_\infty^{s-1} = \Pi_\infty[E_{m-1}]^{<s}$, we clearly have $(\mathcal{I}_\infty^{s-1} \setminus \Pi_\infty[E_m]^{<s}) \cap I_D^s = \emptyset$. Now $\mathcal{I}_0^s = \mathcal{I}_\infty^{s-1} \cup (E \cap \mathsf{O}^s)$ by equation (5). Thus, $\mathcal{I}_0^s \setminus \Pi_\infty[E_m]^{\le s} \subseteq E^- \cap \mathsf{O}^s$, and $E^- \cap \mathsf{O}^s \cap I_D^s = \emptyset$ due to lines 26 and 30. Consequently, property (U4) holds for $i = 0$.

For the induction step, we assume that property (U4) holds for some $i - 1$, and we consider an arbitrary fact $F \in \mathcal{I}_i^s \setminus \mathcal{I}_{i-1}^s$ derived by instance $r\sigma$ of a rule $r \in \Pi^s$ such that $F \notin \Pi_\infty[E_m]^{\le s}$. The last observation implies $\Pi_\infty[E_m]^{\le s} \not\models \mathsf{b}(r\sigma)$. By the definition of module partition, there exists a module $M^{s,k}$ such that $r \in M^{s,k}$. We now consider the following possibilities.

- There exists a negative body literal $L \in \mathsf{b}^-(r\sigma)$ such that $\Pi_\infty[E_m]^{\le s} \not\models L$. Then $\Pi_\infty[E_m]^{<s} = I_D^s \cap \mathsf{O}^s$ and the fact that L contains atoms from stratum with index less than s ensure $I_D^s \not\models L$. Moreover, property (H1) ensures that the call to $\mathsf{Del}^{M^{s,k}}$ in line 28 is correct in the context of E_m, Π, λ, and s, so we have $F \in M_D[I, I \vdots \Delta \cup (D \setminus A), A \setminus D]$. But then, lines 31 and 30 ensure $F \notin I_D^s$, as required.
- $\emptyset \subsetneq \mathsf{b}^+(r\sigma) \setminus \Pi_\infty[E_m]^{\le s} \subseteq \mathsf{O}^{<s}$—that is, at least one positive body all of $r\sigma$ does not hold after update, and all such atoms are from stratum with index less than s. But then, we have $(\mathsf{b}^+(r\sigma) \setminus \Pi_\infty[E_m]^{\le s}) \cap I_D^s = \emptyset$ and $F \notin I_D^s$ by the call in line 28 in the same way as in the previous case.
- $\mathsf{O}^s \cap (\mathsf{b}^+(r\sigma) \setminus \Pi_\infty[E_m]^{\le s}) \neq \emptyset$. Since $\mathsf{b}^+(r\sigma) \subseteq \mathcal{I}_{i-1}^s$, property (U4) for $i - 1$ ensures $\mathsf{O}^s \cap (\mathsf{b}^+(r\sigma) \setminus \Pi_\infty[E_m]^{\le s}) \cap I_D^s = \emptyset$. But then, there is a point when the last fact in $\mathsf{O}^s \cap (\mathsf{b}^+(r\sigma) \setminus \Pi_\infty[E_m]^{\le s})$ is in the set Δ, all other facts are in D, and $\mathsf{Del}^{M^{s,k}}$ is called in either line 28 or 34. But then, we have $F \notin I_D^s$ in the same way as in the previous two cases.

Rederivation maintains property (H1). Consider the call of the form $J_{i,j}^{s,k} := \mathsf{Red}^{M^{s,k}}[I, I_{i,j}^{\mathsf{n}} \vdots \Delta_{i,j}]$ in line 37 for some module $M^{s,k}$. Properties (R.a), (R.b), and (R.c) hold trivially, and property (R.d) follows from (U2). Moreover, we have $I_{i,j}^{\mathsf{n}} = I_D^s$, so property (D1.d) for the first $\mathsf{Del}^{M^{s,k}}$ call and the fact that only facts from O^s are added to D during overdeletion ensure $I_D^s \cap \mathsf{O}^{<s} = \Pi_\infty[E_m]^{<s}$ and $I_D^s \cap \mathsf{O}^s \subseteq \Pi_\infty[E_{m-1}]^s$; moreover, property (U4) implies $(\Pi_\infty[E_{m-1}]^{\le s} \setminus \Pi_\infty[E_m]^{\le s}) \cap I_D^s = \emptyset$,

which together with earlier observations ensures $I_D^s \cap O^s \subseteq \Pi_\infty[E_m]^{\leq s}$, as required for the last inclusion of property (R.e); the left equality in property (R.e) follows from the observation that the first deletion call satisfies properties (D1.d), (D1.e), and (D1.f) and in all calls of type D2 satisfy property (D2.d)−that is, all deleted facts are from stratum with index s. Finally, $D \cap O^s = \Pi_\infty[E_{m-1}]^s \setminus I_D^s$ holds at the point when the call is made, which, together with how argument $\Delta_{i,j}$ is computed, ensures property (R.f).

Addition is sound and it maintains property (H1). Let d be the index of the first $\mathsf{Add}^{M^{s,k}}$ call in the addition phase. Moreover, let $I_{d+1}^s, I_{d+2}^s, \ldots$ and $\Delta_{d+1}^s, \Delta_{d+2}^s, \ldots$ be the contents of I and Δ, respectively, in successive iterations just after line 45 for the stratum with index s. We now show by induction on the steps of Algorithm 2 that, for each $d \leq j \leq h_m$,

(U5) for each $M^{s,k}$, call $J_{m,j} := \mathsf{Add}^{M^{s,k}}[I_{m,j} \vdots \Delta_{m,j}^-, \Delta_{m,j}^+ \vdots \Delta_{m,j}^m]$ satisfies conditions in Table 2, and $I_j^s = (I_{m,j} \setminus \Delta_{m,j}^-) \cup \Delta_{m,j}^+$, and

(U6) $I_j^s \cap O^{\leq s} \subseteq \Pi_\infty[E_m]^{\leq s}$ and $\Delta_j^s \subseteq \Pi_\infty[E_m]^s$.

For the base case, consider line 42. For an arbitrary module $M^{s,k}$, the call is of the form $J_{m,j}^{s,k} := \mathsf{Add}^{M^{s,k}}[I \vdots \Delta_{m,j}^-, \Delta_{m,j}^+ \vdots \Delta_{i,j}^m]$, and the previous call is of the form $J_{m,j-1} := \mathsf{Red}^{M^{s,k}}[I, I_{m,j-1}^n \vdots \Delta_{m,j-1}]$. Properties (A3.a) and (A3.b) clearly hold. Properties (A3.c) and (A3.e) hold because the first call to the $\mathsf{Del}^{M^{s,k}}$ function satisfies properties (D1.e) and (D1.f); property (U3) ensures that all facts removed from the materialisation during the deletion phase are in O^s; and property (R.f) ensures that all facts introduced by rederivation are also in O^s. Moreover, $I_{i,j-1}^n = I_D^s$, which implies property (A3.d). Furthermore, line 38 ensures $(E_{m-1} \cap E_m \cap \Delta_{m,j-1}) \subseteq \Delta_{m,j}^+$ and $\Delta_{m,j}^m = J_{m,j-1}^{s,k} \subseteq \Delta_{m,j}^+$; also, the call to $\mathsf{Add}^{M^{s,k}}$ is correct in the context of E_m, Π, λ, and s by the assumption of Theorem 15, which implies $J_{m,j}^{s,k} \subseteq \Pi_\infty[E_m] \cap O^s = \Pi_\infty[E_m]^s$; thus, property (A3.f) holds. In addition, $I_d^s = (I \setminus \Delta_{m,j}^-) \cup \Delta_{m,j}^+$ clearly holds, as required for property (U5). By combining the outputs of all modules in line 38, we have $\Delta_d^s \subseteq \Pi_\infty[E_m]^s$, so property (U6) holds.

The proof for the inductive step is analogous to the proof used to prove Claim 34, so we omit the details for the sake of brevity.

Addition is complete. We now prove that the addition phase indeed derives all facts of O^s that hold after the update. Towards this goal, let $\mathcal{I}_0^s, \mathcal{I}_1^s, \ldots$ be the sequence of datasets and let $\mathcal{I}_\infty^s$ be the dataset from the construction of $\Pi_\infty[E_m]$ for stratum with index s from equation (5). By induction on this sequence, we next show the following property:

(U7) $\mathcal{I}_i^s \subseteq \Pi_\infty[E_m]^{\leq s} = I^s$.

For the base case, property (U4) ensures $I_D^s \cap O^{<s} = \Pi_\infty[E_m]^{<s}$ so, since all facts added to sets D and A are from O^s, we clearly have $I^s \cap O^{<s} = \Pi_\infty[E_m]^{<s} = \mathcal{I}_\infty^{s-1}$. By equation (5), $\mathcal{I}_0^s = \mathcal{I}_\infty^{s-1} \cup (E_m \cap O^s)$. Now consider an arbitrary fact $F \in \mathcal{I}_0^s$. If $F \in I_D^s$, then $I_D^s \subseteq I^s$ ensures $F \in I^s$. If $F \notin I_D^s$ but $F \in E_{m-1} \cap E_m$, then $F \in D$ holds in line 38, so F is added to Δ in that line and line 44 ensures $F \in I^s$. Finally, $F \notin E_{m-1}$, then F is added to Δ in line 40, so line 44 ensures $F \in I^s$.

For the inductive step, we assume that property (U7) holds for some $i - 1$, and we consider an arbitrary fact $F \in \mathcal{I}_i^s \setminus \mathcal{I}_{i-1}^s$ derived by instance $r\sigma$ of a rule $r \in \Pi^s$; note that this implies $F \notin E_m$. There exists a module $M^{s,k}$ such that $r \in M^{s,k}$. Now if $F \in I_D^s$, then $I_D^s \subseteq I^s$ ensures $F \in I^s$; thus, in the rest of thus proof we assume that $F \notin I_D^s$ holds. We consider the following possibilities.

- $I_D^s \models \mathsf{b}(r\sigma)$. Clearly, then $\Pi_\infty[E_{m-1}] \models \mathsf{b}(r\sigma)$ holds as well, so $F \in (\Pi^s)_\mathsf{R}[I, I_D^s \vdots \Delta]$ holds assuming $F \in \Delta$ is satisfied. Moreover, $F \in D$ and $F \notin E \setminus E^-$ both hold in line 37, and the calls to $\mathsf{Red}^{M^{s,k}}$ in this line are correct the context of E_m, Π, λ, and s. Thus, $F \in \Delta$ holds in line 38, so line 44 ensures $F \in I^s$.
- $I_D^s \not\models \mathsf{b}(r\sigma)$ and all atoms occurring $\mathsf{b}(r\sigma)$ are from stratum with index less than s. By the assumptions thus far, we clearly have $\Pi_\infty[E_{m-1}]^{<s} \not\models \mathsf{b}(r\sigma)$ and $\Pi_\infty[E_m]^{<s} \models \mathsf{b}(r\sigma)$. Moreover, in line 42, we have $A \setminus D = \Pi_\infty[E_m]^{<s} \setminus \Pi_\infty[E_{m-1}]^{<s}$ and $(D \setminus A) \cap O^{<s} = \Pi_\infty[E_{m-1}]^{<s} \setminus \Pi_\infty[E_m]^{<s}$. Thus, there exists either a positive body atom $B \in \mathsf{b}^+(r\sigma)$ such that $A \setminus D \models B$, or a negative body literal $L \in \mathsf{b}^-(r\sigma)$ such that $D \setminus A \models L$. The call to $\mathsf{Add}^{M^{s,k}}$ is correct in the context of E_m, Π, λ, and s, so it returns F, which is eventually added to A in line 44. Thus, we have $F \in I^s$.
- There exists a positive body atom $B \in \mathsf{b}^+(r\sigma)$ from stratum with index s such that $F \notin I_D^s$. By induction assumption, we have $\mathsf{b}^+(r\sigma) \subseteq I^s$, which implies $B \in I^s$. Thus, at some point during the algorithm's run, either $\mathsf{b}^+(r\sigma) \subseteq (I \setminus D) \cup A \cup \Delta$ holds in line 42, or $\mathsf{b}^+(r\sigma) \cap \Delta \neq \emptyset$ holds in line 48. The call to $\mathsf{Add}^{M^{s,k}}$ in that point is correct in the context of E_m and Π, so it returns F, which is eventually added to A in line 44. Thus, we have $F \in I^s$. □

Appendix B. Proof of Theorem 18

Theorem 18. *Functions $\mathsf{Add}^{tc(R)}$, $\mathsf{Del}^{tc(R)}$, and $\mathsf{Red}^{tc(R)}$ are correct.*

Consider an arbitrary binary predicate R, program Π, stratification λ of Π, stratum index s such that $\text{tc}(R) \subseteq \Pi^s$, sequence of datasets $E_0, \ldots, E_m$ of datasets, and a call history H for $\text{tc}(R)$ that is compatible with Π, λ, s and $E_0, \ldots, E_m$. We assume that H is of the form as specified in Definition 12. Moreover, for each $0 \leq i \leq m$ and each $1 \leq j \leq n_i$, let $X_{i,j}^R$ and $Y_{i,j}^R$ be the values of X^R and Y^R, respectively, after call $C_{i,j}$. Finally, we define $X_{0,0}^R = Y_{0,0}^R = \emptyset$ and, for $1 \leq i \leq m$, we let $X_{i,0}^R = X_{i-1,n_{i-1}}^R$ and $Y_{i,0}^R = Y_{i-1,n_{i-1}}^R$.

We next introduce some useful abbreviations and notation. To simplify the notation, let $M = \text{tc}(R)$ for the rest of this section. Also, let S be a set of facts. Then, $R\text{-part}[S]$ is the subset of S containing precisely all facts of S whose predicate is R. Moreover, for constants u and v and integer $\ell \geq 0$, we write $u \leadsto^\ell v \in S$ if there exist constants $w_0, w_1, \ldots, w_\ell$ such that $u = w_0$, $w_\ell = v$, and $R(w_{i-1}, w_i) \in S$ for each $1 \leq i \leq \ell$; in other words, u and v are connected in S by a chain of R-facts of length ℓ. Moreover, we write $u \leadsto v \in S$ if there exists $\ell \geq 0$ such that $u \leadsto^\ell v \in S$; in other words, u and v are connected in S by a chain of R-facts of an arbitrary (possibly zero) length. Finally, for Δ another set of facts, let

$$\text{close}[S, \Delta] = \{R(u, v) \mid \text{there exist } \ell \geq 1, \text{ constants } w_0, \ldots, w_\ell, \text{ and an integer } k \text{ with } 0 \leq k < \ell \text{ such that}$$
$$w_0 = u, \ w_\ell = v, \text{ and } R(w_k, w_{k+1}) \in \Delta \text{ and } R(w_m, w_{m+1}) \in S \text{ for } m \tag{B.1}$$
$$\text{with } 0 \leq m < \ell \text{ and } m \neq k\}$$

We prove Theorem 18 by showing that, for each $0 \leq i \leq m$ and each $1 \leq j \leq h_i$, call $C_{i,j}$ in H satisfies properties (T2)–(T4).

(T1) If call $C_{i,j}$ is of the form $J_{i,j} := \text{Add}^M[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^m]$, then the following properties hold.

$$R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j} = M_\infty[X_{i,j}^R] \subseteq \Pi_\infty[E_i]^s \tag{B.2}$$
$$Y_{i,j}^R = \emptyset \tag{B.3}$$

(T2) If call $C_{i,j}$ is of the form $J_{i,j} := \text{Del}^M[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^m]$, then the following properties hold.

$$\text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-] = R\text{-part}[(I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-] \cup J_{i,j} \cup Y_{i,j}^R \tag{B.4}$$
$$X_{i,j}^R = X_{i,0}^R \setminus ((I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-) \subseteq R\text{-part}[I_{i,j}^n] \subseteq R\text{-part}[I_{i,j}^o] = M_\infty[X_{i,0}^R] \tag{B.5}$$
$$Y_{i,j}^R \subseteq I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s} \tag{B.6}$$

(T3) If call $C_{i,j}$ is of the form $J_{i,j} := \text{Red}^M[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}]$, then the following properties hold.

$$R\text{-part}[I_{i,j}^n] \cup J_{i,j} \subseteq M_\infty[X_{i,j}^R] \subseteq (R\text{-part}[I_{i,j}^o] \setminus \Delta_{i,j}) \cup J_{i,j} \subseteq \Pi_\infty[E_i]^s \tag{B.7}$$
$$Y_{i,j}^R = \emptyset \tag{B.8}$$

(T4) Call $C_{i,j}$ is correct.

We prove this by double induction on i and j, where we consider each call $C_{i,j}$ of type from Table 2. The base case involves a call of type A1, and the inductive step involves all remaining calls. We split the proof into a separate claim for each module function.

Claim 36. *If $C_{i,j}$ is of the form $J_{i,j} := \text{Add}^M[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^m]$ and either $i = 0$ and $j = 1$, or $j > 1$ and call $C_{i,j-1}$ satisfies properties (T1)–(T4), then $C_{i,j}$ satisfies properties (T1) and (T4).*

Proof. We first capture the preconditions for the call $C_{i,j}$. These are established either at the beginning of the algorithm, or by the preceding call $C_{i,j-1}$. In particular, we define datasets I', I'', and J' and prove that they satisfy the following properties:

$$R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^m] \subseteq R\text{-part}[I''] \cup J' \subseteq M_\infty[X_{i,j-1}^R] \subseteq R\text{-part}[I'] \cup J' \subseteq \Pi_\infty[E_i]^s \tag{B.9}$$
$$R\text{-part}[I'] \cup J' \subseteq (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \tag{B.10}$$

Towards this goal, call $C_{i,j}$ can be of one of the following three types.

- Assume that $C_{i,j}$ is of type A1. We let $I' = I'' = J' = \emptyset$, and it is obvious that properties (B.9) and (B.10) hold.
- Assume that $C_{i,j}$ is of type A2. Thus, $\Delta_{i,j}^- = \emptyset$ and call $C_{i,j-1}$ is of the form $J_{i,j-1} := \text{Add}^M[I_{i,j-1} \vdots \Delta_{i,j-1}^-, \Delta_{i,j-1}^+ \vdots \Delta_{i,j-1}^m]$. We let $J' = J_{i,j-1}$ and $I' = I'' = I_{i,j} = (I_{i,j-1} \setminus \Delta_{i,j-1}^-) \cup \Delta_{i,j-1}^+$, where the last equality holds by condition (A2.b) from Table 2. Property (A2.c) ensures $J' = \Delta_{i,j}^m$, which with $I'' = I_{i,j}$ and $\Delta_{i,j}^- = \emptyset$ ensures the left-most inclusion of property (B.9). Moreover, condition (B.2) holds for i and $j - 1$ by the induction assumption, so the remaining inclusions

of property (B.9) hold as well. Finally, $J' \subseteq \Delta_{i,j}^+$ holds by property (A2.c), which together with $I' = I_{i,j}$ and $\Delta_{i,j}^- = \emptyset$ ensures property (B.10).

- Assume that $C_{i,j}$ is of type A3. Thus, call $C_{i,j-1}$ is of the form $J_{i,j-1} := \text{Red}^M[I_{i,j-1}^o, I_{i,j-1}^n : \Delta_{i,j-1}]$. We let $J' = J_{i,j-1}$, we let $I'' = I_{i,j-1}^n$, and we let $I' = I_{i,j-1}^o \setminus \Delta_{i,j-1}$. Properties (A3.b) and (A3.d), and $M \subseteq \Pi^s$ ensure $R\text{-part}[I_{i,j} \setminus \Delta_{i,j}^-] \subseteq R\text{-part}[I'']$; and property (A3.f) ensures $\Delta_{i,j}^m = J'$; together, these ensure the left-most inclusion of property (B.9). Moreover, condition (B.7) holds for i and $j-1$ by the induction assumption, so it clearly ensures the remaining inclusions property (B.9). Finally, conditions (R.f) and (A3.d) jointly ensure $\Delta_{i,j}^- \cap O^s = \Delta_{i,j-1} \cup ((E_i \cap O^s) \setminus I_{i,j-1}^n)$; and condition (A3.f) ensures $(E_i \cap O^s) \setminus I_{i,j-1}^n \subseteq \Delta_{i,j}^+$; jointly, these observations clearly imply property (B.10).

We are now ready to prove that call $C_{i,j}$ satisfies properties (T1) and (T4). In particular, property (B.3) holds trivially: it is established either at the beginning of the history or by the preceding call, and Algorithm 4 does not modify set Y^R.

Inclusions $R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j} \subseteq M_\infty[X_{i,j}^R]$ and $R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j} \subseteq \Pi_\infty[E_i]^s$ in property (B.2) can be shown from the first and second inclusion in property (B.9) by a straightforward induction on the steps of Algorithm 4. Roughly speaking, for each fact $R(u, w)$ produced by the algorithm, there are facts $R(u, v)$ and $R(v, w)$ that have been produced previously and that satisfy these inclusions; hence, $R(u, w)$ is both in the transitive closure of $X_{i,j}^R$ and in $\Pi_\infty[E_i]^s$. The proof is routine and we omit the details for the sake of brevity.

The remaining part of property (B.2) involves proving inclusion $M_\infty[X_{i,j}^R] \subseteq R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j}$. For an arbitrary fact $R(u, w) \in M_\infty[X_{i,j}^R]$, if $R(u, w) \in M_\infty[X_{i,j-1}^R]$ holds, then property (B.9) ensures $R(u, w) \in R\text{-part}[I'] \cup J'$, so property (B.10) ensures $R(u, w) \in (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, as required. To complete the proof, we next show that, for all constants u and w and integer ℓ such that $u \leadsto^\ell w \in X_{i,j}^R$ and $R(u, w) \notin M_\infty[X_{i,j-1}^R]$, we have

($\Diamond$) $R(u, w) \in R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j}$, and
($\blacklozenge$) fact $R(u, w)$ is added to the set Q at some point during the call $C_{i,j}$ of Algorithm 4.

Note that $R(u, w) \notin M_\infty[X_{i,j-1}^R]$ implies $R(u, w) \notin R\text{-part}[I''] \cup J'$ by the first inclusion of (B.9), which, by the left-most inclusion of (B.9), implies $R(u, w) \notin R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^m]$. Property (B.2) then holds because $R(u, w) \in M_\infty[X_{i,j}^R]$ implies that there exists ℓ such that $u \leadsto^\ell w \in X_{i,j}^R$. We prove ($\Diamond$) and ($\blacklozenge$) by induction on ℓ.

- For the base case, consider arbitrary constants u and w such that $u \leadsto^1 w \in X_{i,j}^R$ and $R(u, w) \notin M_\infty[X_{i,j-1}^R]$. Thus, fact $R(u, w)$ was added to set X^R during the algorithm's run, which is only possible in line 50. But then, $R(u, w) \in \Delta'$ implies $R(u, w) \in \Delta^+ \setminus \Delta^m$, which is clearly sufficient for ($\Diamond$); moreover, fact $R(u, w)$ is clearly added to Q in line 49, so ($\blacklozenge$) holds.
- For the inductive step, assume that properties ($\Diamond$) and ($\blacklozenge$) hold for $\ell - 1$, and consider arbitrary constants u and w such that $u \leadsto^\ell w \in X_{i,j}^R$ and $R(u, w) \notin M_\infty[X_{i,j-1}^R]$. Clearly, there exists a constant v such that $u \leadsto^1 v \in X_{i,j}^R$ and $v \leadsto^{\ell-1} w \in X_{i,j}^R$ hold. We have the following possibilities.

  – Assume $R(v, w) \in M_\infty[X_{i,j-1}^R] \setminus (\Delta_{i,j}^+ \setminus \Delta_{i,j}^m)$. Thus, we have $R(v, w) \in M_\infty[X_{i,j-1}^R]$ and $R(v, w) \notin \Delta_{i,j}^+ \setminus \Delta_{i,j}^m$. By the above observation, the former ensures $R(v, w) \in (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$. Moreover, if $R(v, w) \in \Delta_{i,j}^+$, then $R(v, w) \notin \Delta_{i,j}^+ \setminus \Delta_{i,j}^m$ ensures $R(v, w) \in \Delta_{i,j}^m$. Consequently, we have $R(v, w) \in (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^m$. Furthermore, $R(v, w) \in M_\infty[X_{i,j-1}^R]$ and $R(u, w) \notin M_\infty[X_{i,j-1}^R]$ ensure $R(u, v) \in X_{i,j}^R \setminus X_{i,j-1}^R$. In other words, $R(u, v)$ is added to X^R during the call, which is possible only if $R(u, v) \in \Delta'$. Thus, facts $R(u, v)$ and $R(v, w)$ are considered in line 52.
  – Assume $R(v, w) \notin M_\infty[X_{i,j-1}^R] \setminus (\Delta_{i,j}^+ \setminus \Delta_{i,j}^m)$. Then, either $R(v, w) \in M_\infty[X_{i,j-1}^R] \cap (\Delta_{i,j}^+ \setminus \Delta_{i,j}^m)$ or $R(v, w) \notin M_\infty[X_{i,j-1}^R]$. In the former case, $R(v, w)$ is added to Q in line 49; and in the latter case, fact $R(v, w)$ is added to Q since the inductive assumption holds for $\ell - 1$. Since $R(u, v) \in X_{i,j}^R$, facts $R(u, v)$ and $R(v, w)$ are considered in lines 55 and 56.

  Either way, fact $R(u, w)$ is considered either in line 53 or in line 57. Now if $R(u, w) \in \Delta_{i,j}^+ \setminus \Delta_{i,j}^m$, then ($\Diamond$) clearly holds, and $R(u, w)$ is in added to Q in line 49 so ($\blacklozenge$) holds as well. Otherwise, we have $R(u, w) \notin R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^m]$ by the left-most inclusion of property (B.9), so $R(u, w)$ is added to Q and J in line 53 or 57.

This completes our proof of property (B.2).

Finally, we prove (T4)—that is, that call $C_{i,j}$ is correct. For the upper bound, consider an arbitrary fact $R(u, v) \in J_{i,j}$; then, property (B.2) ensures $R(u, v) \in \Pi_\infty[E_i]^s$, and lines 53 and 57 ensure $R(u, v) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, as required. To see that

$J_{i,j}$ also satisfies the lower bound, consider arbitrary $F \in M_A[I_{i,j} : \Delta_{i,j}^-, \Delta_{i,j}^+]$. By Definition 8, we have $F \in M\big[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+\big]$ and $F \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$. The former and the equality in property (B.2) jointly imply $F \in M\big[M_\infty[X_{i,j}^R]\big] \subseteq M_\infty[X_{i,j}^R]$; thus, property (B.2) ensures $F \in R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j}$. But then, $F \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ clearly ensures $F \in J_{i,j}$, as required. $\square$

Claim 37. *If $C_{i,j}$ is of the form $J_{i,j} := \text{Del}^M[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}^-, \Delta_{i,j}^+ : \Delta_{i,j}^m]$ and*

- *$i \geq 1$, $j = 1$, and call $C_{i-1,h_{i-1}}$ satisfies properties (T1)–(T4), or*
- *$j > 1$ and call $C_{i,j-1}$ satisfies properties (T1)–(T4),*

then $C_{i,j}$ satisfies properties (T2) and (T4).

Proof. We first capture the preconditions for the call $C_{i,j}$ by proving the following properties. Please remember that, at the beginning of this section, we defined $X_{i,0}^R = X_{i-1,h_{i-1}}^R$ and $Y_{i,0}^R = Y_{i-1,h_{i-1}}^R$ for each $i > 0$.

$$\text{close}[X_{i,0}^R, I_{i,j}^o \setminus I_{i,j}^n] = R\text{-part}[I_{i,j}^o \setminus I_{i,j}^n] \cup \Delta_{i,j}^m \cup Y_{i,j-1}^R \tag{B.11}$$

$$X_{i,j-1}^R = X_{i,0}^R \setminus (I_{i,j}^o \setminus I_{i,j}^n) \subseteq R\text{-part}[I_{i,j}^n] \subseteq R\text{-part}[I_{i,j}^o] = M_\infty[X_{i,0}^R] \tag{B.12}$$

$$Y_{i,j-1}^R \subseteq I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s} \tag{B.13}$$

Towards this goal, call $C_{i,j}$ can be of one of the following two types.

- Assume that $C_{i,j}$ is of type D1. Condition (D1.a) and property (B.3) for $C_{i-1,h_{i-1}}$ ensure $Y_{i,j-1}^R = \emptyset$, as required for property (B.13). Moreover, condition (D1.d) ensures $I_{i,j}^o \setminus I_{i,j}^n = \emptyset$, which, together with $\Delta_{i,j}^m = \emptyset$ and $Y_{i,j-1}^R = \emptyset$, clearly ensures (B.11). Finally, property (B.2) for call $C_{i-1,h_{i-1}}$ and conditions (D1.b) and (D1.c) jointly ensure $R\text{-part}[I_{i,j}^o] = M_\infty[X_{i,0}^R]$, which, together with $I_{i,j}^o \setminus I_{i,j}^n = \emptyset$, clearly ensures property (B.12).

- Assume that $C_{i,j}$ is of type D2. Condition (D2.c) ensures $R\text{-part}[I_{i,j}^o \setminus I_{i,j}^n] = R\text{-part}[I_{i,j}^o \setminus ((I_{i,j-1}^n \setminus \Delta_{i,j-1}^-) \cup \Delta_{i,j-1}^+)]$; set $\Delta_{i,j-1}^+$ contains no fact from stratum s, and so $M \subseteq \Pi^s$ ensures $R\text{-part}[I_{i,j}^o \setminus I_{i,j}^n] = R\text{-part}[I_{i,j}^o \setminus (I_{i,j-1}^n \setminus \Delta_{i,j-1}^-)]$. Call history H is compatible with Π, λ, s, and $E_0, \ldots, E_m$, so Definition 11 ensures $\Delta_{i,j-1}^- \subseteq I_{i,j-1}^n$; property (B.5) holds for i and $j-1$ by the inductive assumption; and conditions (D1.d) and (D2.b) ensure $I_{i,j}^o = I_{i,j-1}^o$; together, all of these observations ensure $R\text{-part}[I_{i,j}^o \setminus I_{i,j}^n] = R\text{-part}[(I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^-]$. Together with condition (D2.d) and property (B.4) for i and $j-1$, this ensures (B.11). Moreover, condition (D1.d) or (D2.b) (depending on the type of the previous call) ensures $I_{i,j-1}^o = \Pi_\infty[E_{i-1}]$, and property (D2.b) ensures $I_{i,j}^o = \Pi_\infty[E_{i-1}]$; thus, $I_{i,j-1}^o = I_{i,j}^o$; but then, since property (B.6) holds for i and $j-1$ by the induction assumption, this implies property (B.13). Finally, property (B.5) holds for i and $j-1$, and so we have $X_{i,j-1}^R = X_{i,0}^R \setminus ((I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^-)$; together with conditions (D2.b) and (D2.c), and the fact that $\Delta_{i,j-1}^+$ contains no R-facts, this implies $X_{i,j-1}^R = X_{i,0}^R \setminus (I_{i,j}^o \setminus I_{i,j}^n)$; in addition, $X_{i,j-1}^R = X_{i,0}^R \setminus ((I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^-)$ clearly ensures that $X_{i,j-1}^R$ is disjoint from $\Delta_{i,j-1}^-$, which together with $X_{i,j-1}^R \subseteq R\text{-part}[I_{i,j-1}^n]$ and condition (D2.c) ensures $X_{i,j-1}^R \subseteq R\text{-part}[I_{i,j}^n]$; moreover, $R\text{-part}[I_{i,j}^n] \subseteq R\text{-part}[I_{i,j}^o]$ follows from the same property for i and $j-1$, conditions (D2.b) and (D2.c), and the fact that $\Delta_{i,j-1}^+$ contains no R-facts; finally, the rightmost equality of property (B.12) follows from the same property for i and $j-1$ in property (B.5) and condition (D2.b). Hence, (B.12) holds.

Next we prove that call $C_{i,j}$ satisfies properties (T2) and (T4).

For the $\subseteq$ direction of property (B.4), we consider an arbitrary fact $R(u, v) \in \text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$, and we next prove $R(u, v) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j}^R$. By the definition of $\text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$ in equation (B.1), there exist constants $w_0, w_1, \ldots, w_\ell$ with $u = w_0$ and $v = w_\ell$, and also there exists an index k with $0 \leq k < \ell$ such that $R(w_k, w_{k+1}) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-$ and $R(w_m, w_{m+1}) \in X_{i,0}^R$ for $m \neq k$. Now $R(w_k, w_{k+1}) \in I_{i,j}^o \setminus I_{i,j}^n$ clearly implies $R(w_k, w_{k+1}) \in I_{i,j}^o$. Otherwise, $R(w_k, w_{k+1}) \in \Delta_{i,j}^-$ holds; call history H is compatible with Π, λ, s, and $E_0, \ldots, E_m$ so Definition 11 ensures $\Delta_{i,j}^- \subseteq I_{i,j}^n$; and property (B.12) ensures $R\text{-part}[I_{i,j}^n] \subseteq R\text{-part}[I_{i,j}^o]$; hence, $R(w_k, w_{k+1}) \in I_{i,j}^o$ holds as well. But then, property (B.12) ensures $R(w_k, w_{k+1}) \in M_\infty[X_{i,0}^R]$. In other words, for each m with $0 \leq m \leq \ell$, we have $u \rightsquigarrow w_m \in X_{i,0}^R$ and $w_m \rightsquigarrow v \in X_{i,0}^R$. We next consider the following two cases.

- There exist (not necessarily consecutive) m and n with $0 \leq m < n \leq \ell$ such that $R(w_m, w_n) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^m \cup Y_{i,j-1}^R$. Then, the $\supseteq$ direction of property (B.11) ensures $R(w_m, w_n) \in \text{close}[X_{i,0}^R, I_{i,j}^o \setminus I_{i,j}^n]$, which, together with $u \rightsquigarrow w_m \in X_{i,0}^R$ and $w_n \rightsquigarrow v \in X_{i,0}^R$ established earlier, ensures $R(u, v) \in \text{close}[X_{i,0}^R, I_{i,j}^o \setminus I_{i,j}^n]$. The $\subseteq$ direction of (B.11) then ensures

$R(u, v) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^m \cup Y_{i,j-1}^R$; but then, Definition 13 ensures $\Delta_{i,j}^m \subseteq \Delta_{i,j}^-$, and $Y_{i,j-1}^R \subseteq Y_{i,j}^R$ clearly holds, all of which together ensure $R(u, v) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j}^R$, as required.

- For all integers m and n with $0 \le m < n \le \ell$, we have $R(w_m, w_n) \notin (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^m \cup Y_{i,j-1}^R$. Thus, we have

$$R(w_k, w_{k+1}) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^- \text{ and } R(w_k, w_{k+1}) \notin (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^m \cup Y_{i,j-1}^R,$$

which imply $R(w_k, w_{k+1}) \in \Delta_{i,j}^- \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$; consequently, $R(w_k, w_{k+1}) \in \Delta'$ holds after line 60 of Algorithm 5. We next show by induction on s from k to 0 that

$$R(w_s, v) \in (\Delta_{i,j}^- \setminus \Delta_{i,j}^m) \cup J_{i,j} \cup (Y_{i,j}^R \setminus Y_{i,j-1}^R) \tag{B.14}$$

holds for each $0 \le s \le k$, and that fact $R(w_s, v)$ is added to set Q at some point during the execution of Algorithm 5 in call $C_{i,j}$. Then, (B.14) for $s = 0$ clearly implies the $\subseteq$ direction of property (B.4).

In the base case $s = k$, we have two possibilities. If $k + 1 = \ell$, then $R(w_s, v) = R(k, k_{k+1})$; thus, fact $R(w_s, v)$ is added to Q in line 60, and moreover $R(w_k, w_{k+1}) \in \Delta_{i,j}^- \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$ implies (B.14) for $s = k$. In the rest of this proof, we assume $k + 1 < \ell$. Then, $w_{k+1} \rightsquigarrow v \in X_{i,0}^R$ and property (B.12) ensure $R(w_{k+1}, v) \in I_{i,j}^o$; by $R(w_{k+1}, v) \notin (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^m \cup Y_{i,j-1}^R$, we have $R(w_{k+1}, v) \in I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$; hence, $R(w_{k+1}, v) \in I'$ holds in line 60. Thus, $R(w_k, w_{k+1}) \in \Delta'$ and $R(w_{k+1}, v) \in I'$ are considered in line 62. Now $R(w_k, v) \notin (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^m \cup Y_{i,j-1}^R$ holds by our assumption, so $R(w_k, v) \in M_\infty[X_{i,0}^R] \subseteq I_{i,j}^o$ implies $R(w_k, v) \in I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$. Now if fact $R(w_k, v)$ passes the check in line 63, the fact will be added to Q in line 64, and to either J in line 65 or Y^R in line 66; hence, $R(w_k, v) \in J_{i,j} \cup Y_{i,j}^R$, which alongside $R(w_k, v) \notin (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^m \cup Y_{i,j-1}^R$ ensures $R(w_k, v) \in J_{i,j} \cup (Y_{i,j}^R \setminus Y_{i,j-1}^R)$, as required. The only remaining possibility is if condition in line 63 is not satisfied. Let J' and Y' be the values of J and Y^R, respectively, against which fact $R(w_k, v)$ is checked in line 63. Then, $R(w_k, v) \notin I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J' \cup Y')$ and $R(w_k, v) \in I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$ imply $R(w_k, v) \in (\Delta_{i,j}^- \cup J' \cup Y') \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$. Thus, if $R(w_k, v) \in \Delta_{i,j}^- \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$, then fact $R(w_k, v)$ is added to Q in line 59. Moreover, if either $R(w_k, v) \in J' \subseteq J_{i,j}$ or $R(w_k, v) \in Y' \setminus Y_{i,j-1}^R \subseteq Y_{i,j}^R \setminus Y_{i,j-1}^R$, then fact $R(w_k, v)$ is added to Q earlier in line 64. Either way, property (B.14) holds.

For the inductive step, we consider arbitrary $0 \le s < k$ such that property (B.14) holds for $s + 1$ and fact $R(w_{s+1,v})$ is added to Q. Thus, fact $R(w_{s+1}, v)$ is extracted from Q in line 68 at some iteration of the loop in lines 67–73. We have established $R(w_s, w_{s+1}) \in X_{i,0}^R$ earlier, and we have $R(w_s, w_{s+1}) \notin I_{i,j}^o \setminus I_{i,j}^n$ by our assumption; together with property (B.12), we have $R(w_s, w_{s+1}) \in X_{i,j-1}^R$, and so fact $R(w_s, w_{s+1})$ is considered in line 69. Consequently, fact $R(w_s, v)$ is considered in line 70. Now analogously to the base case, we can conclude that fact $R(w_s, v)$ satisfies property (B.14) and that it is added to Q.

This completes our proof for the $\subseteq$ direction of (B.4).

For the $\supseteq$ direction of property (B.4), let N be the number of iterations of the loop in lines 67–73 during the execution of call $C_{i,j}$; moreover, let J^0, Y^0, and Q^0 be the values of J, Y^R, and Q, respectively, before line 67; finally, for each $1 \le n \le N$, let J^n, Y^n, and Q^n be the values of J, Y^R, and Q, respectively, after the n-th iteration of lines 67–73. We prove by induction on n that

$$(I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^- \cup J^n \cup Y^n \cup Q^n \subseteq \text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]. \tag{B.15}$$

Since $J_{i,j} = J^N$ and $Y_{i,j}^R = Y^N$, the $\supseteq$ direction of property (B.4) follows from property (B.15) for $n = N$.

- For the induction base, consider an arbitrary fact $R(u, w) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^- \cup J^n \cup Y^n \cup Q^n$. If $R(u, w) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-$, we have $R(u, w) \in \text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$ trivially by the definition in equation (B.1). If $R(u, w) \in Y_{i,j-1}^R$, then the $\supseteq$ direction of property (B.11) ensures $R(u, w) \in \text{close}[X_{i,0}^R, I_{i,j}^o \setminus I_{i,j}^n] \subseteq \text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$. If $R(u, w)$ is added to Q^0 in line 59, then $R(u, w) \in \Delta_{i,j}^-$ trivially implies the required property. The only remaining possibility is that $R(u, w)$ is added to Q^0 and either J^0 or Y^0 in lines 65–66. Then, there exists a constant v that satisfies the condition in line 62—that is, $R(u, v) \in \Delta_{i,j}^- \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$ and $R(v, w) \in I_{i,j}^n \setminus (\Delta_{i,j}^- \cup Y_{i,j-1}^R)$. The latter observation and property (B.12) ensure $R(v, w) \in M_\infty[X_{i,0}^R]$, so clearly $R(u, w) \in \text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$, as required.
- For the inductive step, we assume that property (B.15) holds for $n - 1$ with $0 < n < N$, and we consider an arbitrary fact $R(u, w) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^- \cup J^n \cup Y^n \cup Q^n$. The only nontrivial case is when fact $R(u, w)$ is added to this set in the n-th iteration. This, in turn, is possible only if $R(u, w)$ is added to J^n, Y^n, or Q^n in lines 72–73 of the n-th iteration. Then, there exists a constant v such that fact $R(v, w)$ is extracted from Q^{n-1} in line 68, and $R(u, v) \in X_{i,j-1}^R$. The former and the inductive assumption ensure $R(v, w) \in \text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$. Moreover, property (B.12) ensures $X_{i,j-1}^R \subseteq X_{i,0}^R$, so $R(u, v) \in X_{i,0}^R$ holds. Clearly, we have $R(u, w) \in \text{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$, as required.

For property (B.6), consider arbitrary $R(u, w) \in Y_{i,j}^R$. If $R(u, w) \in Y_{i,j-1}^R$, then property (B.13) ensures $R(u, w) \in I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s}$. If $R(u, w)$ is added to Y^R in line 66, then line 63 and the R-part$[I_{i,j}^n] \subseteq R$-part$[I_{i,j}^o]$ part of property (B.12) ensure $R(u, w) \in I_{i,j}^o$; moreover, line 65 and Definition 16 ensure $R(u, w) \in \Pi_\infty[E_i]^{\leq s}$; thus, $R(u, w) \in I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s}$. The remaining possibility of $R(u, w)$ being added to Y^R in line 73 is analogous, so we omit the details for the sake of brevity.

For property (B.5), the leftmost equality follows from the leftmost equality in (B.12) and how X^R is updated in line 74; the rest of the property follows immediately from property (B.12).

To see that $J_{i,j}$ satisfies its lower bound, consider an arbitrary fact $R(u, v) \in M_D[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}^-, \Delta_{i,j}^+] \setminus \Pi_\infty[E]$. Let r be the only rule in M. Then, by Definition 9 there exists rule instance $r\sigma$ of r such that $I_{i,j}^o \models \mathsf{b}(r\sigma)$, $I_{i,j}^n \models \mathsf{b}(r\sigma)$, $(I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \not\models \mathsf{b}(r\sigma)$, and $R(u, v) = \mathsf{h}(r\sigma) \in (I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ all hold. $\Delta_{i,j}^+$ contains no R-facts, so we clearly have $I_{i,j}^n \setminus \Delta_{i,j}^- \not\models \mathsf{b}(r\sigma)$ and $R(u, v) \in I_{i,j}^n \setminus \Delta_{i,j}^-$. Let $\mathsf{b}(r\sigma) = \{R(u, w), R(w, v)\}$. Then, $I_{i,j}^n \models \mathsf{b}(r\sigma)$ and $I_{i,j}^n \setminus \Delta_{i,j}^- \not\models \mathsf{b}(r\sigma)$ ensure $\mathsf{b}(r\sigma) \cap \Delta_{i,j}^- \neq \emptyset$. Without loss of generality, assume that $R(u, w) \in \Delta_{i,j}^-$. Together with $I_{i,j}^o \models \mathsf{b}(r\sigma)$ and R-part$[I_{i,j}^o] = M_\infty[X_{i,0}^R]$ from property (B.5), this implies $R(u, v) \in \mathsf{close}[X_{i,0}^R, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$. Then, property (B.4) clearly ensures $R(u, v) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j}^R$. Together with $R(u, v) \in I_{i,j}^n \setminus \Delta_{i,j}^-$, this implies $R(u, v) \in J_{i,j} \cup Y_{i,j}^R$. Moreover, $R(u, v) \notin \Pi_\infty[E]$ and property (B.6) ensure $R(u, v) \notin Y_{i,j}^R$. Hence, $R(u, v) \in J_{i,j}$ holds, as required.

Finally, we show that $J_{i,j}$ satisfies the upper bound. Lines 63 and 70 imply $J_{i,j} \subseteq R$-part$[I_{i,j}^n \setminus \Delta_{i,j}^-]$, which together with property (B.5) implies $J_{i,j} \subseteq I_{i,j}^o$. Moreover, R belongs to stratum s, so $J_{i,j} \subseteq I_{i,j}^o \cap O^s \cap ((I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+)$ holds, as required. $\square$

Claim 38. *If call $C_{i,j}$ is of type* (R) *and call $C_{i,j-1}$ satisfies properties* (T1)–(T4)*, then call $C_{i,j}$ satisfies properties* (T3) *and* (T4).

Proof. For the leftmost inclusion of property (B.7), call $C_{i,j-1}$ involves the Del^M module function so properties (B.4) and (B.5) for i and $j-1$ ensure R-part$[I_{i,j-1}^n] \setminus (\Delta_{i,j-1}^- \cup J_{i,j-1} \cup Y_{i,j-1}^R) \subseteq M_\infty[X_{i,0}^R] \setminus \mathsf{close}[X_{i,0}^R, (I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^-]$. Now consider an arbitrary fact $R(u, v) \in R$-part$[I_{i,j-1}^n] \setminus (\Delta_{i,j-1}^- \cup J_{i,j-1} \cup Y_{i,j-1}^R)$; then, there exists a chain of facts in $X_{i,0}$ that connect u and v, and none of these facts belong to $(I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^-$. But then, property (B.5) for i and $j-1$ ensures that all these facts belong to $X_{i,j-1}$, so we have $R(u, v) \in M_\infty[X_{i,j-1}^R]$. The choice of $R(u, v)$ is arbitrary, so we have R-part$[I_{i,j-1}^n] \setminus (\Delta_{i,j-1}^- \cup J_{i,j-1} \cup Y_{i,j-1}^R) \subseteq M_\infty[X_{i,j-1}^R]$. Together with properties (R.b) and (R.d), and the fact that $\Delta_{i,j-1}^+$ contains no R facts this implies R-part$[I_{i,j}^n] \subseteq Y_{i,j-1}^R \cup M_\infty[X_{i,j-1}^R]$. Line 76 ensures $X_{i,j}^R = X_{i,j-1}^R \cup Y_{i,j-1}^R$, so we have R-part$[I_{i,j}^n] \subseteq M_\infty[X_{i,j}^R]$. Moreover, due to line 79, each fact $R(u, v) \in J_{i,j}$ satisfies $R(u, v) \in M_\infty[X_{i,j}^R]$. Consequently, the required property holds.

For the second inclusion of property (B.7), property (B.5) for i and $j-1$ and the fact that call history H is compatible with Π, λ, s and $E_0, \ldots, E_m$ ensure $X_{i,j-1}^R \subseteq \Pi_\infty[E_{i-1}]^{\leq s}$. Moreover, property (B.6) for i and $j-1$ ensures $Y_{i,j-1}^R \subseteq \Pi_\infty[E_{i-1}]^{\leq s}$. As a result, line 76 ensures $X_{i,j}^R = X_{i,j-1}^R \cup Y_{i,j-1}^R \subseteq \Pi_\infty[E_{i-1}]^{\leq s}$, and so $M_\infty[X_{i,j}^R] \subseteq M_\infty[\Pi_\infty[E_{i-1}]^{\leq s}] \subseteq \Pi_\infty[E_{i-1}]^{\leq s}$ holds. We next prove $M_\infty[X_{i,j}^R] \setminus (R$-part$[I_{i,j}^o] \setminus \Delta_{i,j}) \subseteq J_{i,j}$, which implies our claim. Consider an arbitrary fact $R(u, v) \in M_\infty[X_{i,j}^R] \setminus (R$-part$[I_{i,j}^o] \setminus \Delta_{i,j})$ holds. Then, $M_\infty[X_{i,j}^R] \subseteq \Pi_\infty[E_{i-1}]^{\leq s}$ and property (R.c) ensure $R(u, v) \in I_{i,j}^o$. Together with $R(u, v) \notin R$-part$[I_{i,j}^o] \setminus \Delta_{i,j}$ this implies $R(u, v) \in \Delta_{i,j}$. Consequently, $R(u, v)$ is considered in line 78 during the execution of call $C_{i,j}$. Since $R(u, v) \in M_\infty[X_{i,j}^R]$ holds, v is clearly reachable from u via R facts in $X_{i,j}^R$, and so $R(u, v) \in \Delta_{i,j}$ ensures that $R(u, v)$ is added to J in line 80. The choice of $R(u, v)$ is arbitrary, so the second inclusion of property (B.7) holds.

For the third inclusion of property (B.7), conditions (R.c), (R.e), and (R.f), and $M \subseteq \Pi^s$ ensure R-part$[I_{i,j}^o] \setminus \Delta_{i,j} \subseteq \Pi_\infty[E_i]^s$. Moreover, property (B.5) for i and $j-1$ ensures $X_{i,j-1}^R \subseteq I_{i,j-1}^n \setminus \Delta_{i,j-1}^-$, which together with conditions (R.d) and (R.e), and $M \subseteq \Pi^s$ ensures $X_{i,j-1}^R \subseteq \Pi_\infty[E_i]^s$; in addition, property (B.6) for i and $j-1$ ensures $Y_{i,j-1}^R \subseteq \Pi_\infty[E_i]^s$; hence, line 76 ensures $X_{i,j}^R = X_{i,j-1}^R \cup Y_{i,j-1}^R \subseteq \Pi_\infty[E_i]^s$, which in turn implies $M_\infty[X_{i,j}^R] \subseteq \Pi_\infty[E_i]^s$. Together with the leftmost inclusion of (B.7) this implies $J_{i,j} \subseteq \Pi_\infty[E_i]^s$, as required.

Property (B.8) holds by line 77.

Next we show that $J_{i,j}$ satisfies its lower bound—that is, $M_R[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}] \subseteq J_{i,j}$ holds. To this end, consider arbitrary $F \in M_R[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}]$. Let r be the only rule in M. By Definition 10, there exists rule instance $r\sigma$ such that $F = \mathsf{h}(r\sigma) \in \Delta_{i,j}$, and $I_{i,j}^o \models \mathsf{b}(r\sigma)$, and $I_{i,j}^n \models \mathsf{b}(r\sigma)$ all hold. Without loss of generality, let $\mathsf{b}(r\sigma = \{R(u, v), R(v, w)\})$ and $\mathsf{h}(r\sigma) = R(u, w)$. Then, $I_{i,j}^n \models \mathsf{b}(r\sigma)$ implies $R(u, v) \in I_{i,j}^n$ and $R(v, w) \in I_{i,j}^n$. By property (B.7) we have $R(u, v) \in M_\infty[X_{i,j}^R]$ and $R(v, w) \in M_\infty[X_{i,j}^R]$, and so $R(u, w) \in M_\infty[X_{i,j}^R]$ holds as well. $R(u, w) \in \Delta_{i,j}$ ensures that $R(u, w)$ is considered in line 78 during the execution of $C_{i,j}$, and $R(u, w) \in M_\infty[X_{i,j}^R]$ ensures that $R(u, w)$ is added to J in line 80—that is, $R(u, w) \in J_{i,j}$ holds.

Finally, $J_{i,j} \subseteq \Pi_\infty[E_i]^s$ follows from the rightmost inclusion of (B.7); moreover, line 80 ensures $J_{i,j} \subseteq \Delta_{i,j}$, which together with condition (R.f) ensures $J_{i,j} \cap I_{i,j}^n = \emptyset$; hence, $J_{i,j} \subseteq \Pi_\infty[E_i]^s \setminus I_{i,j}^n$ holds, so $J_{i,j}$ satisfies the upper bound. $\square$

This completes the proof of Theorem 18. To see that Proposition 19 is correct, note that property (B.2) holds even if set X^R is replaced with another set X' of R-facts satisfying $M_\infty[X^R] = M_\infty[X']$.

Appendix C. Proof of Theorem 24

Theorem 24. *Functions* $\mathsf{Add}^{stc(R)}$, $\mathsf{Del}^{stc(R)}$, *and* $\mathsf{Red}^{stc(R)}$ *are correct.*

Consider an arbitrary binary predicate R, program Π, stratification λ of Π, stratum index s such that $stc(R) \subseteq \Pi^s$, sequence of datasets $E_0, \ldots, E_m$, and a call history H for $stc(R)$ that is compatible with Π, λ, and $E_0, \ldots, E_m$. We assume that H is of the form as specified in Definition 12. For each $0 \leq i \leq m$ and each $1 \leq j \leq n_i$, let $X_{i,j}^R$ and $Y_{i,j}^R$ be the values of X^R and Y^R, respectively, after call $C_{i,j}$. We define $X_{0,0}^R = Y_{0,0}^R = \emptyset$ and, for $1 \leq i \leq m$, we let $X_{i,0}^R = X_{i-1,n_{i-1}}^R$ and $Y_{i,0}^R = Y_{i-1,n_{i-1}}^R$.

We next introduce some useful abbreviations and notation. To simplify the notation, let $M = stc(R)$ for the rest of this section. Just like in Appendix B, for S a set of facts, R-part$[S]$ is the subset of S containing precisely all facts of S whose predicate is R. Finally, for X^R a set of sets of constants, we say that X^R is *valid* if the sets in X^R are disjoint, and we let

$$\mathsf{close}[X^R] = \{R(u, v) \mid \exists U \in X^R \text{ such that } u \in U \text{ and } v \in U\} \tag{C.1}$$

Before proceeding with the main proof, we first prove a clam that characterises the CLOSEEDGES function. This will allow us to later simplify the analysis of the Add^M and Red^M functions.

Claim 39. *For each call of the form* $J := \textsc{CloseEdges}(\Delta)$, *let* X_1 *and* X_2 *be the values of* X^R *before and after the call. Then, if* X_1 *is valid, then* X_2 *is also valid and*

$$R\text{-part}[\Delta] \subseteq \mathsf{close}[X_1] \cup J = \mathsf{close}[X_2] \subseteq M_\infty[\mathsf{close}[X_1] \cup R\text{-part}[\Delta]]. \tag{C.2}$$

Proof. Consider an arbitrary execution of the function where X_1 is valid. Notice that X^R can only be updated in line 86, line 87, or line 89; clearly, none of these operations breaks the validity of X^R, so X_2 is valid as well. To see that R-part$[\Delta] \subseteq \mathsf{close}[X_2]$ holds, consider an arbitrary fact $R(u, v) \in R$-part$[\Delta]$. This fact is considered in line 85; moreover, lines 88 and 89 ensure that u and v end up in the same set in X_2, so $R(u, v) \in \mathsf{close}[X_2]$ holds. To prove $\mathsf{close}[X_2] \subseteq \mathsf{close}[X_1] \cup J$, consider an arbitrary fact $R(u, v) \in \mathsf{close}[X_2] \setminus \mathsf{close}[X_1]$. If $u = v$, then u appears in X_2 but not in X_1, so u is introduced into X_2 in either line 86 or line 87; either way, $R(u, v) \in J$ holds. If $u \neq v$, then u and v are put into the same set in line 89, so line 90 ensures $R(u, v) \in J$. To prove $\mathsf{close}[X_1] \cup J \subseteq \mathsf{close}[X_2]$, note that $\mathsf{close}[X_1] \subseteq \mathsf{close}[X_2]$ holds trivially since each set in X^R only grows during the execution of the call; moreover, for each fact $R(u, v)$ added to J in lines 86, 87, or 90, constants u and v belong to the same set in X_2, which clearly ensures $R(u, v) \in \mathsf{close}[X_2]$. Finally, $\mathsf{close}[X_2] \subseteq M_\infty[\mathsf{close}[X_1] \cup R\text{-part}[\Delta]]$ holds intuitively because the rules in M axiomatise R as symmetric and transitive; as a consequence, predicate R is also reflexive on every constant occurring in X_2. Thus, R corresponds to an equivalence relation whose equivalence classes are contained in X_2. This intuition can be proved formally by a simple induction on the steps of function CLOSEEDGES; the proof is routine but somewhat verbose, so we omit the details for the sake of brevity. $\quad\square$

We are now ready to prove Theorem 24. To this end, we show that, for each $0 \leq i \leq m$ and each $1 \leq j \leq h_i$, call $C_{i,j}$ in H satisfies properties (S1)–(S4).

(S1) If call $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Add}^M[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^{\mathsf{m}}]$, then the following properties hold.

$$R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j} = \mathsf{close}[X_{i,j}^R] \subseteq \Pi_\infty[E_i]^s \tag{C.3}$$

$$Y_{i,j}^R = \emptyset \tag{C.4}$$

(S2) If call $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Del}^M[I_{i,j}^{\mathsf{o}}, I_{i,j}^{\mathsf{n}} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^{\mathsf{m}}]$, then the following properties hold.

$$R\text{-part}[I_{i,j}^{\mathsf{n}}] \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j}^R) = \mathsf{close}[X_{i,j}^R] \subseteq R\text{-part}[I_{i,j}^{\mathsf{n}}] \subseteq R\text{-part}[I_{i,j}^{\mathsf{o}}] \tag{C.5}$$

$$Y_{i,j}^R \subseteq I_{i,j}^{\mathsf{o}} \cap \Pi_\infty[E_i]^{\leq s} \tag{C.6}$$

(S3) If call $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Red}^M[I_{i,j}^{\mathsf{o}}, I_{i,j}^{\mathsf{n}} \vdots \Delta_{i,j}]$, then the following properties hold.

$$R\text{-part}[I_{i,j}^{\mathsf{n}}] \cup J_{i,j} \subseteq \mathsf{close}[X_{i,j}^R] \subseteq (R\text{-part}[I_{i,j}^{\mathsf{o}}] \setminus \Delta_{i,j}) \cup J_{i,j} \subseteq \Pi_\infty[E_i]^s \tag{C.7}$$

$$Y_{i,j}^R = \emptyset \tag{C.8}$$

(S4) Call $C_{i,j}$ is correct, and set $X_{i,j}^R$ is valid.

We prove this by double induction on i and j, where we consider each call $C_{i,j}$ of type from Table 2. The base case involves a call of type A1, and the inductive step involves all remaining calls. We split the proof into a separate claim for each module function.

Claim 40. *If $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Add}^M[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^m]$ and either $i = 0$ and $j = 1$, or call $C_{i,j-1}$ satisfies properties (S1)–(S4), then $C_{i,j}$ satisfies properties (S1) and (S4).*

Proof. We first capture the preconditions for the call $C_{i,j}$. These are established either at the beginning of the algorithm, or by the preceding call $C_{i,j-1}$. In particular, we show that set $X_{i,j-1}^R$ is valid; moreover, we define datasets I', I'', and J' and prove that they satisfy the following properties:

$$R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^m] \subseteq R\text{-part}[I''] \cup J' \subseteq \text{close}[X_{i,j-1}^R] \subseteq R\text{-part}[I'] \cup J' \subseteq \Pi_\infty[E_i]^s \tag{C.9}$$

$$R\text{-part}[I'] \cup J' \subseteq (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \tag{C.10}$$

Towards this goal, call $C_{i,j}$ can be of one of the following three types.

- Assume that $C_{i,j}$ is of type A1. Then, $X_{i,j-1}^R = \emptyset$ so set $X_{i,j-1}^R$ is clearly valid; moreover, we let $I' = I'' = J' = \emptyset$, and it is obvious that properties (C.9) and (C.10) hold.
- Assume that $C_{i,j}$ is of type A2. Thus, $\Delta_{i,j}^- = \emptyset$ and call $C_{i,j-1}$ is of the form $J_{i,j-1} := \mathsf{Add}^M[I_{i,j-1} \vdots \Delta_{i,j-1}^-, \Delta_{i,j-1}^+ \vdots \Delta_{i,j-1}^m]$. Property (S4) holds for i and $j-1$ by the inductive assumption, so set $X_{i,j-1}^R$ is valid. We let $J' = J_{i,j-1}$, and we let $I' = I'' = I_{i,j} = (I_{i,j-1} \setminus \Delta_{i,j-1}^-) \cup \Delta_{i,j-1}^+$, where the last equality holds by condition (A2.b) from Table 2. Property (A2.c) ensures $J' = \Delta_{i,j}^m$, which with $I'' = I_{i,j}$ and $\Delta_{i,j}^- = \emptyset$ ensures the left-most inclusion of property (C.9). Moreover, condition (C.3) holds for i and $j-1$ by the induction assumption, so the remaining inclusions of property (C.9) hold as well. Finally, $J' \subseteq \Delta_{i,j}^+$ holds by property (A2.c), which together with $I' = I_{i,j}$ and $\Delta_{i,j}^- = \emptyset$ ensures property (C.10).
- Assume that $C_{i,j}$ is of type A3. Thus, call $C_{i,j-1}$ is of the form $J_{i,j-1} := \mathsf{Red}^M[I_{i,j-1}^o, I_{i,j-1}^n \vdots \Delta_{i,j-1}]$. Property (S4) holds for i and $j-1$ by the inductive assumption, so set $X_{i,j-1}^R$ is valid. We let $J' = J_{i,j-1}$, we let $I'' = I_{i,j-1}^n$, and we let $I' = I_{i,j-1}^o \setminus \Delta_{i,j-1}$. Properties (A3.b) and (A3.d), and $M \subseteq \Pi^s$ ensure $R\text{-part}[I_{i,j} \setminus \Delta_{i,j}^-] \subseteq R\text{-part}[I'']$; and property (A3.f) ensures $\Delta_{i,j}^m = J'$; together, these ensure the left-most inclusion of property (C.9). Moreover, condition (C.7) holds for i and $j-1$ by the induction assumption, so it clearly ensures the remaining inclusions of property (C.9). Finally, conditions (R.f) and (A3.d) jointly ensure $\Delta_{i,j}^- \cap O^s = \Delta_{i,j-1} \cup ((E_i \cap O^s) \setminus I_{i,j-1}^n)$; and condition (A3.f) ensures $(E_i \cap O^s) \setminus I_{i,j-1}^n \subseteq \Delta_{i,j}^+$; jointly, these observations clearly imply property (C.10).

This completes our proof for the preconditions for $C_{i,j}$, and we are now ready to show that $C_{i,j}$ satisfies (S1) and (S4).

For the $\subseteq$ direction of the equality in property (C.3), consider an arbitrary fact $R(u,v) \in R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j}$. If $R(u,v) \in R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^m]$, then by property (C.9) we have $R(u,v) \in \text{close}[X_{i,j-1}^R]$; since $X_{i,j-1}^R$ and $X_{i,j}^R$ correspond to the values of X^R before and after the execution of the CLOSEEDGES call made in line 82 during the execution of call $C_{i,j}$, Claim 39 ensures $\text{close}[X_{i,j-1}^R] \subseteq \text{close}[X_{i,j}^R]$; hence, $R(u,v) \in \text{close}[X_{i,j}^R]$ holds, as required. If $R(u,v) \notin R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^m]$, then $R(u,v) \in R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j}$ ensures $R(u,v) \in R\text{-part}[\Delta_{i,j}^+ \setminus \Delta_{i,j}^m] \cup J_{i,j}$; set $\Delta_{i,j}^+ \setminus \Delta_{i,j}^m$ is the argument of the call to function CLOSEEDGES in line 82, and $J_{i,j}$ is contained in the call's result, so Claim 39 ensures $R\text{-part}[\Delta_{i,j}^+ \setminus \Delta_{i,j}^m] \cup J_{i,j} \subseteq \text{close}[X_{i,j}^R]$; hence, $R(u,v) \in \text{close}[X_{i,j}^R]$ holds in this case as well.

For the $\supseteq$ direction of the equality in property (C.3), we prove $\text{close}[X_{i,j}^R] \setminus R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \subseteq J_{i,j}$. To this end, we consider an arbitrary fact $R(u,v) \in \text{close}[X_{i,j}^R] \setminus R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+]$. Then, $R(u,v) \notin R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+]$ and properties (C.9) and (C.10) jointly ensure $R(u,v) \notin \text{close}[X_{i,j-1}^R]$; together with $R(u,v) \in \text{close}[X_{i,j}^R]$ and Claim 39, this ensures that $R(u,v)$ is contained in the result of the call to CLOSEEDGES in line 82. Finally, $R(u,v) \notin R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+]$ ensures that $R(u,v)$ is not removed from the result in line 82. Consequently, we have $R(u,v) \in J_{i,j}$, as required.

For the rightmost $\subseteq$ of property (C.3), line 82 and Claim 39 ensure $\text{close}[X_{i,j}^R] \subseteq M_\infty[\text{close}[X_{i,j-1}^R] \cup R\text{-part}[\Delta_{i,j}^+ \setminus \Delta_{i,j}^m]]$. Property (C.9) ensures $\text{close}[X_{i,j-1}^R] \subseteq \Pi_\infty[E_i]^s$; moreover, condition (A1.a), (A2.c), or (A3.f), depending on the type of $C_{i,j}$, ensures $R\text{-part}[\Delta_{i,j}^+ \setminus \Delta_{i,j}^m] \subseteq \Pi_\infty[E_i]^s$. Hence, we have $\text{close}[X_{i,j}^R] \subseteq M_\infty[\Pi_\infty[E]^s] \subseteq \Pi_\infty[E]^s$, as required.

Property (C.4) holds trivially: it is established either at the beginning of the history or by the preceding call, and Algorithm 7 does not modify set Y^R.

Finally, we show that property (S4) holds. We have already established that $X_{i,j-1}^R$ is valid; furthermore, X^R is updated in line 82 of call $C_{i,j}$, so Claim 39 ensures that $X_{i,j}^R$ is also valid. Property (C.3) and line 82 ensure that $J_{i,j}$ satisfies the upper bound. To see that $J_{i,j}$ also satisfies the lower bound, consider an arbitrary fact $R(u,w) \in M_A[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+]$. Thus,

there exists a rule instance $r\sigma$ with $r \in M$ such that $\mathsf{b}(r\sigma) \nsubseteq I_{i,j}$, and $\mathsf{b}(r\sigma) \subseteq (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, and $R(u,w) = \mathsf{h}(r\sigma) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ all hold. We have the following possibilities for the choice of r.

- Rule $r \in M$ is of the form (30). Then, $R(u,w) = \mathsf{h}(r\sigma)$ implies $\mathsf{b}(r\sigma) = \{R(w,u)\} \subseteq (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$. Thus, property (C.3) ensures $R(w,u) \in \mathsf{close}[X_{i,j}^R]$, and so w and u belong to the same set in $X_{i,j}^R$. Hence, we have $R(u,w) \in \mathsf{close}[X_{i,j}^R]$; thus, property (C.3) ensures $R(u,w) \in R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j}$; finally $R(u,w) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ ensures $R(u,w) \in J_{i,j}$.

- Rule $r \in M$ is of the form (31). Then, there exists a constant v such that $\mathsf{b}(r\sigma) = \{R(u,v), R(v,w)\} \subseteq (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$. Analogously to above, u, v, and w belong to the same set in $X_{i,j}^R$. Hence, $R(u,w) \in \mathsf{close}[C_R^{i,j}]$; thus, property (C.3) ensures $R(u,w) \in R\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cup J_{i,j}$; finally, $R(u,w) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ ensures $R(u,w) \in J_{i,j}$. □

Claim 41. *If $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Del}^M[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^m]$ and*

- *$i \geq 1$, $j = 1$, and call $C_{i-1,h_{i-1}}$ satisfies properties (S1)–(S4), or*
- *$j > 1$ and call $C_{i,j-1}$ satisfies properties (S1)–(S4),*

then $C_{i,j}$ satisfies properties (S2) and (S4).

Proof. We first capture the preconditions for the call $C_{i,j}$. Please remember that, at the beginning of this section, we defined $X_{i,0}^R = X_{i-1,h_{i-1}}^R$ and $Y_{i,0}^R = Y_{i-1,h_{i-1}}^R$ for each $i > 0$. Set $X_{i,j-1}^R$ is valid by property (S4). We next prove the following.

$$R\text{-part}[I_{i,j}^n] \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R) = \mathsf{close}[X_{i,j-1}^R] \subseteq R\text{-part}[I_{i,j}^n] \subseteq R\text{-part}[I_{i,j}^o] \tag{C.11}$$

$$Y_{i,j-1}^R \subseteq I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s} \tag{C.12}$$

Inclusion $R\text{-part}[I_{i,j}^n] \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R) \subseteq R\text{-part}[I_{i,j}^n]$ holds trivially. Call $C_{i,j}$ can be of one of the following two types.

- Assume that $C_{i,j}$ is of type D1. Condition (D1.a) and property (C.4) for $C_{i-1,h_{i-1}}$ ensure $Y_{i,j-1}^R = \emptyset$, as required for property (C.12). Moreover, condition (D1.d) ensures the rightmost inclusion of (C.11). Furthermore, conditions (D1.c) and (D1.d) jointly ensure $R\text{-part}[I_{i,j}^n] = R\text{-part}[(I_{i-1,h_{i-1}} \setminus \Delta_{i-1,h_{i-1}}^-) \cup \Delta_{i-1,h_{i-1}}^+]$, which, together with $\Delta_{i,j}^m = Y_{i,j-1}^R = \emptyset$, property (C.3) for $i-1$ and h_{i-1}, and condition (D1.b), clearly ensures the equality in (C.11).

- Assume that $C_{i,j}$ is of type D2. Condition (D2.c) ensures $R\text{-part}[I_{i,j}^n] = R\text{-part}[(I_{i,j-1}^n \setminus \Delta_{i,j-1}^-) \cup \Delta_{i,j-1}^+]$; set $\Delta_{i,j-1}^+$ contains no fact from stratum s, and so $M \subseteq \Pi^s$ ensures $R\text{-part}[I_{i,j}^n] = R\text{-part}[I_{i,j-1}^n \setminus \Delta_{i,j-1}^-]$. Together with condition (D2.d) and property (C.5) for i and $j-1$, this ensures the equality in (C.11). For the rightmost inclusion of (C.11), condition (D1.d) or (D2.b) (depending on the type of the previous call) ensures $I_{i,j-1}^o = \Pi_\infty[E_{i-1}]$, and property (D2.b) ensures $I_{i,j}^o = \Pi_\infty[E_{i-1}]$; thus, $I_{i,j-1}^o = I_{i,j}^o$; but then, $R\text{-part}[I_{i,j}^n] = R\text{-part}[I_{i,j-1}^n \setminus \Delta_{i,j-1}^-]$ and the rightmost inclusion of property (C.5) for i and $j-1$ jointly ensure $R\text{-part}[I_{i,j}^n] \subseteq R\text{-part}[I_{i,j}^o]$, as required. Finally, property (C.12) follows from property (C.6) for i and $j-1$, and $I_{i,j-1}^o = I_{i,j}^o$.

This completes our proof for the preconditions for $C_{i,j}$, and we are now ready to show that $C_{i,j}$ satisfies (S2) and (S4).

For the $\subseteq$ direction of the equality in (C.5), we prove $R\text{-part}[I_{i,j}^n] \setminus (\Delta_{i,j}^- \cup Y_{i,j-1}^R \cup \mathsf{close}[X_{i,j}^R]) \subseteq J_{i,j} \cup Y_{i,j}^R$; together with $Y_{i,j-1}^R \subseteq Y_{i,j}^R$, this ensures the required property. Consider an arbitrary fact $R(u,v) \in R\text{-part}[I_{i,j}^n] \setminus (\Delta_{i,j}^- \cup Y_{i,j-1}^R \cup \mathsf{close}[X_{i,j}^R])$. Then, $\Delta_{i,j}^m \subseteq \Delta_{i,j}^-$ and property (C.11) ensure $R(u,v) \in \mathsf{close}[X_{i,j-1}^R]$; together with $R(u,v) \notin \mathsf{close}[X_{i,j}^R]$, this ensures that u and v belong to a set that is removed from X^R in line 97 during the execution of $C_{i,j}$. But then, lines 94–96 ensure that $R(u,v)$ is added to either J in line 95 or Y^R in line 96; in the former case, $R(u,v) \in R\text{-part}[I_{i,j}^n] \setminus \Delta_{i,j}^-$ and line 98 ensure $R(u,v) \in J_{i,j}$; and in the latter case, we clearly have $R(u,v) \in Y_{i,j}^R$. Either way, we have $R(u,v) \in J_{i,j} \cup Y_{i,j}^R$, as required.

For the $\supseteq$ direction of the equality in (C.5), consider arbitrary $R(u,v) \in \mathsf{close}[X_{i,j}^R]$. Then, the way X^R is updated in line 97 ensures $R(u,v) \in \mathsf{close}[X_{i,j-1}^R]$, which together with property (C.11) ensures $R(u,v) \in R\text{-part}[I_{i,j}^n] \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$; this clearly ensures $R(u,v) \in R\text{-part}[I_{i,j}^n]$. Now assume for the sake of a contradiction that $R(u,v) \in \Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j}^R$. Then, $R(u,v) \notin \Delta_{i,j}^m \cup Y_{i,j-1}^R$ ensures $R(u,v) \in (\Delta_{i,j}^- \setminus \Delta_{i,j}^m) \cup J_{i,j} \cup (Y_{i,j}^R \setminus Y_{i,j-1}^R)$; thus, $R(u,v)$ is considered in line 93, 95, or 96. In each of these cases, the set in $X_{i,j-1}^R$ containing u and v is removed in line 97; since $X_{i,j-1}^R$ is valid, we have $R(u,v) \notin \mathsf{close}[X_{i,j}^R]$, which is a contradiction. Consequently, we have $R(u,v) \notin \Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j}^R$, so fact $R(u,v)$ belongs to the required set.

The way X^R is updated in line 97 ensures $\mathsf{close}[X_{i,j}^R] \subseteq \mathsf{close}[X_{i,j-1}^R]$; together with $\mathsf{close}[X_{i,j-1}^R] \subseteq R\text{-part}[I_{i,j}^n] \subseteq R\text{-part}[I_{i,j}^o]$ from property (C.11), this ensures the remaining inclusions of property (C.5).

For property (C.6), consider arbitrary $R(u, v) \in Y_{i,j}^R$. If $R(u, v) \in Y_{i,j-1}^R$, then property (C.12) ensures $R(u, v) \in I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s}$. If $R(u, v)$ is added to Y^R in line 96, then lines 93 and 94 ensure that u and v belong to a set in $X_{i,j-1}^R$; thus, $R(u, v) \in \text{close}[X_{i,j-1}^R]$, which together with the rightmost inclusion of property (C.11) ensures $R(u, v) \in I_{i,j}^o$. Moreover, the oracle function returns true for $R(u, v)$ in line 95, so by Definition 16 we have $R(u, v) \in \Pi_\infty[E_i]$; together with $M \subseteq \Pi^s$, this ensures $R(u, v) \in \Pi_\infty[E_i]^{\leq s}$. Consequently, $R(u, v) \in I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s}$ holds, as required.

Finally, we show that property (S4) holds. We have already established that $X_{i,j-1}^R$ is valid; furthermore, X^R is updated during the execution of $C_{i,j}$ is line 97, which clearly does not affect the validity of X^R; hence, set $X_{i,j}^R$ is also valid. Moreover, line 98 ensures $J_{i,j} \subseteq I_{i,j}^n \setminus \Delta_{i,j}^-$; set $J_{i,j}$ contains only R facts, so the rightmost inclusion of property (C.11) and $M \subseteq \Pi^s$ ensure $J_{i,j} \subseteq R\text{-part}[I_{i,j}^n] \subseteq I_{i,j}^o \cap \mathsf{O}^s$; hence, $J_{i,j} \subseteq I_{i,j}^o \cap \mathsf{O}^s \cap ((I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+)$ holds, as required by the upper bound. To see that $J_{i,j}$ also satisfies the lower bound, consider an arbitrary fact $R(u, w) \in M_\mathsf{D}[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}^-, \Delta_{i,j}^+] \setminus \Pi_\infty[E_i]$. By Definition 9, there exists a rule instance $r\sigma$ with $r \in M$ such that $\mathsf{b}(r\sigma) \subseteq I_{i,j}^o$, $\mathsf{b}(r\sigma) \subseteq I_{i,j}^n$, $\mathsf{b}(r\sigma) \nsubseteq (I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, and $R(u, w) = \mathsf{h}(r\sigma) \in (I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ all hold. Set $\Delta_{i,j}^+$ contains no R-facts, so $R(u, w) \in I_{i,j}^n \setminus \Delta_{i,j}^-$ holds. We have the following possibilities for the choice of r.

- Rule $r \in M$ is of the form (30). Then, $R(u, w) = \mathsf{h}(r\sigma)$ ensures $\mathsf{b}(r\sigma) = \{R(w, u)\}$. Together with $\mathsf{b}(r\sigma) \nsubseteq (I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ and $\mathsf{b}(r\sigma) \subseteq I_{i,j}^n$, this implies $R(w, u) \in \Delta_{i,j}^-$. Assume for the sake of a contradiction that $R(w, u) \in \Delta_{i,j}^m$ holds. Then, the call must be of type D2, and so property (D2.d) implies $R(w, u) \in J_{i,j-1}$—that is, w and u belong to a set removed from X^R during the execution of $C_{i,j-1}$. Together with $R(u, w) \notin \Pi_\infty[E_i]$, this implies $R(u, w) \in J_{i,j-1} = \Delta_{i,j}^m \subseteq \Delta_{i,j}^-$, which is a contradiction. Hence, we have $R(w, u) \in \Delta_{i,j}^- \setminus \Delta_{i,j}^m$. Moreover, $R(u, w) \in I_{i,j}^n \setminus \Delta_{i,j}^-$, and $R(u, w) \notin \Pi_\infty[E_i]$, and $\Delta_{i,j}^m \subseteq \Delta_{i,j}^-$, and property (C.12) jointly ensure $R(u, w) \in R\text{-part}[I_{i,j}^n] \setminus (\Delta_{i,j}^m \cup Y_{i,j-1}^R)$; but then, property (C.11) ensures $R(u, w) \in \text{close}[X_{i,j-1}^R]$. Consequently, the set in $X_{i,j-1}^R$ containing u and w must be considered in line 93 during the execution of $C_{i,j}$. Then, fact $R(u, w)$ is considered in line 95; moreover, function T is correct in the context of E_i and Π, so $R(u, w) \in J_{i,j}$ holds.
- Rule $r \in M$ is of the form (31). Then, there exists v such that $\mathsf{b}(r\sigma) = \{R(u, v), R(v, w)\}$. Analogously to above, we have $R(u, w) \in \text{close}[C_R^{i,j-1}]$. Moreover, $\mathsf{b}(r\sigma) \subseteq I_{i,j}^n$ and $\mathsf{b}(r\sigma) \nsubseteq (I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ imply that one of the two facts must be in $\Delta_{i,j}^-$. We next consider the case when $R(u, v) \in \Delta_{i,j}^-$; the case of $R(v, w) \in \Delta_{i,j}^-$ is analogous and we omit it for the sake of brevity. For the sake of a contradiction, assume that $R(u, v) \in \Delta_{i,j}^m = J_{i,j-1}$ holds. Then, the set containing u, v, and w is removed from X^R during the execution of $C_{i,j-1}$, which contradicts $R(u, w) \in \text{close}[X_{i,j-1}^R]$. Hence, we have $R(u, v) \in \Delta_{i,j}^- \setminus \Delta_{i,j}^m$. Line 93 and $R(u, w) \in \text{close}[X_{i,j-1}^R]$ ensure that $R(u, w)$ is considered in line 95. Finally, function T in the context of E_i and Π, so $R(u, w) \in J_{i,j}$ holds. $\square$

Claim 42. *If call $C_{i,j}$ is of type (R) and call $C_{i,j-1}$ satisfies properties (S1)–(S4), then call $C_{i,j}$ satisfies properties (S3) and (S4).*

Proof. For the leftmost inclusion of property (C.7), call $C_{i,j-1}$ involves the Del^M module function so properties (C.5) for i and $j-1$ ensures $R\text{-part}[I_{i,j-1}^n] \setminus (\Delta_{i,j-1}^- \cup J_{i,j-1} \cup Y_{i,j-1}^R) \subseteq \text{close}[X_{i,j-1}^R]$. Together with properties (R.b) and (R.d), and the fact that $\Delta_{i,j-1}^+$ contains no R-facts, this implies $R\text{-part}[I_{i,j}^n] \subseteq Y_{i,j-1}^R \cup \text{close}[X_{i,j-1}^R]$. Moreover, line 99 and Claim 39 ensure $Y_{i,j-1}^R \subseteq \text{close}[X_{i,j}^R]$, $\text{close}[X_{i,j-1}^R] \subseteq \text{close}[X_{i,j}^R]$, and $J_{i,j} \subseteq \text{close}[X_{i,j}^R]$. Hence, $R\text{-part}[I_{i,j}^n] \cup J_{i,j} \subseteq \text{close}[X_{i,j}^R]$ holds.

For the second inclusion of property (C.7), property (C.5) for i and $j-1$, $M \subseteq \Pi^s$, and condition (D2.b) or (D1.d) (depending on the type of previous call) ensure $\text{close}[X_{i,j-1}^R] \subseteq \Pi_\infty[E_{i-1}]^s$. Moreover, property (C.6) for i and $j-1$ ensures $Y_{i,j-1}^R \subseteq \Pi_\infty[E_{i-1}]^s$. As a result, line 99 and Claim 39 ensure $\text{close}[X_{i,j}^R] \subseteq M_\infty[\text{close}[X_{i,j-1}^R] \cup Y_{i,j-1}^R] \subseteq \Pi_\infty[E_{i-1}]^s$, and so condition (R.c) ensures $\text{close}[X_{i,j}^R] \subseteq R\text{-part}[I_{i,j}^o]$. We next prove $\text{close}[X_{i,j}^R] \setminus (R\text{-part}[I_{i,j}^o] \setminus \Delta_{i,j}) \subseteq J_{i,j}$, which implies the required property. To this end, consider an arbitrary fact $R(u, v) \in \text{close}[X_{i,j}^R] \setminus (R\text{-part}[I_{i,j}^o] \setminus \Delta_{i,j})$. Then, $\text{close}[X_{i,j}^R] \subseteq R\text{-part}[I_{i,j}^o]$ ensures $R(u, v) \in R\text{-part}[I_{i,j}^o]$. Together with $R(u, v) \notin R\text{-part}[I_{i,j}^o] \setminus \Delta_{i,j}$, this implies $R(u, v) \in \Delta_{i,j}$. Then, condition (R.f) ensures $R(u, v) \notin I_{i,j}^n$, which together with condition (R.d) and property (C.5) for i and $j-1$ ensures $R(u, v) \notin \text{close}[X_{i,j-1}^R]$. But then, $R(u, v) \in \text{close}[X_{i,j}^R]$, line 99, and Claim 39 ensure that $R(u, v)$ is returned by the $\textsc{CloseEdges}(Y^R)$ call made in line 99. Together with $R(u, v) \in \Delta_{i,j}$, this ensures $R(u, v) \in J_{i,j}$, as required.

For the rightmost inclusion of (C.7), conditions (R.c), (R.e), and (R.f), and $M \subseteq \Pi^s$ ensure $R\text{-part}[I_{i,j}^o] \setminus \Delta_{i,j} \subseteq \Pi_\infty[E_i]^s$. Next we show that $J_{i,j} \subseteq \Pi_\infty[E_i]^s$ holds. Property (C.5) for i and $j-1$ ensures $\text{close}[X_{i,j-1}^R] \subseteq R\text{-part}[I_{i,j-1}^n] \setminus \Delta_{i,j-1}^-$, which together with conditions (R.d), (R.e), and $M \subseteq \Pi^s$ ensures $\text{close}[X_{i,j-1}^R] \subseteq \Pi_\infty[E_i]^s$. Moreover, property (C.6) for i and $j-1$ ensures $Y_{i,j-1}^R \subseteq \Pi_\infty[E_i]^s$. As a result, line 99 and Claim 39 ensure $J_{i,j} \subseteq M_\infty[\text{close}[X_{i,j-1}^R] \cup Y_{i,j-1}^R] \subseteq \Pi_\infty[E_i]^s$.

Property (C.8) holds due to line 100.

Finally, we show that property (S4) holds. The inductive assumption ensures that $X_{i,j-1}^R$ is valid, and X^R is only updated in line 99 during the execution of $C_{i,j}$; hence, Claim 39 ensures that $X_{i,j}^R$ is also valid. Moreover, line 99 clearly ensures

$J_{i,j} \subseteq \Delta_{i,j}$; then, by condition (R.f) we have $J_{i,j} \cap I^n_{i,j} = \emptyset$; together with the rightmost inclusion of (C.7), we conclude that $J_{i,j}$ satisfies the upper bound. To see that $J_{i,j}$ also satisfies the lower bound, consider arbitrary $R(u,w) \in M_R[I^o_{i,j}, I^n_{i,j} \vdots \Delta_{i,j}]$. By Definition 10 there exists rule instance $r\sigma$ with $r \in M$ such that $R(u,w) = h(r\sigma) \in \Delta_{i,j}$, and $b(r\sigma) \subseteq I^o_{i,j}$, and $b(r\sigma) \subseteq I^n_{i,j}$ all hold. There are two possibilities, which we discuss below.

- Rule $r \in M$ is of the form (30). Then, $R(u,w) = h(r\sigma)$ implies $b(r\sigma) = \{R(w,u)\}$. Together with $b(r\sigma) \subseteq I^n_{i,j}$, conditions (R.b) and (R.d), and the fact that $\Delta^+_{i,j-1}$ contains no R-facts, we have $R(w,u) \in I^n_{i,j-1} \setminus (\Delta^-_{i,j-1} \cup J_{i,j-1})$. Property (C.5) for i and $j-1$ ensures $R(w,u) \in \text{close}[X^R_{i,j-1}] \cup Y^R_{i,j-1}$; hence, line 99 and Claim 39 ensure $R(w,u) \in \text{close}[X^R_{i,j}]$, which in turn implies $R(u,w) \in \text{close}[X^R_{i,j}]$. Moreover, $R(u,w) \in \Delta_{i,j}$ implies $R(u,w) \notin I^n_{i,j}$, which together with condition (R.d) and property (C.5) for i and $j-1$ implies $R(u,w) \notin \text{close}[X^R_{i,j-1}]$. But then, $R(u,w) \in \text{close}[X^R_{i,j}]$, line 99, and Claim 39 ensure that $R(u,w)$ is returned by the CLOSEEDGES(Y^R) call in line 99; but then, $R(u,w) \in \Delta_{i,j}$ ensures $R(u,w) \in J_{i,j}$, as required.
- Rule $r \in M$ is of the form (31). Then, there exists v such that $b(r\sigma) = \{R(u,v), R(v,w)\}$ holds. Analogously to above, we have $\{R(u,v), R(v,w)\} \subseteq \text{close}[X^R_{i,j}]$ and $R(u,w) \notin \text{close}[X^R_{i,j-1}]$. The former ensures $R(u,w) \in \text{close}[X^R_{i,j}]$. Consequently, line 99 and Claim 39 ensure that $R(u,w)$ is returned by the CLOSEEDGES(Y^R) call in line 99; but then, $R(u,w) \in \Delta_{i,j}$ ensures $R(u,w) \in J_{i,j}$, as required. $\square$

Appendix D. Proof of Theorem 30

Theorem 30. *Functions* $\text{Add}^{\Pi_{rch}}$, $\text{Del}^{\Pi_{rch}}$, *and* $\text{Red}^{\Pi_{rch}}$ *are correct.*

Consider a regular chain program Π_{rch}, program Π, stratification λ of Π, stratum index s such that $\Pi_{rch} \subseteq \Pi^s$, sequence of datasets $E_0, \ldots, E_m$, and a call history H for Π_{rch} that is compatible with Π, λ, s and $E_0, \ldots, E_m$. Let Σ^H be the set of all head predicates appearing in Π_{rch}, let Σ^P be the set of all predicates appearing in Π_{rch}, and let $\Sigma^B = \Sigma^P \setminus \Sigma^H$. For each predicate $R \in \Sigma^H$, we fix an NFA $N^R = \langle Q^R, \Sigma_b, \delta^R, q^R_s, F^R \rangle$ that recognises precisely the unfoldings of R w.r.t. Π_{rch}; we assume that each NFA is free of ε-transitions, satisfies $q^R_s \notin F^R$, and does not share states with any other NFA. We assume that the call history H is of the form as specified in Definition 12. For each $0 \le i \le m$ and each $1 \le j \le n_i$, let $I^q_{i,j}$, $\Delta^q_{i,j}$, $X_{i,j}$, $Y_{i,j}$, and each $Z^R_{i,j}$ with $R \in \Sigma^H$ be the values of I^q, Δ^q, X, Y, and Z^R, respectively, after call $C_{i,j}$. Finally, let $I^q_{0,0} = \Delta^q_{0,0} = X_{0,0} = Y_{0,0} = Z^R_{0,0} = \emptyset$ for each $R \in \Sigma^H$ and, for $1 \le i \le m$, we let $I^q_{i,0} = I^q_{i-1,h_{i-1}}$, $\Delta^q_{i,0} = \Delta^q_{i-1,h_{i-1}}$, $X_{i,0} = X_{i-1,h_{i-1}}$, $Y_{i,0} = Y_{i-1,h_{i-1}}$, and $Z^R_{i,0} = Z^R_{i-1,h_{i-1}}$ for each $R \in \Sigma^H$.

We next introduce some useful abbreviations and notation. To simplify the notation, let $M = \Pi_{rch}$ for the rest of this section. Also, for S a set of facts and T a set of predicates, T-part[S] is the subset of S containing precisely all facts of S whose predicate is in T; by abuse of notation, we write $\{R\}$-part[S] as R-part[S] when R is a predicate. We use analogous notation when S is a set of q-facts and T is a set of states. Furthermore, for $R \in \Sigma^H$ a predicate and X and Δ datasets, we define $\text{close}^R[X, \Delta]$ as follows.

$$
\begin{aligned}
\text{close}^R[X, \Delta] = \{ & q(u,v) \mid \text{there exist } \ell \ge 1, \text{ states } q_0, \ldots, q_\ell \in Q^R, \text{ binary predicates } R_0, \ldots, R_{\ell-1}, \\
& \text{constants } w_0, \ldots, w_\ell, \text{ and an integer } k \text{ with } 0 \le k < \ell \text{ such that } q_0 = q^R_s, \; w_0 = u, \\
& q_\ell = q, \; w_\ell = v, \; q_{i+1} \in \delta^R(q_i, R_i) \text{ for } i \text{ with } 0 \le i < \ell, \text{ and} \\
& R_k(w_k, w_{k+1}) \in \Delta \text{ and } R_m(w_m, w_{m+1}) \in X \text{ for } m \text{ with } 0 \le m < \ell \text{ and } m \ne k \}
\end{aligned}
\tag{D.1}
$$

To simplify the notation, we abbreviate $\text{close}^R[X, X]$ as just $\text{close}^R[X]$. For $R \in \Sigma^H$ a predicate, X a dataset, and B a q-dataset, we define $\text{ext}^R[B, X]$ as follows.

$$
\begin{aligned}
\text{ext}^R[B, X] = \{ & q(u,v) \mid \text{there exist } \ell \ge 0, \text{ states } q_0, \ldots, q_\ell \in Q^R, \text{ binary predicates } R_0, \ldots, R_{\ell-1}, \\
& \text{and constants } w_0, \ldots, w_\ell \text{ such that } q_\ell = q, \; w_\ell = v, \; q_0(u, w_0) \in B, \text{ and} \\
& q_{i+1} \in \delta^R(q_i, R_i) \text{ and } R_i(w_i, w_{i+1}) \in X \text{ for } i \text{ with } 0 \le i < \ell \}
\end{aligned}
\tag{D.2}
$$

Finally, for B a q-dataset, we define $\text{fin}[B]$ as follows.

$$
\text{fin}[B] = \{ R(u,v) \mid \exists q \in F^R \text{ such that } q(u,v) \in B \}
\tag{D.3}
$$

Before proceeding with the main proof, we first prove a clam that characterises the ADDEDGES function. This will allow us to later simplify the analysis of the Add^M and Red^M functions.

Claim 43. *For each call of the form* $J^q := \text{ADDEDGES}(\Delta, Q, B)$, *let* I^q *be the value during the call, and let* X_1 *and* X_2 *be the values of* X *before and after the call. If* $\text{close}^R[X_1] = Q^R\text{-part}[I^q] \cup \text{ext}^R[Q, X_1]$ *and* $\text{ext}^R[B, X_1] \subseteq Q^R\text{-part}[I^q]$ *hold for each binary predicate* $R \in \Sigma^H$, *then* $X_2 = X_1 \cup \Sigma^P\text{-part}[\Delta]$ *and* (D.4) *holds for each binary predicate* $R \in \Sigma^H$.

$$\mathsf{close}^R[X_2] = Q^{\,R}\text{-part}[I^q \cup J^q] \tag{D.4}$$

Proof. Consider an arbitrary call as specified in the claim. Note that X is only updated in line 108, so $X_2 = X_1 \cup \Sigma^P\text{-part}[\Delta]$ clearly holds. Next we show that (D.4) holds for each binary predicate $R \in \Sigma^H$. To this end, consider arbitrary binary predicate $R \in \Sigma^H$ appearing in the head of a rule in M.

For $\subseteq$ direction of property (D.4), we prove (D.5). We next consider the two inclusions of this property in isolation.

$$\mathsf{close}^R[X_2] \setminus Q^{\,R}\text{-part}[I^q] \subseteq \mathsf{ext}^R[B \cup J^q, X_2] \setminus Q^{\,R}\text{-part}[I^q] \subseteq Q^{\,R}\text{-part}[J^q] \tag{D.5}$$

For the first inclusion of (D.5), we consider an arbitrary q-fact $q(u,v) \in \mathsf{close}^R[X_2] \setminus Q^{\,R}\text{-part}[I^q]$. If $q(u,v) \in \mathsf{close}^R[X_1]$, then $q(u,v) \in \mathsf{ext}^R[Q, X_1]$ holds by the claim assumption; line 106 ensures $Q \subseteq J^q$; and $X_1 \subseteq X_2$ clearly holds; thus, we have $q(u,v) \in \mathsf{ext}^R[B \cup J^q, X_2]$, as required. Otherwise, we have $q(u,v) \in \mathsf{close}^R[X_2] \setminus \mathsf{close}^R[X_1]$. Then, by (D.1), there exist an integer $\ell \geq 1$, states $q_0, \dots, q_\ell \in Q^R$, and facts $\{R_0(w_0, w_1), \dots, R_{\ell-1}(w_{\ell-1}, w_\ell)\} \subseteq X_2$ such that $q_0 = q_s^R$, $w_0 = u$, $q_\ell = q$, and $w_\ell = v$, and $q_{i+1} \in \delta^R(q_i, R_i)$ for $0 \leq i < \ell$. Moreover, $q(u,v) \notin \mathsf{close}^R[X_1]$ ensures that $R_k(w_k, w_{k+1}) \in X_1$ for at least one k with $0 \leq k < \ell$. Without loss of generality, we can choose k to be the smallest index satisfying $R_k(w_k, w_{k+1}) \in X_2 \setminus X_1 \subseteq \Sigma^P\text{-part}[\Delta]$. We consider the following possibilities.

- $k = 0$. We thus have, $R_0(w_0, w_1) = R_k(w_k, w_{k+1}) \in \Sigma^P\text{-part}[\Delta]$, so fact $R_0(w_0, w_1)$ is considered in line 107. But then, $q_0 = q_s^R$ and $q_1 \in \delta^R(q_0, R_0)$ ensure that $q_1(w_0, w_1)$ is considered in line 110; this ensures $q_1(w_0, w_1) \in Q^{\,R}\text{-part}[B \cup J^q]$, which in turn ensures $q(u,v) = q_\ell(w_0, w_\ell) \in \mathsf{ext}^R[B \cup J^q, X_2]$, as required.
- $k > 0$. Since k is the smallest index such that $R_k(w_k, w_{k+1}) \in X_2 \setminus X_1$ holds, $R_i(w_i, w_{i+1}) \in X_1$ holds for each $0 \leq i < k$. Thus, $q_k(w_0, w_k) \in \mathsf{close}^R[X_1]$ holds. We further have the following possibilities.

  - $q_k(w_0, w_k) \in Q^{\,R}\text{-part}[I^q]$. Then, $R_k(w_k, w_{k+1}) \in \Sigma^P\text{-part}[\Delta]$ ensures that $R_k(w_k, w_{k+1})$ is considered in line 107. But then, $q_k(w_0, w_k) \in Q^{\,R}\text{-part}[I^q]$ and $q_{k+1} \in \delta^R(q_k, R_k)$ ensure that $q_{k+1}(w_0, w_{k+1})$ is considered in line 112; this ensures $q_{k+1}(w_0, w_{k+1}) \in Q^{\,R}\text{-part}[B \cup J^q]$, which in turn ensures $q(u,v) = q_\ell(w_0, w_\ell) \in \mathsf{ext}^R[B \cup J^q, X_2]$.
  - $q_k(w_0, w_k) \notin Q^{\,R}\text{-part}[I^q]$. Then, $\mathsf{close}^R[X_1] = Q^{\,R}\text{-part}[I^q] \cup \mathsf{ext}^R[Q, X_1]$ implies $q_k(w_0, w_k) \in \mathsf{ext}^R[Q, X_1]$; moreover, $Q \subseteq B \cup J^q$ and $X_1 \subseteq X_2$ imply $\mathsf{ext}^R[Q, X_1] \subseteq \mathsf{ext}^R[B \cup J^q, X_2]$, so we have $q_k(w_0, w_k) \in \mathsf{ext}^R[B \cup J^q, X_2]$; finally, this ensures $q(u,v) = q_\ell(w_0, w_\ell) \in \mathsf{ext}^R[B \cup J^q, X_2]$.

For the second inclusion of (D.5), we prove instead that $\mathsf{ext}^R[B \cup J^q, X_2] \subseteq Q^{\,R}\text{-part}[I^q \cup J^q]$ holds. Consider an arbitrary q-fact $q(u,v) \in \mathsf{ext}^R[B \cup J^q, X_2]$; then, by (D.2), there exist an integer $\ell \geq 0$, states $q_0, \dots, q_\ell$, binary predicates $R_0, \dots, R_{\ell-1}$, and constants $w_0, \dots, w_\ell$ such that $q_\ell = q$, $w_\ell = v$, $q_0(u, w_0) \in B \cup J^q$, and $q_{i+1} \in \delta^R(q_i, R_i)$ and $R_i(w_i, w_{i+i}) \in X_2$ for i with $0 \leq i < \ell$. We prove by induction on i with $0 \leq i < \ell$ that property $(\lozenge)$ holds.

$(\lozenge)$ $q_i(w_0, w_i) \in \mathsf{ext}^R[B, X_1] \cup Q^{\,R}\text{-part}[J^q]$

For $i = \ell - 1$, property $(\lozenge)$ ensures $q(u,v) \in \mathsf{ext}^R[B, X_1] \cup Q^{\,R}\text{-part}[J^q] \subseteq Q^{\,R}\text{-part}[I^q \cup J^q] = Q^{\,R}\text{-part}[I^q \cup J^q]$, as required. For the base case, property $(\lozenge)$ holds trivially for $i = 0$. For the inductive step, consider arbitrary i with $0 < i < \ell$ such that property $(\lozenge)$ holds for $i - 1$; we show that property $(\lozenge)$ holds for i as well. We have the following three cases.

- $q_{i-1}(w_0, w_{i-1}) \in \mathsf{ext}^R[B, X_1]$ and $R_{i-1}(w_{i-1}, w_i) \in X_1$. Then, $q_i \in \delta^R(q_{i-1}, R_{i-1})$ ensures $q_i(w_0, w_i) \in \mathsf{ext}^R[B, X_1]$, as required by property $(\lozenge)$.
- $q_{i-1}(w_0, w_{i-1}) \in \mathsf{ext}^R[B, X_1]$ and $R_{i-1}(w_{i-1}, w_i) \in X_2 \setminus X_1 \subseteq \Sigma^P\text{-part}[\Delta]$. Then, lines 107, 111, and 112 of the algorithm ensure that $q_i(w_0, w_i) \in Q^{\,R}\text{-part}[B \cup J^q]$; note that $q_{i-1}(w_0, w_{i-1}) \in I^q$ holds in line 111 because $\mathsf{ext}^R[B, X_1] \subseteq Q^{\,R}\text{-part}[I^q]$ holds by the assumption of our claim. Thus, $q_i(w_0, w_i) \in \mathsf{ext}^R[B, X_1] \cup Q^{\,R}\text{-part}[J^q]$ holds, as required by property $(\lozenge)$.
- $q_{i-1}(w_0, w_{i-1}) \notin \mathsf{ext}^R[B, X_1]$. Then, property $(\lozenge)$ for $i - 1$ ensures $q_{i-1}(w_0, w_{i-1}) \in Q^{\,R}\text{-part}[J^q]$, so $q_{i-1}(w_0, w_{i-1})$ is added to J^q and Q in line 106, 110, 112, or 116. The q-fact $q_{i-1}(w_0, w_{i-1})$ is thus removed from Q in line 114 at some point. But then, $R_{i-1}(w_{i-1}, w_i) \in X_2$ and line 115 ensure that $q_i(w_0, w_i)$ is considered in line 116, which in turn ensures that $q_i(w_0, w_i) \in Q^{\,R}\text{-part}[B \cup J^q] \subseteq \mathsf{ext}^R[B, X_1] \cup Q^{\,R}\text{-part}[J^q]$ holds, as required by property $(\lozenge)$.

We now prove the $\supseteq$ direction of property (D.4). Precondition $\mathsf{close}^R[X_1] = Q^{\,R}\text{-part}[I^q] \cup \mathsf{ext}^R[Q, X_1]$ and $X_1 \subseteq X_2$ ensure $Q^{\,R}\text{-part}[I^q] \subseteq \mathsf{close}^R[X_1] \subseteq \mathsf{close}^R[X_2]$. To complete the proof, we show that $Q^{\,R}\text{-part}[J^q] \subseteq \mathsf{close}^R[X_2]$ holds as well. To this end, let N be the total number of iterations for the loop of lines 113–116; moreover, let J_0^q be the value of J^q right before line 113, and, for each $1 \leq k \leq N$, let J_k^q be the value of J^q after the k-th iteration of lines 114–116. We show by induction on k with $0 \leq k \leq N$ that $Q^{\,R}\text{-part}[J_k^q] \subseteq \mathsf{close}^R[X_2]$ holds; then, $Q^{\,R}\text{-part}[J_k^q] \subseteq \mathsf{close}^R[X_2]$ for $k = N$ ensures the required property.

- For the base case, note that each q-fact $q(u,v) \in Q^{\,R}\text{-part}[J_0^q]$ is added to J_0^q in line 106, 110, or 112. In the first case, assumption $\mathsf{close}^R[X_1] = Q^{\,R}\text{-part}[I^q] \cup \mathsf{ext}^R[Q, X_1]$ ensures $q(u,v) \in \mathsf{close}^R[X_1] \subseteq \mathsf{close}^R[X_2]$. In the second case,

lines 107–110 ensure that there exists $R_i(u, v) \in X_2$ such that $q \in \delta^R(q_s^R, R_i)$, which implies $q(u, v) \in \mathsf{close}^R[X_2]$. In the third case, lines 107, 109, and 111 ensure that there exist $q_1(u, w) \in Q^R\text{-part}[I^q]$ and $R_i(w, v) \in X_2$ such that $q \in \delta^R(q_1, R_i)$ holds; then, $\mathsf{close}^R[X_1] = Q^R\text{-part}[I^q] \cup \mathsf{ext}^R[Q, X_1]$ and $\mathsf{close}^R[X_1] \subseteq \mathsf{close}^R[X_2]$ ensure $q_1(u, w) \in \mathsf{close}^R[X_2]$, which in turn ensures $q(u, v) \in \mathsf{close}^R[X_2]$, as required.

- For the inductive step, consider an arbitrary k with $1 \le k \le N$ such that $Q^R\text{-part}[J_{k-1}^q] \subseteq \mathsf{close}^R[X_2]$ holds, and consider an arbitrary q-fact $q(u, v) \in Q^R\text{-part}[J_k^q]$. If $q(u, v) \in Q^R\text{-part}[J_{k-1}^q]$, then the inductive assumption for $k-1$ ensures $q(u, v) \in \mathsf{close}^R[X_2]$. Otherwise, $q(u, v)$ is added to J_k^q in line 116. Thus, line 114 ensures that some q-fact $q_1(u, w)$ is extracted at some point from the set Q; since all facts added to Q are also added to J^q, we also have $q_1(u, w) \in Q^R\text{-part}[J_{k-1}^q]$; but then, the inductive assumption for $k-1$ ensures $q_1(u, w) \in \mathsf{close}^R[X_2]$. In addition, line 115 ensures that there exists $R_i(w, v) \in X_2$ such that $q \in \delta^R(q_1, R_i)$ holds. Together, these observations ensure $q(u, v) \in \mathsf{close}^R[X_2]$, as required. □

We now prove Theorem 30 by showing that, for each $0 \le i \le m$ and each $1 \le j \le h_i$, call $C_{i,j}$ in the history H satisfies properties (N1)–(N4).

(N1) If call $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Add}^M[I_{i,j} \colon \Delta_{i,j}^-, \Delta_{i,j}^+ \colon \Delta_{i,j}^m]$, then the following properties hold.

$$\mathsf{close}^R[X_{i,j}] = Q^R\text{-part}[I_{i,j}^q] \text{ for each } R \in \Sigma^H \tag{D.6}$$

$$\Sigma^B\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] = \Sigma^B\text{-part}[X_{i,j}] \subseteq X_{i,j} \subseteq \Pi_\infty[E_i]^{\le s} \tag{D.7}$$

$$\mathsf{fin}[I_{i,j}^q] = \Sigma^H\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \cup J_{i,j}] \subseteq \Pi_\infty[E_i]^s \tag{D.8}$$

$$\Delta_{i,j}^q = Y_{i,j} = Z_{i,j}^R = \emptyset \text{ for each } R \in \Sigma^H \tag{D.9}$$

(N2) If call $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Del}^M[I_{i,j}^o, I_{i,j}^n \colon \Delta_{i,j}^-, \Delta_{i,j}^+ \colon \Delta_{i,j}^m]$, then the following properties hold.

$$Q^R\text{-part}[I_{i,j}^q] \subseteq \mathsf{close}^R[X_{i,j}] \subseteq \mathsf{close}^R[X_{i,0}] = Q^R\text{-part}[I_{i,0}^q] \text{ for each } R \in \Sigma^H \tag{D.10}$$

$$\mathsf{ext}^R[\Delta_{i,j}^q, X_{i,j}] \subseteq Q^R\text{-part}[\Delta_{i,j}^q] \subseteq \mathsf{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-] \text{ for each } R \in \Sigma^H \tag{D.11}$$

$$\Sigma^B\text{-part}[I_{i,j}^o \cap (I_{i,j}^n \setminus \Delta_{i,j}^-)] = \Sigma^B\text{-part}[X_{i,j}] \subseteq X_{i,j} = X_{i,0} \setminus ((I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-) \subseteq I_{i,j}^o \cap (I_{i,j}^n \setminus \Delta_{i,j}^-) \tag{D.12}$$

$$\mathsf{fin}[\mathsf{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]] = R\text{-part}[(I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j}] \text{ for each } R \in \Sigma^H \tag{D.13}$$

$$\mathsf{fin}[I_{i,j}^q] = \Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})] \subseteq \Sigma^H\text{-part}[I_{i,j}^n] \subseteq \Sigma^H\text{-part}[I_{i,j}^o] = \mathsf{fin}[I_{i,0}^q] \tag{D.14}$$

$$Y_{i,j} \subseteq I_{i,j}^o \cap \Pi_\infty[E_i]^{\le s} \tag{D.15}$$

$$Z_{i,j}^R \supseteq \{u \mid \text{there exist } q \text{ and } v \text{ such that } q(u, v) \in Q^R\text{-part}[I_{i,0}^q \setminus I_{i,j}^q]\} \text{ for each } R \in \Sigma^H \tag{D.16}$$

$$\Delta_{i,j}^q \supseteq I_{i,0}^q \setminus I_{i,j}^q \text{ and } \Delta_{i,j}^q \cap I_{i,j}^q = \emptyset \text{ and } \mathsf{fin}[\Delta_{i,j}^q] \cap (I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})) = \emptyset \tag{D.17}$$

(N3) If call $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Red}^M[I_{i,j}^o, I_{i,j}^n \colon \Delta_{i,j}]$, then the following properties hold.

$$\mathsf{close}^R[X_{i,j}] = Q^R\text{-part}[I_{i,j}^q] \text{ for each } R \in \Sigma^H \tag{D.18}$$

$$\Sigma^B\text{-part}[I_{i,j}^o \cap I_{i,j}^n] = \Sigma^B\text{-part}[X_{i,j}] \subseteq X_{i,j} \subseteq \Pi_\infty[E_i]^{\le s} \tag{D.19}$$

$$\Sigma^H\text{-part}[I_{i,j}^n \cup J_{i,j}] \subseteq \mathsf{fin}[I_{i,j}^q] \subseteq \Sigma^H\text{-part}[(I_{i,j}^o \setminus \Delta_{i,j}) \cup J_{i,j}] \subseteq \Pi_\infty[E_i]^s \tag{D.20}$$

$$\Delta_{i,j}^q = Y_{i,j} = Z_{i,j}^R = \emptyset \text{ for each } R \in \Sigma^H \tag{D.21}$$

(N4) Call $C_{i,j}$ is correct.

We prove this by double induction on i and j, where we consider each call $C_{i,j}$ of type from Table 2. The base case involves a call of type A1, and the inductive step involves all remaining calls. We split the proof into a separate claim for each module function.

Claim 44. *If $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Add}^M[I_{i,j} \colon \Delta_{i,j}^-, \Delta_{i,j}^+ \colon \Delta_{i,j}^m]$ and either $i = 0$ and $j = 1$, or $j > 1$ and call $C_{i,j-1}$ satisfies properties (N1)–(N4), then $C_{i,j}$ satisfies properties (N1) and (N4).*

Proof. We first capture the preconditions for the call $C_{i,j}$. These are established either at the beginning of the algorithm, or by the preceding call $C_{i,j-1}$. In particular, we prove that the following properties hold.

$$\mathsf{close}^R[X_{i,j-1}] = Q^{\,R}\text{-}\mathsf{part}[I^q_{i,j-1}] \text{ for each } R \in \Sigma^H \tag{D.22}$$

$$\Sigma^B\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^{\mathsf{m}}_{i,j}] = \Sigma^B\text{-}\mathsf{part}[X_{i,j-1}] \subseteq X_{i,j-1} \subseteq \Pi_\infty[E_i]^{\leq s} \tag{D.23}$$

$$\Sigma^H\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^{\mathsf{m}}_{i,j}] \subseteq \mathsf{fin}[I^q_{i,j-1}] \subseteq \Sigma^H\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^+_{i,j}] \subseteq \Pi_\infty[E_i]^s \tag{D.24}$$

Towards this goal, call $C_{i,j}$ can be of one of the following three types.

- Assume that $C_{i,j}$ is of type A1. For each $R \in \Sigma^H$, both sides of the equality in property (D.22) are empty, so the equality holds. Moreover, $I_{i,j} = \Delta^-_{i,j} = \Delta^{\mathsf{m}}_{i,j} = X_{i,j-1} = \emptyset$ ensures property (D.23). Finally, $I_{i,j} = \Delta^-_{i,j} = \Delta^{\mathsf{m}}_{i,j} = I^q_{i,j-1} = \emptyset$ ensures the first and second inclusion of property (D.24), and condition (A1.a) and $M \subseteq \Pi^s$ ensure the third inclusion.
- Assume that $C_{i,j}$ is of type A2. For each $R \in \Sigma^H$, property (D.6) for i and $j-1$ ensures $\mathsf{close}^R[X_{i,j-1}] = Q^{\,R}\text{-}\mathsf{part}[I^q_{i,j-1}]$, so property (D.22) holds. Moreover, property (D.7) for i and $j-1$ and condition (A2.b) ensure $\Sigma^B\text{-}\mathsf{part}[I_{i,j}] = \Sigma^B\text{-}\mathsf{part}[X_{i,j-1}]$; thus, $\Delta^{\mathsf{m}}_{i,j} = J_{i,j-1}$ from condition (A2.c) and $\Sigma^B\text{-}\mathsf{part}[J_{i,j-1}] = \Delta^-_{i,j} = \emptyset$ ensure $\Sigma^B\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^{\mathsf{m}}_{i,j}] = \Sigma^B\text{-}\mathsf{part}[X_{i,j-1}]$, so the equality in (D.23) holds; property (D.7) for i and $j-1$ also ensures the remaining inclusions in property (D.23). Finally, property (D.8) for i and $j-1$ and conditions (A2.b) and (A2.c) jointly ensure the first inclusion of property (D.24), as well as $\mathsf{fin}[I^q_{i,j-1}] \subseteq \Sigma^H\text{-}\mathsf{part}[I_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}] \subseteq \Pi_\infty[E_i]^s$. Then, $\Delta^-_{i,j} = \emptyset$ and $\Delta^{\mathsf{m}}_{i,j} \subseteq \Delta^+_{i,j}$ from condition (A2.c) ensure the second inclusion of property (D.24); in addition, $\Sigma^H\text{-}\mathsf{part}[I_{i,j} \cup \Delta^{\mathsf{m}}_{i,j}] \subseteq \Pi_\infty[E_i]^s$ and $\Delta^+_{i,j} \subseteq \Pi_\infty[E_i]^s$ from condition (A2.c) ensure the third inclusion of property (D.24).
- Assume that $C_{i,j}$ is of type A3. For each $R \in \Sigma^H$, property (D.18) for i and $j-1$ ensures $\mathsf{close}^R[X_{i,j-1}] = Q^{\,R}\text{-}\mathsf{part}[I^q_{i,j-1}]$, so property (D.22) holds. For (D.23), note that property (D.19) for i and $j-1$ ensures $\Sigma^B\text{-}\mathsf{part}[I^{\mathsf{o}}_{i,j-1} \cap I^{\mathsf{n}}_{i,j-1}] = \Sigma^B\text{-}\mathsf{part}[X_{i,j-1}]$; moreover, $\Delta^{\mathsf{m}}_{i,j} = J_{i,j-1}$ from condition (A3.f) and $\Sigma^B\text{-}\mathsf{part}[J_{i,j-1}] = \emptyset$ ensure $\Sigma^B\text{-}\mathsf{part}[\Delta^{\mathsf{m}}_{i,j}] = \emptyset$; furthermore, conditions (A3.b), (A3.d), and (R.c) ensure $\Sigma^B\text{-}\mathsf{part}[I_{i,j} \setminus \Delta^-_{i,j}] = \Sigma^B\text{-}\mathsf{part}[I^{\mathsf{o}}_{i,j-1} \cap I^{\mathsf{n}}_{i,j-1}]$; together, all of these observations ensure that $\Sigma^B\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^{\mathsf{m}}_{i,j}] = \Sigma^B\text{-}\mathsf{part}[I^{\mathsf{o}}_{i,j-1} \cap I^{\mathsf{n}}_{i,j-1}] = \Sigma^B\text{-}\mathsf{part}[X_{i,j-1}]$ holds, as required by the equality in (D.23). The remaining $\subseteq$ relations in (D.23) directly follow from (D.19) for i and $j-1$. Next we prove property (D.24). Conditions (A3.b) and (A3.d), and $M \subseteq \Pi^s$, ensure $\Sigma^H\text{-}\mathsf{part}[I_{i,j} \setminus \Delta^-_{i,j}] \subseteq \Sigma^H\text{-}\mathsf{part}[I^{\mathsf{n}}_{i,j-1}]$; together with the leftmost inclusion of property (D.20) for i and $j-1$ and $\Delta^{\mathsf{m}}_{i,j} = J_{i,j-1}$ from condition (A3.f) this ensures the leftmost inclusion of property (D.24). Furthermore, conditions (R.c), (R.f), (A3.b), (A3.d), and (A3.f), and $M \subseteq \Pi^s$ ensure

$$\Sigma^H\text{-}\mathsf{part}[(I^{\mathsf{o}}_{i,j-1} \setminus \Delta_{i,j-1}) \cup J_{i,j-1}] \subseteq \Sigma^H\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^+_{i,j}];$$

together with property (D.20) for i and $j-1$ this ensures the second inclusion of property (D.24). Finally, conditions (A3.b), (A3.d), and (R.e), and $M \subseteq \Pi^s$ jointly ensure $\Sigma^H\text{-}\mathsf{part}[I_{i,j} \setminus \Delta^-_{i,j}] \subseteq \Pi_\infty[E_i]^s$; together with $\Delta^+_{i,j} \subseteq \Pi_\infty[E_i]^{\leq s}$ from condition (A3.f), this ensures the rightmost inclusion of property (D.24).

We are now ready to prove that call $C_{i,j}$ satisfies properties (N1) and (N4). For each $R \in \Sigma^H$, property (D.22) ensures $\mathsf{ext}^R[I^q_{i,j-1}, X_{i,j-1}] \subseteq Q^{\,R}\text{-}\mathsf{part}[I^q_{i,j-1}]$; thus, the preconditions for the ADDEDGES call in line 102 are satisfied.

For property (D.6), Claim 43 and the way I^q is updated in line 103 ensures $\mathsf{close}^R[X_{i,j}] = Q^{\,R}\text{-}\mathsf{part}[I^q_{i,j}]$, as required.

We now prove property (D.7). Line 102 and Claim 43 ensure $X_{i,j} = X_{i,j-1} \cup \Sigma^P\text{-}\mathsf{part}[\Delta^+_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}]$. Now for the equality of the property, $X_{i,j} = X_{i,j-1} \cup \Sigma^P\text{-}\mathsf{part}[\Delta^+_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}]$ ensures $\Sigma^B\text{-}\mathsf{part}[X_{i,j}] = \Sigma^B\text{-}\mathsf{part}[X_{i,j-1}] \cup \Sigma^B\text{-}\mathsf{part}[\Delta^+_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}]$; together with the precondition $\Sigma^B\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^{\mathsf{m}}_{i,j}] = \Sigma^B\text{-}\mathsf{part}[X_{i,j-1}]$, we have $\Sigma^B\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^+_{i,j}] = \Sigma^B\text{-}\mathsf{part}[X_{i,j}]$, as required. Moreover, condition (A1.a), (A2.c), or (A3.f), depending on the type of $C_{i,j}$, ensures $\Delta^+_{i,j} \subseteq \Pi_\infty[E_i]^{\leq s}$; together with property (D.23) and $X_{i,j} = X_{i,j-1} \cup \Sigma^P\text{-}\mathsf{part}[\Delta^+_{i,j} \setminus \Delta^{\mathsf{m}}_{i,j}]$, this ensures the remaining inclusions of property (D.7).

For the $\subseteq$ direction of the equality in (D.8), consider an arbitrary fact $R(u, v) \in \mathsf{fin}[I^q_{i,j}]$. By (D.3), there exists $q \in F^R$ such that $q(u, v) \in I^q_{i,j}$ holds. If $q(u, v) \in I^q_{i,j-1}$, then property (D.24) ensures that $R(u, v)$ belongs to the required set. Otherwise, line 103 ensures that $q(u, v)$ belongs to J^q; but then, line 104 ensures $R(u, v) \in (I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^+_{i,j} \cup J_{i,j}$, as required. For the $\supseteq$ direction of the equality, consider an arbitrary fact $R(u, v) \in \Sigma^H\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^+_{i,j} \cup J_{i,j}]$. We discuss the following three possibilities.

- $R(u, v) \in \Sigma^H\text{-}\mathsf{part}[(I_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^{\mathsf{m}}_{i,j}]$. Then, property (D.24) ensures $R(u, v) \in \mathsf{fin}[I^q_{i,j-1}] \subseteq \mathsf{fin}[I^q_{i,j}]$.
- $R(u, v) \in \Sigma^H\text{-}\mathsf{part}[\Delta^+_{i,j} \setminus \Delta^-_{i,j}]$. Since R is an unfolding of itself, automaton N^R recognises R. Hence, there exists $q \in F^R$ such that $q \in \delta^R(q^R_s, R)$. Moreover, line 102 and Claim 43 ensure $R(u, v) \in \Sigma^H\text{-}\mathsf{part}[\Delta^+_{i,j} \setminus \Delta^-_{i,j}] \subseteq X_{i,j}$. Thus, $q(u, v) \in \mathsf{close}^R[X_{i,j}]$ holds; but then, property (D.6) ensures $q(u, v) \in Q^{\,R}\text{-}\mathsf{part}[I^q_{i,j}]$; together with $q \in F^R$, this ensures $R(u, v) \in \mathsf{fin}[I^q_{i,j}]$.
- $R(u, v) \in \Sigma^H\text{-}\mathsf{part}[J_{i,j}]$. Then, lines 103 and 104 ensure $R(u, v) \in \mathsf{fin}[I^q_{i,j}]$.

To complete the proof of property (D.8), we prove $\text{fin}[I_{i,j}^q] \subseteq \Pi_\infty[E_i]^s$. To this end, consider an arbitrary fact $R(u,v) \in \text{fin}[I_{i,j}^q]$. By (D.3), there exists a state $q \in F^R$ such that $q(u,v) \in I_{i,j}^q$ holds. Moreover, property (D.7), line 102, and Claim 43 then ensure $q(u,v) \in \text{close}^R[X_{i,j}] = \text{close}^R[X_{i,j-1} \cup \Sigma^P\text{-part}[\Delta_{i,j}^+ \setminus \Delta_{i,j}^m]]$. Together with $X_{i,j-1} \subseteq \Pi_\infty[E_i]^{\leq s}$ from precondition (D.23) and $\Delta_{i,j}^+ \subseteq \Pi_\infty[E_i]^{\leq s}$, this ensures $q(u,v) \in \text{close}^R[\Pi_\infty[E_i]^{\leq s}]$. But then, our assumption on $q \in F^R$ and Proposition 27 ensure that we have $R(u,v) \in M_\infty[\Pi_\infty[E_i]^{\leq s}] \subseteq \Pi_\infty[E_i]^s$, as required.

Property (D.9) trivially holds since these sets are empty before the call and they are never updated during the call. We now prove that $C_{i,j}$ is correct. The rightmost inclusion of property (D.8) and line 104 ensure that $J_{i,j}$ satisfies the upper bound. For the lower bound, consider an arbitrary fact $R(u,v) \in M_A[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+]$. Then, there exists an instance of a rule in M of the form

$$S_1(c_0,c_1) \wedge \cdots \wedge S_n(c_{n-1},c_n) \to R(c_0,c_n) \tag{D.25}$$

such that $R(c_0,c_n) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ holds, and $S_k(c_{k-1},c_k) \in (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ holds for each k with $1 \leq k \leq n$. We prove that for each k with $1 \leq k \leq n$, there exist facts $R_{k,1}(c_{k,0},c_{k,1}), \ldots, R_{k,m_k}(c_{k,m_k-1},c_{k,m_k}) \in X_{i,j}$ such that the sequence of predicates $R_{k,1}, \ldots, R_{k,m_k}$ is an unfolding of S_k, and $c_{k,0} = c_{k-1}$ and $c_{k,m_k} = c_k$ both hold. To this end, consider arbitrary k with $1 \leq k \leq n$. There are two cases.

- $S_k \in \Sigma^H$—that is, S_k is a head predicate in M. Then, $S_k(c_{k-1},c_k) \in (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, the equality in property (D.8), and property (D.6) ensure the required property.
- $S_k \in \Sigma^P \setminus \Sigma^H$—that is, S_k is a predicate in M that never appears in the head of a rule. Then, $S_k(c_{k-1},c_k) \in (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ and property (D.7) ensure $S_k(c_{k-1},c_k) \in X_{i,j}$, and so the required property holds.

Either way, the required property holds. The sequence of predicates $R_{1,1}, \ldots, R_{1,m_1}, \ldots, R_{n,1}, \ldots R_{n,m_n}$ is an unfolding of R, and all facts belong to $X_{i,j}$, so there exists $q(u,v) \in \text{close}^R[X_{i,j}]$ such that $q \in F^R$ holds. Then, property (D.6) and property (D.8) ensure $R(u,v) \in \Sigma^H\text{-part}[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \cup J_{i,j}]$; together with $R(u,v) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, this ensures $R(u,v) \in J_{i,j}$. $\quad\square$

Claim 45. *If $C_{i,j}$ is of the form $J_{i,j} := \text{Del}^M[I_{i,j}^o, I_{i,j}^n \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^m]$ and*

- $i \geq 1$, $j = 1$, *and call $C_{i-1,h_{i-1}}$ satisfies properties (N1)–(N4), or*
- $j > 1$ *and call $C_{i,j-1}$ satisfies properties (N1)–(N4),*

then $C_{i,j}$ satisfies properties (N2) and (N4).

Proof. We first capture the preconditions for the call $C_{i,j}$ by proving the following properties.

$$Q^R\text{-part}[I_{i,j-1}^q] \subseteq \text{close}^R[X_{i,j-1}] \subseteq \text{close}^R[X_{i,0}] = Q^R\text{-part}[I_{i,0}^q] \text{ for each } R \in \Sigma^H \tag{D.26}$$

$$\text{ext}^R[\Delta_{i,j-1}^q, X_{i,j-1}] \subseteq Q^R\text{-part}[\Delta_{i,j-1}^q] \subseteq \text{close}^R[X_{i,0}, I_{i,j}^o \setminus I_{i,j}^n] \text{ for each } R \in \Sigma^H \tag{D.27}$$

$$\Sigma^B\text{-part}[I_{i,j}^o \cap I_{i,j}^n] = \Sigma^B\text{-part}[X_{i,j-1}] \subseteq X_{i,j-1} = X_{i,0} \setminus (I_{i,j}^o \setminus I_{i,j}^n) \subseteq I_{i,j}^o \cap I_{i,j}^n \tag{D.28}$$

$$\text{fin}[\text{close}^R[X_{i,0}, I_{i,j}^o \setminus I_{i,j}^n]] = R\text{-part}[(I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^m \cup Y_{i,j-1}] \text{ for each } R \in \Sigma^H \tag{D.29}$$

$$\text{fin}[I_{i,j-1}^q] = \Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1})] \subseteq \Sigma^H\text{-part}[I_{i,j}^n] \subseteq \Sigma^H\text{-part}[I_{i,j}^o] = \text{fin}[I_{i,0}^q] \tag{D.30}$$

$$Y_{i,j-1} \subseteq I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s} \tag{D.31}$$

$$\{u \mid \text{there exist } q \text{ and } v \text{ such that } q(u,v) \in Q^R\text{-part}[I_{i,0}^q \setminus I_{i,j-1}^q]\} \subseteq Z_{i,j-1}^R \text{ for each } R \in \Sigma^H \tag{D.32}$$

$$I_{i,0}^q \setminus I_{i,j-1}^q \subseteq \Delta_{i,j-1}^q, \quad \Delta_{i,j-1}^q \cap I_{i,j-1}^q = \emptyset, \quad \text{and} \quad \text{fin}[\Delta_{i,j-1}^q] \cap (I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1})) = \emptyset \tag{D.33}$$

Towards this goal, call $C_{i,j}$ can be of one of the following two types.

- Assume that $C_{i,j}$ is of type D1. Property (D.6) for $i-1$ and h_{i-1} ensures property (D.26), and $\Delta_{i,j-1}^q = \emptyset$ ensures property (D.27). Furthermore, condition (D1.c), property (D.7) for $i-1$ and h_{i-1}, and $M \subseteq \Pi^s$ ensure $\Sigma^B\text{-part}[I_{i,j}^o] = \Sigma^B\text{-part}[X_{i,j-1}]$; in addition, the rightmost inclusion of (D.7) for $i-1$ and h_{i-1} and condition (D1.d) ensure the remaining part of property (D.28). For property (D.29), both sides of the equality are empty for each $R \in \Sigma^H$, so the property holds. For (D.30), property (D.8) for $i-1$ and h_{i-1} clearly ensures $\text{fin}[I_{i,j-1}^q] = \Sigma^H\text{-part}[(I_{i-1,h_{i-1}} \setminus \Delta_{i-1,h_{i-1}}^-) \cup \Delta_{i-1,h_{i-1}}^+ \cup J_{i-1,h_{i-1}}]$; together with $M \subseteq \Pi^s$ and conditions (D1.c) and (D1.b), this ensures $\text{fin}[I_{i,j-1}^q] = \Sigma^H\text{-part}[I_{i,j}^n]$; moreover, property (D.9) ensures $Y_{i,j-1} = \emptyset$, which together with $\Delta_{i,j}^m = \emptyset$ ensures $\text{fin}[I_{i,j-1}^q] = \Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1})]$,

as required by the leftmost equality in (D.30); in addition, $\mathsf{fin}[I_{i,j-1}^q] = \Sigma^H\text{-part}[I_{i,j}^n]$ and condition (D1.d) ensure $\mathsf{fin}[I_{i,j-1}^q] = \Sigma^H\text{-part}[I_{i,j}^n] = \Sigma^H\text{-part}[I_{i,j}^o]$, so the remaining part of (D.30) holds as well. Property (D.31) trivially holds since $Y_{i,j-1}$ is empty. For property (D.32), consider an arbitrary $R \in \Sigma^H$; property (D.9) for $i - 1$ and h_{i-1} ensures $Z_{i,j-1}^R = \emptyset$, and the right-hand side of the inequality is empty as well, so the property clearly holds. Finally, property (D.33) trivially holds since $\Delta_{i,j-1}^q = I_{i,0}^q \setminus I_{i,j-1}^q = \emptyset$.

- Assume that $C_{i,j}$ is of type D2. Property (D.10) for i and $j - 1$ ensures (D.26). To show property (D.27), consider an arbitrary predicate $R \in \Sigma^H$. Inclusion $\mathsf{ext}^R[\Delta_{i,j-1}^q, X_{i,j-1}] \subseteq Q^R\text{-part}[\Delta_{i,j-1}^q]$ follows from property (D.11) for i and $j - 1$; moreover, $\Delta_{i,j-1}^- \subseteq I_{i,j-1}^n$, $\Delta_{i,j-1}^- \subseteq I_{i,j-1}^o$, $\Delta_{i,j-1}^+ \cap I_{i,j-1}^o = \emptyset$, and conditions (D1.d), (D2.b), and (D2.c) ensure

$$(I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^- = I_{i,j}^o \setminus I_{i,j}^n;$$

then, property (D.11) for i and $j - 1$ ensures $Q^R\text{-part}[\Delta_{i,j-1}^q] \subseteq \mathsf{close}^R[X_{i,0}, I_{i,j}^o \setminus I_{i,j}^n]$, as required. For (D.28), property (D.12) for i and $j - 1$ ensures $\Sigma^B\text{-part}[I_{i,j-1}^o \cap (I_{i,j-1}^n \setminus \Delta_{i,j-1}^-)] = \Sigma^B\text{-part}[X_{i,j-1}]$; then, $\Delta_{i,j-1}^+ \cap I_{i,j-1}^o = \emptyset$, $I_{i,j}^o = I_{i,j-1}^o = \Pi_\infty[E_{i-1}]$, and condition (D2.c) ensure $\Sigma^B\text{-part}[I_{i,j}^o \cap I_{i,j}^n] = \Sigma^B\text{-part}[X_{i,j-1}]$, as required. Property (D.12) for i and $j - 1$ also ensures $X_{i,j-1} = X_{i,0} \setminus ((I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^-) = X_{i,0} \setminus (I_{i,j}^o \setminus I_{i,j}^n) \subseteq I_{i,j}^o \cap I_{i,j}^n$, as required by the remaining part of (D.28). For property (D.29), consider an arbitrary $R \in \Sigma^H$. Property (D.13) for i and $j - 1$ ensures

$$\mathsf{fin}[\mathsf{close}^R[X_{i,0}, (I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^-]] = R\text{-part}[(I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^- \cup J_{i,j-1} \cup Y_{i,j-1}];$$

together with $(I_{i,j-1}^o \setminus I_{i,j-1}^n) \cup \Delta_{i,j-1}^- = I_{i,j}^o \setminus I_{i,j}^n$ and condition (D2.d), this ensures the required property. For (D.30), property (D.14) for i and $j - 1$ ensures

$$\mathsf{fin}[I_{i,j-1}^q] = \Sigma^H\text{-part}[I_{i,j-1}^n \setminus (\Delta_{i,j-1}^- \cup J_{i,j-1} \cup Y_{i,j-1})];$$

set $\Delta_{i,j-1}^+$ contains no fact from stratum s, so we have $\Sigma^H\text{-part}[I_{i,j-1}^n \setminus \Delta_{i,j-1}^-] = \Sigma^H\text{-part}[(I_{i,j-1}^n \setminus \Delta_{i,j-1}^-) \cup \Delta_{i,j-1}^+]$; together with conditions (D2.c) and (D2.d), these observations ensure $\mathsf{fin}[I_{i,j-1}^q] = \Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1})]$, as required in (D.30). Property (D.14) for i and $j - 1$ also ensures $\Sigma^H\text{-part}[I_{i,j-1}^n] \subseteq \Sigma^H\text{-part}[I_{i,j-1}^o] = \mathsf{fin}[I_{i,0}^q]$, which together with $I_{i,j}^o = I_{i,j-1}^o$, and $\Sigma^H\text{-part}[\Delta_{i,j-1}^+] = \emptyset$, and condition (D2.c) ensures $\Sigma^H\text{-part}[I_{i,j}^n] \subseteq \Sigma^H\text{-part}[I_{i,j}^o] = \mathsf{fin}[I_{i,0}^q]$, as required by the remaining part of (D.30). Properties (D.31) and (D.32) follow from (D.15) and (D.16) for the previous call, respectively. Finally, $\Delta_{i,j-1}^q \supseteq I_{i,0}^q \setminus I_{i,j-1}^q$, $\Delta_{i,j-1}^q \cap I_{i,j-1}^q = \emptyset$, and $\mathsf{fin}[\Delta_{i,j-1}^q] \cap (I_{i,j-1}^n \setminus (\Delta_{i,j-1}^- \cup J_{i,j-1} \cup Y_{i,j-1})) = \emptyset$ follow from property (D.17) for i and $j - 1$; then, (D2.c) and (D2.d), and $M \subseteq \Pi^s$ ensure $\mathsf{fin}[\Delta_{i,j-1}^q] \cap (I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1})) = \emptyset$, as required.

This completes our proof for the preconditions of $C_{i,j}$. Next we show that $C_{i,j}$ satisfies (N2) and (N4).

We first prove a property that is useful for establishing (D.10) and (D.11). In particular, we prove property (D.34) for each $R \in \Sigma^H$, where J^q is the value of the set after line 128.

$$\mathsf{ext}^R[\Delta_{i,j-1}^q \cup J^q, X_{i,j-1}] \subseteq Q^R\text{-part}[\Delta_{i,j-1}^q \cup J^q] \tag{D.34}$$

To this end, consider an arbitrary $R \in \Sigma^H$ and an arbitrary q-fact $q(u, v) \in \mathsf{ext}^R[\Delta_{i,j-1}^q \cup J^q, X_{i,j-1}]$. By (D.2), there exist $\ell \geq 0$, states $q_0, \ldots, q_\ell \in Q^R$, binary predicates $R_0, \ldots, R_{\ell-1}$, and constants $w_0, \ldots, w_\ell$ such that $q_\ell = q$, $w_\ell = v$, $q_0(u, w_0) \in \Delta_{i,j-1}^q \cup J^q$, and $q_{i+1} \in \delta^R(q_i, R_i)$ and $R_i(w_i, w_{i+1}) \in X_{i,j-1}$ for $0 \leq i < \ell$. By induction on $0 \leq m \leq \ell$, we show $q_m(u, w_m) \in Q^R\text{-part}[\Delta_{i,j-1}^q \cup J^q]$; then, for $m = \ell$, this ensures $q(u, v) \in Q^R\text{-part}[\Delta_{i,j-1}^q \cup J^q]$. The base case where $m = 0$ holds trivially. For the inductive step, consider m with $0 < m \leq \ell$ such that $q_{m-1}(u, w_{m-1}) \in Q^R\text{-part}[\Delta_{i,j-1}^q \cup J^q]$; we show that $q_m(u, w_m) \in Q^R\text{-part}[\Delta_{i,j-1}^q \cup J^q]$ holds as well. We have two possibilities. If $q_{m-1}(u, w_{m-1}) \in Q^R\text{-part}[\Delta_{i,j-1}^q]$, then precondition (D.27) ensures $q_m(u, w_m) \in Q^R\text{-part}[\Delta_{i,j-1}^q]$. Otherwise, we have $q_{m-1}(u, w_{m-1}) \in Q^R\text{-part}[J^q]$. Each q-fact added to J^q is also added to Q, so $q_{m-1}(u, w_{m-1})$ is added to Q at some point during the execution of the call. Consider the execution of the loop of lines 124–128 when $q_{m-1}(u, w_{m-1})$ is extracted from Q in line 125. Then, $R_{m-1}(w_{m-1}, w_m) \in X_{i,j-1}$ and $q_m \in \delta^R(q_{m-1}, R_{m-1})$ satisfy the condition in line 127; hence, $q_m(u, w_m)$ is considered in line 128, which in turn ensures $q_m(u, w_m) \in Q^R\text{-part}[\Delta_{i,j-1}^q \cup J^q]$, as required.

For property (D.10), we prove $Q^R\text{-part}[I_{i,j}^q] \subseteq \mathsf{close}^R[X_{i,j}]$ for each $R \in \Sigma^H$; then, property (D.26) ensures (D.10). To this end, we consider an arbitrary $R \in \Sigma^H$ and an arbitrary q-fact $q(u, v) \in Q^R\text{-part}[I_{i,j}^q]$, and we show that $q(u, v) \in \mathsf{close}^R[X_{i,j}]$ holds. First, property (D.26) ensures $q(u, v) \in \mathsf{close}^R[X_{i,j-1}]$. Then, by equation (D.1), there exist $\ell \geq 1$, states $q_0, \ldots, q_\ell \in Q^R$, binary predicates $R_0, \ldots, R_{\ell-1}$, constants $w_0, \ldots, w_\ell$, and an integer k with $0 \leq k < \ell$ such that $q_0 = q_s^R$, $w_0 = u$, $q_\ell = q$, $w_\ell = v$, $q_{i+1} \in \delta^R(q_i, R_i)$ for i with $0 \leq i < \ell$, and $R_m(w_m, w_{m+1}) \in X_{i,j-1}$ for each m with $0 \leq m < \ell$. Assume for the sake of a contradiction that $q(u, v) \notin \mathsf{close}^R[X_{i,j}]$ holds. Then, there exists an integer k with $0 \leq k < \ell$ such that

$R_k(w_k, w_{k+1}) \in X_{i,j-1} \setminus X_{i,j}$ holds. Without loss of generality, assume that k is the smallest such integer. The way X is updated in line 135 ensures $R_k(w_k, w_{k+1}) \in \Delta_{i,j}^-$. We discuss the following cases.

- $k = 0$. Then, $R_k(w_k, w_{k+1}) \in \Delta_{i,j}^-$ ensures that $R_k(w_k, w_{k+1})$ is considered in line 119. Then, $q_{k+1} \in \delta^R(q_k, R_k)$, $q_k = q_0 = q_s^R$, and lines 120–121 jointly ensure $q_{k+1}(w_0, w_{k+1}) \in Q^R$-part$[\Delta_{i,j-1}^q \cup J^q]$.

- $k > 0$ and $q_k(w_0, w_k) \in Q^R$-part$[I_{i,j-1}^q]$. Then, $R_k(w_k, w_{k+1}) \in \Delta_{i,j}^-$ ensures that $R_k(w_k, w_{k+1})$ is considered in line 119. Moreover, $q_{k+1} \in \delta^R(q_k, R_k)$, $q_k(w_0, w_k) \in Q^R$-part$[I_{i,j-1}^q]$, and lines 120 and 122 ensure that $q_{k+1}(w_0, w_{k+1})$ is considered in line 123, which in turn ensures $q_{k+1}(w_0, w_{k+1}) \in Q^R$-part$[\Delta_{i,j-1}^q \cup J^q]$.

- $k > 0$ and $q_k(w_0, w_k) \notin Q^R$-part$[I_{i,j-1}^q]$. Then, $q_k(w_0, w_k) \in \text{close}^R[X_{i,j-1}]$ and (D.26) ensure $q_k(w_0, w_k) \in Q^R$-part$[I_{i,0}^q]$; together with $q_k(w_0, w_k) \notin Q^R$-part$[I_{i,j-1}^q]$ and (D.33), this ensures $q_k(w_0, w_k) \in Q^R$-part$[\Delta_{i,j-1}^q]$. But then, $q_{k+1} \in \delta^R(q_k, R_k)$ and precondition (D.27) jointly ensure $q_{k+1}(w_0, w_{k+1}) \in Q^R$-part$[\Delta_{i,j-1}^q]$.

Thus, $q_{k+1}(w_0, w_{k+1}) \in Q^R$-part$[\Delta_{i,j-1}^q \cup J^q]$ holds. But then, property (D.34) for R ensures $q_\ell(w_0, w_\ell) \in Q^R$-part$[\Delta_{i,j-1}^q \cup J^q]$, which in turn ensures $q(u, v) \in Q^R$-part$[\Delta_{i,j-1}^q \cup J^q]$. Together with precondition (D.33) and the way I^q is updated in line 135, this ensures $q(u, v) \notin Q^R$-part$[I_{i,j}^q]$, which is a contradiction. As a result, $q(u, v) \in \text{close}^R[X_{i,j}]$ holds, as required.

For property (D.11), consider an arbitrary $R \in \Sigma^H$. Then, line 135 ensures $X_{i,j} \subseteq X_{i,j-1}$ and $\Delta_{i,j}^q = \Delta_{i,j-1}^q \cup J^q$; together with property (D.34) for R, these ensure $\text{ext}^R[\Delta_{i,j}^q, X_{i,j}] \subseteq Q^R$-part$[\Delta_{i,j}^q]$. We prove Q^R-part$[J^q] \subseteq \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$, which together with Q^R-part$[\Delta_{i,j-1}^q] \subseteq \text{close}^R[X_{i,0}, I_{i,j}^o \setminus I_{i,j}^n]$ from precondition (D.27) and the way Δ^q is updated in line 135 ensures Q^R-part$[\Delta_{i,j}^q] \subseteq \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$. To this end, let N be the total number of iterations for the loop of lines 124–128; let J_0^q and Q_0 be the values of J^q and Q, respectively, before the loop, and for each k with $1 \le k \le N$, let J_k^q and Q_k be the values of J^q and Q, respectively, after the k-th iteration of the loop. We next prove Q^R-part$[J_k^q] \subseteq \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$ by induction on k; then, for $k = N$, we have Q^R-part$[J^q] \subseteq \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$, as required.

- For the induction base $k = 0$, consider an arbitrary q-fact $q(u, v) \in Q^R$-part$[J_0^q]$. Then, $q(u, v)$ is added to J_0^q in line 121 or 123. In the former case, lines 119, 120, and 121 ensure $q(u, v) \in \text{close}^R[X_{i,0}, \Delta_{i,j}^-] \subseteq \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$. In the latter case, lines 119, 120, and 122 ensure that there exists $q_1(u, w) \in Q^R$-part$[I_{i,j-1}^q]$ and $R_1(w, v) \in \Delta_{i,j}^-$ such that $q \in \delta^R(q_1, R_1)$ holds; but then, Q^R-part$[I_{i,j-1}^q] \subseteq \text{close}^R[X_{i,j-1}]$ from precondition (D.26) and $X_{i,j-1} \subseteq X_{i,0}$ from precondition (D.28) ensure $q(u, v) \in \text{close}^R[X_{i,0}, \Delta_{i,j}^-] \subseteq \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$, as required.

- For the inductive step, consider arbitrary k with $1 \le k \le N$ such that Q^R-part$[J_{k-1}^q] \subseteq \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$ holds; we show that Q^R-part$[J_k^q] \subseteq \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$ holds as well. Consider an arbitrary q-fact $q(u, v) \in Q^R$-part$[J_k^q]$. If $q(u, v) \in Q^R$-part$[J_{k-1}^q]$, then the inductive assumption ensures $q(u, v) \in \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$. Otherwise, $q(u, v)$ is added to J_k^q in line 128 during the k-th iteration of the loop. Then, lines 125 and 127 ensure that there exist a q-fact $q_1(u, w) \in Q^R$-part$[Q^{k-1}]$ and $R_1(w, v) \in X_{i,j-1} \subseteq X_{i,0}$ such that $q \in \delta^R(q_1, R_1)$ holds. Each q-fact added to Q^{k-1} is also added to J_{k-1}^q, so by the inductive assumption we have $q_1(u, w) \in \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$, which ensures $q(u, v) \in \text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$, as required.

For property (D.12), we first prove Σ^B-part$[I_{i,j}^o \cap (I_{i,j}^n \setminus \Delta_{i,j}^-)] = \Sigma^B$-part$[X_{i,j}]$. For the $\subseteq$ direction, consider an arbitrary fact $R(u, v) \in \Sigma^B$-part$[I_{i,j}^o \cap (I_{i,j}^n \setminus \Delta_{i,j}^-)]$; then, $R(u, v) \in \Sigma^B$-part$[I_{i,j}^o \cap I_{i,j}^n]$ and precondition (D.28) ensure $R(u, v) \in \Sigma^B$-part$[X_{i,j-1}]$; together with $R(u, v) \notin \Delta_{i,j}^-$ this ensures $R(u, v) \in \Sigma^B$-part$[X_{i,j-1} \setminus \Delta_{i,j}^-]$; then, line 135 ensures $R(u, v) \in \Sigma^B$-part$[X_{i,j}]$, as required. For the $\supseteq$ direction, consider an arbitrary fact $R(u, v) \in \Sigma^B$-part$[X_{i,j}]$; then, the way X is updated in line 135 and precondition (D.28) ensure $R(u, v) \in \Sigma^B$-part$[X_{i,j-1}] \subseteq \Sigma^B$-part$[I_{i,j}^o \cap I_{i,j}^n]$; line 135 also ensures $R(u, v) \notin \Delta_{i,j}^-$, so Σ^B-part$[I_{i,j}^o \cap (I_{i,j}^n \setminus \Delta_{i,j}^-)]$ holds, as required. The remaining part of (D.12) also follows from precondition (D.28) and the way X is updated in line 135.

Next, we prove property (D.13). Consider an arbitrary $R \in \Sigma^H$. For the $\subseteq$ direction, consider an arbitrary fact $R(u, v)$ such that $R(u, v) \in \text{fin}[\text{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]]$. Then, by (D.1) and (D.2), there exist $\ell \ge 1$, states $q_0, \dots, q_\ell \in Q^R$ with $q_\ell \in F^R$, binary predicates $R_0, \dots, R_{\ell-1}$, constants $w_0, \dots, w_\ell$, and an integer k with $0 \le k < \ell$ such that $q_0 = q_s^R$, $w_0 = u$, $w_\ell = v$, $q_{i+1} \in \delta^R(q_i, R_i)$ for $0 \le i < \ell$, and $R_k(w_k, w_{k+1}) \in (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-$ and $R_m(w_m, w_{m+1}) \in X_{i,0}$ for $0 \le m < \ell$ and $m \ne k$. We prove that for each m with $0 \le m < \ell$, there exist facts $\{R_{m,0}(c_{m,0}, c_{m,1}), \dots, R_{m,n_m-1}(c_{m,n_m-1}, c_{m,n_m})\} \subseteq X_{i,0}$ such that the sequence of predicates $R_{m,0} \cdots R_{m,n_m-1}$ is an unfolding of R_m, and $c_{m,0} = w_m$ and $c_{m,n_m} = w_{m+1}$ hold. Note that this property trivially holds for each m with $0 \le m < \ell$ and $m \ne k$. For $m = k$, we discuss the following two possibilities.

If $R_k \in \Sigma^B$, then it is straightforward to see that $R_k(w_k, w_{k+1}) \in \Sigma^B\text{-part}[(I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^-_{i,j}] \subseteq \Sigma^B\text{-part}[I^o_{i,j}] = \Sigma^B\text{-part}[X_{i,0}]$ holds, where the last equality follows either from property (D.7) for $i - 1$ and h_{i-1} or from the same property for the previous call. Otherwise, we have $R_k \in \Sigma^H$, so the last equality in (D.30) and the last equality in (D.26) ensure the required property. Now $R_{m,0} \cdots R_{m,n_m-1}$ is an unfolding of R_m for each m, and $R_0 \cdots R_{\ell-1}$ is an unfolding of R, sequence $R_{0,0} \cdots R_{0,n_0-1} \cdots R_m \cdots R_{\ell-1,0} \cdots R_{\ell-1,n_{\ell-1}-1}$ is clearly an unfolding of R for each $0 \leq m < \ell$; moreover, $R_{m,t}(c_{m,t}, c_{m,t+1}) \in X_{i,0}$ holds for each $0 \leq m < \ell$ and $0 \leq t < n_m$. These observations ensure $R(u, v) \in \text{fin}[\text{close}^R[X_{i,0}]]$, so preconditions (D.26) and (D.30) ensure $R(u, v) \in R\text{-part}[I^o_{i,j}]$. We have the following possibilities.

- There exists m with $0 \leq m < \ell$ such that $R_m(w_m, w_{m+1}) \in (I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^m_{i,j} \cup Y_{i,j-1}$ holds. If $R_m \in \Sigma^B$, then $R_m(w_m, w_{m+1})$ cannot be an element of $\Delta^m_{i,j} \cup Y_{i,j-1}$ since $\Delta^m_{i,j}$ and $Y_{i,j-1}$ only contain derived facts. Thus, we have $R_m(w_m, w_{m+1}) \in I^o_{i,j} \setminus I^n_{i,j}$; together with the fact that $R_{0,0} \cdots R_{0,n_0-1} \cdots R_m \cdots R_{\ell-1,0} \cdots R_{\ell-1,n_{\ell-1}-1}$ is an unfolding of R and that the relevant facts are all in $X_{i,0}$ this ensures $R(u, v) \in \text{fin}[\text{close}^R[X_{i,0}, I^o_{i,j} \setminus I^n_{i,j}]]$; but then, precondition (D.29) ensures that $R(u, v)$ belongs to the required set. Otherwise, we have $R_m \in \Sigma^H$, so precondition (D.29) ensures $R_m(w_m, w_{m+1}) \in \text{fin}[\text{close}^R[X_{i,0}, I^o_{i,j} \setminus I^n_{i,j}]]$, which in turn ensures $R(u, v) \in \text{fin}[\text{close}^R[X_{i,0}, I^o_{i,j} \setminus I^n_{i,j}]]$; hence, $R(u, v)$ belongs to the required set in this case as well.

- There exists no m with $0 \leq m < \ell$ such that $R_m(w_m, w_{m+1}) \in (I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^m_{i,j} \cup Y_{i,j-1}$ holds. Thus, for each m with $0 \leq m < \ell$ and $m \neq k$, we have $R_m(w_m, w_{m+1}) \in X_{i,0} \setminus (I^o_{i,j} \setminus I^n_{i,j})$, and so precondition (D.28) ensures $R_m(w_m, w_{m+1}) \in X_{i,j-1}$. Moreover, $R_k(w_k, w_{k+1}) \in (I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^-_{i,j}$ and $R_k(w_k, w_{k+1}) \notin (I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^m_{i,j} \cup Y_{i,j-1}$ ensure $R_k(w_k, w_{k+1}) \in \Delta^-_{i,j} \setminus (\Delta^m_{i,j} \cup Y_{i,j-1})$. We have the following possibilities.

  - $k = 0$. Then, $R_k(w_k, w_{k+1}) \in \Delta^-_{i,j} \setminus (\Delta^m_{i,j} \cup Y_{i,j-1})$ ensures that fact $R_k(w_k, w_{k+1})$ is considered in line 119. Moreover, $q_{k+1} \in \delta^R(q_k, R_k)$, $q_k = q_0 = q^R_s$, and lines 120–121 ensure $q_{k+1}(w_0, w_{k+1}) \in Q^R\text{-part}[\Delta^q_{i,j-1} \cup J^q]$. Together with $q_{i+1} \in \delta^R(q_i, R_i)$ for $0 \leq i < \ell$, $q_\ell \in F^R$, and precondition (D.27), this ensures that $q_\ell(u, v) \in F^R\text{-part}[\Delta^q_{i,j-1} \cup J^q]$ holds. If $q_\ell(u, v) \in F^R\text{-part}[\Delta^q_{i,j-1}]$, then $R(u, v) \in \text{fin}[\Delta^q_{i,j-1}]$; by (D.33), we have $R(u, v) \notin I^n_{i,j} \setminus (\Delta^m_{i,j} \cup Y_{i,j-1})$; together with $R(u, v) \in I^o_{i,j}$, this ensures $R(u, v) \in (I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^m_{i,j} \cup Y_{i,j-1}$, which is a subset of $(I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^-_{i,j} \cup J_{i,j} \cup Y_{i,j}$. Otherwise, we have $q_\ell(u, v) \in F^R\text{-part}[J^q]$, and so $R(u, v)$ is considered in line 132. If it passes the check, lines 133 and 134 ensure $R(u, v) \in J_{i,j} \cup Y_{i,j}$; if $R(u, v)$ does not pass the check, then $R(u, v) \notin I^n_{i,j} \setminus (\Delta^-_{i,j} \cup J_{i,j} \cup Y_{i,j})$ holds, which together with $R(u, v) \in I^o_{i,j}$ ensures $R(u, v) \in (I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^-_{i,j} \cup J_{i,j} \cup Y_{i,j}$, as required.

  - $k > 0$ and $q_k(w_0, w_k) \in Q^R\text{-part}[I^q_{i,j-1}]$. Then, $R_k(w_k, w_{k+1}) \in \Delta^-_{i,j} \setminus (\Delta^m_{i,j} \cup Y_{i,j-1})$ ensures that fact $R_k(w_k, w_{k+1})$ is considered in line 119. Moreover, $q_{k+1} \in \delta^R(q_k, R_k)$, $q_k(w_0, w_k) \in Q^R\text{-part}[I^q_{i,j-1}]$, and lines 120 and 122 ensure that $q_{k+1}(w_0, w_{k+1})$ is considered in line 123, which in turn ensures $q_{k+1}(w_0, w_{k+1}) \in Q^R\text{-part}[\Delta^q_{i,j-1} \cup J^q]$. In the same way as in the first case, we have $q_\ell(u, v) \in F^R\text{-part}[\Delta^q_{i,j-1} \cup J^q]$. Then, in the same way as in the previous case, we have $R(u, v) \in (I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^-_{i,j} \cup J_{i,j} \cup Y_{i,j}$, as required.

  - $k > 0$ and $q_k(w_0, w_k) \notin Q^R\text{-part}[I^q_{i,j-1}]$. Then, $q_k(w_0, w_k) \in \text{close}^R[X_{i,j-1}]$ and (D.26) imply $q_k(w_0, w_k) \in Q^R\text{-part}[I^q_{i,0}]$; by (D.33) we have $q_k(w_0, w_k) \in Q^R\text{-part}[\Delta^q_{i,j-1}]$; then, precondition (D.27) ensures $q_k(w_0, w_k) \in \text{close}^R[X_{i,0}, I^o_{i,j} \setminus I^n_{i,j}]$. Hence, by (D.1), there exist $k' \geq 1$, states $q'_0, \ldots, q'_{k'} \in Q^R$, predicates $P_0, \ldots, P_{k'-1}$, constants $v_0, \ldots, v_{k'}$, and an integer t with $0 \leq t < k'$ such that $q'_0 = q^R_s$, $q'_{k'} = q_k$, $v_0 = w_0$, $v_{k'} = w_k$, $q'_{i+1} \in \delta^R(q'_i, P_i)$ for each i with $0 \leq i < k'$, and $P_t(v_t, v_{t+1}) \in I^o_{i,j} \setminus I^n_{i,j}$ and $P_m(v_m, v_{m+1}) \in X_{i,0}$ for $0 \leq m < k'$ and $m \neq t$. Then, $q'_0 = q^R_s$, $q'_{i+1} \in \delta^R(q'_i, P_i)$ for each i with $0 \leq i < k'$, $q'_{k'} = q_k$, $q_{i+1} \in \delta^R(q_i, R_i)$ for $k \leq i < \ell$, and $q_\ell \in F^R$ ensure that N^R recognises $P_0 \cdots P_{k'-1} R_k \cdots R_{\ell-1}$. Hence, sequence $P_0 \cdots P_{k'-1} R_k \cdots R_{\ell-1}$ is unfolding of R; since $R_{k,0} \cdots R_{k,n_k-1}$ is an unfolding of R_k, sequence $P_0 \cdots P_{k'-1} R_{k,0} \cdots R_{k,n_k-1} R_{k+1} \cdots R_{\ell-1}$ is an unfolding of R, and so N^R must recognise this sequence. Thus, there exist integer n and states $q''_0, \ldots, q''_n \in Q^R$ with $q''_n \in F^R$ such that $n = k' + n_k + (\ell - k - 1)$, $q''_{i+1} \in \delta^R(q''_i, P_i)$ for each integer i with $0 \leq i < k'$, $q''_{i+1} \in \delta^R(q''_i, R_{k,i-k'})$ for each integer i with $k' \leq i < k' + n_k$, and finally $q''_{i+1} \in \delta^R(q''_i, R_{i-k'-n_k+k+1})$ for each integer i with $k' + n_k \leq i < n$. By $P_t(v_t, v_{t+1}) \in I^o_{i,j} \setminus I^n_{i,j}$, $P_m(v_m, v_{m+1}) \in X_{i,0}$ for $0 \leq m < k'$ and $m \neq t$, we have $q''_{k'}(w_0, w_k) = q''_{k'}(v_0, v_{k'}) \in \text{close}^R[X_{i,0}, I^o_{i,j} \setminus I^n_{i,j}]$. Then, $\{R_{k,0}(c_{k,0}, c_{k,1}), \ldots, R_{k,n_k-1}(c_{k,n_k-1}, c_{k,n_k})\} \subseteq X_{i,0}$, $c_{k,0} = w_k$, and $c_{k,n_k} = w_{k+1}$ imply $q''_{k'+n_k}(w_0, w_{k+1}) \in \text{close}^R[X_{i,0}, I^o_{i,j} \setminus I^n_{i,j}]$. Finally, $\{R_{k+1}(w_{k+1}, w_{k+2}), \ldots, R_{\ell-1}(w_{\ell-1}, w_\ell)\} \subseteq X_{i,0}$ ensures $q''_n(u, v) \in \text{close}^R[X_{i,0}, I^o_{i,j} \setminus I^n_{i,j}]$; together with $q''_n \in F^R$ this ensures $R(u, v) \in \text{fin}[\text{close}^R[X_{i,0}, I^o_{i,j} \setminus I^n_{i,j}]]$; by precondition (D.29), fact $R(u, v)$ belongs to the required set.

This completes our proof of the $\subseteq$ direction of the property. For the $\supseteq$ direction, consider an arbitrary fact $R(u, v)$ such that $R(u, v) \in R\text{-part}[(I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^-_{i,j} \cup J_{i,j} \cup Y_{i,j}]$. If $R(u, v) \in R\text{-part}[(I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^m_{i,j} \cup Y_{i,j-1}]$, then precondition (D.29)

ensures $R(u, v) \in \mathsf{fin}[\mathsf{close}^R[X_{i,0}, I_{i,j}^o \setminus I_{i,j}^n]] \subseteq \mathsf{fin}[\mathsf{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]]$, as required. Otherwise, it is straightforward to see that $R(u, v) \in R\text{-part}[(\Delta_{i,j}^- \setminus \Delta_{i,j}^m) \cup J_{i,j} \cup (Y_{i,j} \setminus Y_{i,j-1})]$ holds, and we discuss the following two cases.

- $R(u, v) \in R\text{-part}[\Delta_{i,j}^- \setminus \Delta_{i,j}^m]$. Then, since R is an unfolding of itself, there exists a state $q \in F^R$ such that $q \in \delta^R(q_s^R, R)$ holds; hence, we have $q(u, v) \in \mathsf{close}^R[X_{i,0}, \Delta_{i,j}^- \setminus \Delta_{i,j}^m] \subseteq \mathsf{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]$; together with $q \in F^R$, this ensures $R(u, v) \in \mathsf{fin}[\mathsf{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]]$, as required.

- $R(u, v) \in R\text{-part}[J_{i,j} \cup (Y_{i,j} \setminus Y_{i,j-1})]$. Then, $R(u, v)$ must be considered in line 132 so, due to line 129, there exists $q \in F^R$ such that $q(u, v) \in J^q$ holds. Therefore, line 135 ensures $q(u, v) \in \Delta_{i,j}^q$, and so $q \in F^R$ and property (D.11) ensure that $R(u, v) \in \mathsf{fin}[\mathsf{close}^R[X_{i,0}, (I_{i,j}^o \setminus I_{i,j}^n) \cup \Delta_{i,j}^-]]$ holds, as required.

To see that property (D.14) holds, we prove $\mathsf{fin}[I_{i,j}^q] = \Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})]$; the rest of property (D.14) follows directly from precondition (D.30). For the $\subseteq$ direction, we prove $\Sigma^H\text{-part}[(\Delta_{i,j}^- \setminus \Delta_{i,j}^m) \cup J_{i,j} \cup (Y_{i,j} \setminus Y_{i,j-1})] \subseteq \mathsf{fin}[\Delta_{i,j-1}^q \cup J^q]$; then, preconditions (D.30) and (D.33), and line 135 ensure $\mathsf{fin}[I_{i,j}^q] \subseteq \Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})]$. To this end, consider an arbitrary fact $R(u, v) \in \Sigma^H\text{-part}[(\Delta_{i,j}^- \setminus \Delta_{i,j}^m) \cup J_{i,j} \cup (Y_{i,j} \setminus Y_{i,j-1})]$. If $R(u, v) \in \Sigma^H\text{-part}[\Delta_{i,j}^- \setminus \Delta_{i,j}^m]$, then $R(u, v)$ is considered in line 119; moreover, R is an unfolding of itself, so N^R recognises R—that is, there exists $q \in F^R$ such that $q \in \delta^R(q_s^R, R)$ holds; but then, lines 120 and 121 ensure $q(u, v) \in F^R\text{-part}[\Delta_{i,j-1}^q \cup J^q]$, as required. If $R(u, v) \in \Sigma^H\text{-part}[J_{i,j} \cup (Y_{i,j} \setminus Y_{i,j-1})]$, then the fact is considered in either line 133 or line 134; either way, lines 129–132 ensure $R(u, v) \in \mathsf{fin}[J^q]$, as required. For the $\supseteq$ direction (i.e., $\Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})] \subseteq \mathsf{fin}[I_{i,j}^q]$), consider an arbitrary fact $R(u, v) \in \Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})]$; then, $\Delta_{i,j}^m \subseteq \Delta_{i,j}^-$, $Y_{i,j-1} \subseteq Y_{i,j}$, and precondition (D.30) ensure $R(u, v) \in \mathsf{fin}[I_{i,j-1}^q]$. Assume for the sake of a contradiction that $R(u, v) \in \mathsf{fin}[J^q]$ holds. Then, there exists $q \in F^R$ such that $q(u, v) \in J^q$ holds, and so lines 129 and 130 ensure that $R(u, v)$ is considered in line 132; our assumption $R(u, v) \in \Sigma^H\text{-part}[I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})]$ ensures that $R(u, v)$ passes the check in line 132; hence, $R(u, v)$ is added either to Y in line 133 or to J in line 134, which is a contradiction. Therefore, $R(u, v) \notin \mathsf{fin}[J^q]$ holds, and so $R(u, v) \in \mathsf{fin}[I_{i,j-1}^q]$ and line 135 ensure $R(u, v) \in \mathsf{fin}[I_{i,j}^q]$, as required.

For property (D.15), consider an arbitrary fact $R(u, v) \in Y_{i,j}$. If $R(u, v) \in Y_{i,j-1}$, then (D.31) ensures $R(u, v) \in I_{i,j}^o \cap \Pi_\infty[E_i]^{\leq s}$, as required. Otherwise, we have $R(u, v) \in Y_{i,j} \setminus Y_{i,j-1}$, and so $R(u, v)$ is added to Y in line 133. Since the oracle function returns true, we clearly have $R(u, v) \in \Pi_\infty[E_i]$; line 130 ensures $R \in \Sigma^H$, and so $M \subseteq \Pi^s$ ensures $R(u, v) \in \Pi_\infty[E_i]^{\leq s}$. Moreover, line 132 ensures $R(u, v) \in \Sigma^H\text{-part}[I_{i,j}^n]$, which together with $\Sigma^H\text{-part}[I_{i,j}^n] \subseteq \Sigma^H\text{-part}[I_{i,j}^o]$ from precondition (D.30) ensures $R(u, v) \in I_{i,j}^o$.

For property (D.16), consider arbitrary $R \in \Sigma^H$ and u for which there exist q and v such that $q(u, v) \in Q^R\text{-part}[I_{i,0}^q \setminus I_{i,j}^q]$. If $q(u, v) \in Q^R\text{-part}[I_{i,0}^q \setminus I_{i,j-1}^q]$, then precondition (D.32) ensures $u \in Z_{i,j-1}^R \subseteq Z_{i,j}^R$. Otherwise, $q(u, v) \in Q^R\text{-part}[I_{i,j-1}^q \setminus I_{i,j}^q]$ holds, so line 135 ensures $q(u, v) \in Q^R\text{-part}[J^q]$. As a result, lines 129–131 ensure $u \in Z_{i,j}^R$, as required.

For property (D.17), properties $\Delta_{i,j}^q \supseteq I_{i,j}^o \setminus I_{i,j}^q$ and $\Delta_{i,j}^q \cap I_{i,j}^q = \emptyset$ clearly follow from precondition (D.33) and the way I^q and Δ^q are updated in line 135. To see that $\mathsf{fin}[\Delta_{i,j}^q] \cap (I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})) = \emptyset$ holds, consider an arbitrary fact $R(u, v) \in \mathsf{fin}[\Delta_{i,j}^q]$. If $R(u, v) \in \mathsf{fin}[\Delta_{i,j-1}^q]$, then precondition (D.33) ensures $R(u, v) \notin I_{i,j}^n \setminus (\Delta_{i,j}^m \cup Y_{i,j-1})$; together with $\Delta_{i,j}^m \subseteq \Delta_{i,j}^-$ and $Y_{i,j-1} \subseteq Y_{i,j}$ this ensures $R(u, v) \notin I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})$, as required. Otherwise, we have $R(u, v) \in \mathsf{fin}[J^q]$, and so $R(u, v)$ must be considered in line 132, which clearly ensures $R(u, v) \notin I_{i,j}^n \setminus (\Delta_{i,j}^- \cup J_{i,j} \cup Y_{i,j})$.

Finally, we show that $C_{i,j}$ is correct. To see that $J_{i,j}$ satisfies the upper bound, consider an arbitrary fact $R(u, v) \in J_{i,j}$. Then, lines 130 and 132 ensure $R(u, v) \in \Sigma^H\text{-part}[I_{i,j}^n \setminus \Delta_{i,j}^-]$; together with $\Sigma^H\text{-part}[I_{i,j}^n] \subseteq \Sigma^H\text{-part}[I_{i,j}^o]$ from property (D.14), this ensures $R(u, v) \in \Sigma^H\text{-part}[I_{i,j}^o]$; moreover, $M \subseteq \Pi^s$ ensures $R(u, v) \in O^s$; thus, we have $R(u, v) \subseteq I_{i,j}^o \cap O^s \cap ((I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+)$, as required. To show that $J_{i,j}$ also satisfies the lower bound, consider an arbitrary fact $R(u, v) \in M_\mathsf{D}[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}^-, \Delta_{i,j}^+] \setminus \Pi_\infty[E_i]$. Then, there exists an instance of a rule in M of the form

$$S_1(c_0, c_1) \wedge \cdots \wedge S_n(c_{n-1}, c_n) \rightarrow R(c_0, c_n) \tag{D.35}$$

such that $u = c_0$, $v = c_n$, $R(c_0, c_n) \in (I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, $S_m(c_{m-1}, c_m) \in I_{i,j}^o \cap I_{i,j}^n$ holds for each m with $1 \leq m \leq n$, and there exists k with $1 \leq k \leq n$ such that $S_k(c_{k-1}, c_k) \notin (I_{i,j}^n \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ holds. Then, properties $S_k(c_{k-1}, c_k) \notin I_{i,j}^n$ and $S_k(c_{k-1}, c_k) \notin I_{i,j}^n \setminus \Delta_{i,j}^-$ ensure $S_k(c_{k-1}, c_k) \in \Delta_{i,j}^-$. Next we prove that, for each m with $1 \leq m \leq n$ and $m \neq k$, there exist a total number of ℓ_m facts $R_{m,1}(c_{m,0}, c_{m,1}), \ldots, R_{m,\ell_m}(c_{m,\ell_m-1}, c_{m,\ell_m}) \in X_{i,0}$ such that the sequence of predicates $R_{m,1} \cdots R_{m,\ell_m}$ is an unfolding of S_m, and $c_{m,0} = c_{m-1}$ and $c_{m,\ell_m} = c_m$ both hold. To this end, consider arbitrary m with $1 \leq m \leq n$. There are two cases.

- $S_m \in \Sigma^H$ —that is, S_m is a head predicate in M. Then, $S_m(c_{m-1}, c_m) \in I^o_{i,j}$, the rightmost equality in (D.14), and the rightmost equality in (D.10) ensure that there exists state $q \in F^R$ such that $q(u, v) \in \text{close}^R[X_{i,0}]$, so the required property holds.
- $S_m \in \Sigma^B$ —that is, S_m is a predicate in M but it does not appear in the head of any rule. Then, $S_m(c_{m-1}, c_m) \in \Sigma^B$-part$[I^o_{i,j}]$ ensures $S_m(c_{m-1}, c_m) \in X_{i,0}$; hence, the required property holds as well since S_m is an unfolding of itself.

Since $S_1 \cdots S_n$ is an unfolding of R and, for each m with $1 \leq m \leq n$ and $m \neq k$, sequence $R_{m,1} \cdots R_{m,\ell_m}$ is an unfolding of S_m, it is straightforward to see that sequence $R_{1,1} \cdots R_{1,\ell_1} \cdots R_{k-1,1} \cdots R_{k-1,\ell_{k-1}} S_k R_{k+1,1} \cdots R_{k+1,\ell_{k+1}} \cdots R_{n,1} \cdots R_{n,\ell_n}$ is an unfolding of R. Hence, N^R recognises this sequence; moreover, $\{R_{m,1}(c_{m,0}, c_{m,1}), \ldots, R_{m,\ell_m}(c_{m,\ell_m-1}, c_{m,\ell_m})\} \subseteq X_{i,0}$ holds for each m with $1 \leq m \leq n$ and $m \neq k$, and $S_k(c_{k-1}, c_k) \in \Delta^-_{i,j}$ holds; jointly we have $R(u, v) \in \text{fin}[\text{close}^R[X_{i,0}, \Delta^-_{i,j}]]$. But then, property (D.13) ensures $R(u, v) \in (I^o_{i,j} \setminus I^n_{i,j}) \cup \Delta^+_{i,j} \cup J_{i,j} \cup Y_{i,j}$; moreover, $\Delta^+_{i,j}$ contains no fact from stratum s, so $R(u, v) \in (I^n_{i,j} \setminus \Delta^-_{i,j}) \cup \Delta^+_{i,j}$ ensures $R(u, v) \in I^n_{i,j} \setminus \Delta^-_{i,j}$; furthermore, $R(u, v) \notin \Pi_\infty[E_i]$ ensures $R(u, v) \notin Y_{i,j}$; together, we have $R(u, v) \in J_{i,j}$, as required. □

Claim 46. *If call $C_{i,j}$ is of type* (R) *and call $C_{i,j-1}$ satisfies properties* (N1)–(N4), *then call $C_{i,j}$ satisfies properties* (N3) *and* (N4).

Proof. For property (D.18), we show that Q^R-part$[I^q_{i,j-1}] \cup \text{ext}^R[Q, X_{i,j-1}] = \text{close}^R[X_{i,j-1}]$ and $\text{ext}^R[B, X_{i,j-1}] \subseteq Q^R$-part$[I^q_{i,j-1}]$ hold for each $R \in \Sigma^H$, where Q and B are the values of the second and third arguments, respectively, of the AddEdges call made in line 140. Then, Claim 43 and the way I^q is updated in line 141 jointly ensure property (D.18). Consider an arbitrary $R \in \Sigma^H$. Property $\text{ext}^R[B, X_{i,j-1}] \subseteq Q^R$-part$[I^q_{i,j-1}]$ trivially holds since B is empty. Next we show that (D.36) holds.

$$Q^R\text{-part}[I^q_{i,j-1}] \cup \text{ext}^R[Q, X_{i,j-1}] = \text{close}^R[X_{i,j-1}] \tag{D.36}$$

For the $\subseteq$ direction, property (D.10) holds for i and $j-1$ by the inductive assumption, so we have Q^R-part$[I^q_{i,j-1}] \subseteq \text{close}^R[X_{i,j-1}]$; moreover, lines 138–139 ensure Q^R-part$[Q] \subseteq \text{close}^R[X_{i,j-1}]$, so we have $\text{ext}^R[Q, X_{i,j-1}] \subseteq \text{close}^R[X_{i,j-1}]$. Consequently, the $\subseteq$ direction of (D.36) holds. For the $\supseteq$ direction, we prove $\text{close}^R[X_{i,j-1}] \setminus Q^R$-part$[I^q_{i,j-1}] \subseteq \text{ext}^R[Q, X_{i,j-1}]$. To this end, consider an arbitrary $q(u, v) \in \text{close}^R[X_{i,j-1}] \setminus Q^R$-part$[I^q_{i,j-1}]$. Then, by (D.1), there exist facts $\{R_0(w_0, w_1), \ldots, R_{\ell-1}(w_{\ell-1}, w_\ell)\} \subseteq X_{i,j-1}$ such that $u = w_0$ and $v = w_\ell$, and states $\{q_0, \ldots, q_\ell\} \subseteq Q^R$ such that $q^R_s = q_0$ and $q = q_\ell$, and $q_{i+1} \in \delta^R(q_i, R_i)$ for $0 \leq i < \ell$. Moreover, properties (D.10) and (D.12) for i and $j-1$ ensure $q(u, v) \in Q^R$-part$[I^q_{i,0} \setminus I^q_{i,j-1}]$; together with (D.16) for i and $j-1$, this ensures $u \in Z^R_{i,j-1}$. Hence, the conditions in line 138 are satisfied, and $q_1(w_0, w_1)$ is added to Q in line 139. Together with $R_i(w_i, w_{i+1}) \in X_{i,j-1}$ and $q_{i+1} \in \delta^R(q_i, R_i)$ for $1 \leq i < \ell$ this ensures $q(u, v) = q_\ell(w_0, w_\ell) \in \text{ext}^R[Q, X_{i,j-1}]$, as required.

Next, we prove that property (D.19) holds. Property (D.12) holds for i and $j-1$ by the inductive assumption, so we clearly have Σ^B-part$[I^o_{i,j-1} \cap (I^n_{i,j-1} \setminus \Delta^-_{i,j-1})] = \Sigma^B$-part$[X_{i,j-1}] \subseteq X_{i,j-1} \subseteq I^o_{i,j-1} \cap (I^n_{i,j-1} \setminus \Delta^-_{i,j-1})$. Call $C_{i,j-1}$ is of type (D1) or (D2); either way, $\Delta^+_{i,j-1}$ is disjoint from $I^o_{i,j-1}$, and so we have Σ^B-part$[I^o_{i,j-1} \cap ((I^n_{i,j-1} \setminus \Delta^-_{i,j-1}) \cup \Delta^+_{i,j-1})] = \Sigma^B$-part$[X_{i,j-1}]$; moreover, conditions (D1.d), (D2.b), and (R.c) ensure $I^o_{i,j} = I^o_{i,j-1}$; together with condition (R.d), this ensures Σ^B-part$[I^o_{i,j} \cap I^n_{i,j}] = \Sigma^B$-part$[X_{i,j-1}]$; furthermore, line 140 and Claim 43 ensure $X_{i,j} = X_{i,j-1} \cup Y_{i,j-1}$, but $Y_{i,j-1}$ contains only derived facts, so it is straightforward to see that Σ^B-part$[X_{i,j}] = \Sigma^B$-part$[X_{i,j-1}]$ holds; putting it all together, we have Σ^B-part$[I^o_{i,j} \cap I^n_{i,j}] = \Sigma^B$-part$[X_{i,j}]$, as required in (D.19). Finally, we prove $X_{i,j} \subseteq \Pi_\infty[E_i]^{\leq s}$. To this end, $X_{i,j-1} \subseteq I^o_{i,j-1} \cap (I^n_{i,j-1} \setminus \Delta^-_{i,j-1})$ and condition (R.d) ensure $X_{i,j-1} \subseteq I^n_{i,j}$; set $X_{i,j-1}$ contains only facts from stratum s or lower, so condition (R.e) ensures $X_{i,j-1} \subseteq \Pi_\infty[E_i]^{\leq s}$; moreover, $Y_{i,j-1} \subseteq \Pi_\infty[E_i]^{\leq s}$ holds by property (D.15) for i and $j-1$; thus, we have $X_{i,j} = X_{i,j-1} \cup Y_{i,j-1} \subseteq \Pi_\infty[E_i]^{\leq s}$, as required.

For the leftmost inclusion of (D.20), we consider an arbitrary fact $R(u, v) \in \Sigma^H$-part$[I^n_{i,j} \cup J_{i,j}]$ and consider the following possibilities.

- $R(u, v) \in \Sigma^H$-part$[I^n_{i,j} \setminus Y_{i,j-1}]$. Then, condition (R.d) ensures $R(u, v) \in \Sigma^H$-part$[((I^n_{i,j-1} \setminus \Delta^-_{i,j-1}) \cup \Delta^+_{i,j-1}) \setminus Y_{i,j-1}]$; predicate R belongs to stratum s, and $\Delta^+_{i,j-1}$ contains no fact from stratum s, so we have $R(u, v) \in \Sigma^H$-part$[I^n_{i,j-1} \setminus (\Delta^-_{i,j-1} \cup Y_{i,j-1})]$; together with condition (R.b), this ensures $R(u, v) \in \Sigma^H$-part$[I^n_{i,j-1} \setminus (\Delta^-_{i,j-1} \cup J_{i,j-1} \cup Y_{i,j-1})]$. Property (D.14) holds for i and $j-1$ by the inductive assumption, so we have $R(u, v) \in \text{fin}[I^q_{i,j-1}]$; then, the way I^q is updated in line 141 ensures $R(u, v) \in \text{fin}[I^q_{i,j}]$, as required.
- $R(u, v) \in \Sigma^H$-part$[Y_{i,j-1}]$. Since R is an unfolding of itself, automaton N^R recognises R. Hence, there exists $q \in F^R$ such that $q \in \delta^R(q^R_s, R)$. Moreover, line 140 and Claim 43 jointly ensure $R(u, v) \in \Sigma^H$-part$[Y_{i,j-1}] \subseteq \Sigma^H$-part$[X_{i,j}]$. Consequently, we have $q(u, v) \in \text{close}^R[X_{i,j}]$; but then, property (D.18) ensures $q(u, v) \in Q^R$-part$[I^q_{i,j}]$; together with $q \in F^R$, this ensures $R(u, v) \in \text{fin}[I^q_{i,j}]$, as required.

- $R(u, v) \in \Sigma^H$-part$[J_{i,j}]$. Then, line 142 ensures $R(u, v) \in \mathsf{fin}[I_{i,j}^q]$, as required.

For the second inclusion of (D.20), consider an arbitrary fact $R(u, v)$ such that there exists $q(u, v) \in F^R$-part$[I_{i,j}^q]$. Then, property (D.18) and Claim 43 ensure $q(u, v) \in \mathsf{close}^R[X_{i,j}] = \mathsf{close}^R[X_{i,j-1} \cup Y_{i,j-1}]$. Together with properties (D.12) and (D.15) for i and $j - 1$, we have $q(u, v) \in \mathsf{close}^R[\Pi_\infty[E_{i-1}]^{\leq s}]$. Then, $q \in F^R$ and Proposition 27 ensure $R(u, v) \in M_\infty[\Pi_\infty[E_{i-1}]^{\leq s}] \subseteq \Pi_\infty[E_{i-1}]$; together with condition (R.c), we have $R(u, v) \in I_{i,j}^o$. Now if $R(u, v) \notin \Delta_{i,j}$, then fact $R(u, v)$ clearly belongs to the required set. Otherwise, $R(u, v) \in \Delta_{i,j}$, $q(u, v) \in F^R$-part$[I_{i,j}^q]$, and line 142 ensure $R(u, v) \in J_{i,j}$.

For the third inclusion of (D.20), consider an arbitrary fact $R(u, v) \in \Sigma^H$-part$[(I_{i,j}^o \setminus \Delta_{i,j}) \cup J_{i,j}]$. If $R(u, v) \in \Sigma^H$-part$[I_{i,j}^o \setminus \Delta_{i,j}]$, then $M \subseteq \Pi^s$ and conditions (R.c), (R.e), and (R.f) ensure $R(u, v) \in \Pi_\infty[E_i]^s$. Otherwise, we have $R(u, v) \in \Sigma^H$-part$[J_{i,j}]$; then, due to line 142, there exists $q(u, v) \in F^R$-part$[I_{i,j}^q]$. Property (D.18) ensures $q(u, v) \in \mathsf{close}^R[X_{i,j}]$, which together with the rightmost inclusion of (D.19) ensures $q(u, v) \in \mathsf{close}^R[\Pi_\infty[E_i]^{\leq s}]$. In the same way as above, we have $R(u, v) \in \Pi_\infty[E_i]^s$, as required.

Property (D.21) trivially holds due to line 141. We next prove that $C_{i,j}$ is correct. Property (D.20) ensures $J_{i,j} \subseteq \Pi_\infty[E_i]^s$; moreover, line 142 ensures $J_{i,j} \subseteq \Delta_{i,j}$, which together with condition (R.f) ensures $J_{i,j} \cap I_{i,j}^n = \emptyset$; hence, $J_{i,j} \subseteq \Pi_\infty[E_i]^s \setminus I_{i,j}^n$ holds, and so $J_{i,j}$ satisfies the upper bound. For the lower bound, consider an arbitrary fact $R(u, v) \in M_R[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}]$. Then, there exists an instance of a rule in Π_{rch} of the form

$$S_1(c_0, c_1) \wedge \cdots \wedge S_n(c_{n-1}, c_n) \to R(c_0, c_n) \tag{D.37}$$

such that $u = c_0$, $v = c_n$, $R(c_0, c_n) \in \Delta_{i,j}$, and $S_k(c_{k-1}, c_k) \in I_{i,j}^o \cap I_{i,j}^n$ holds for each k with $1 \leq k \leq n$. We prove that, for each k with $1 \leq k \leq n$, there exist facts $\{R_{k,1}(c_{k,0}, c_{k,1}), \ldots, R_{k,m_k}(c_{k,m_k-1}, c_{k,m_k})\} \subseteq X_{i,j}$ such that the sequence of predicates $R_{k,1}, \ldots, R_{k,m_k}$ is an unfolding of S_k, and $c_{k,0} = c_{k-1}$ and $c_{k,m_k} = c_k$ both hold. To this end, consider arbitrary k with $1 \leq k \leq n$. We have the following two cases.

- $S_k \in \Sigma^H$—that is, S_k is a head predicate in M. Then, $S_k(c_{k-1}, c_k) \in I_{i,j}^n$, the leftmost inclusion of property (D.20), and property (D.18) ensure the required property.
- $S_k \in \Sigma^B$—that is, S_k is a predicate in M but it does not appear in the head of any rule. Then, $S_k(c_{k-1}, c_k) \in \Sigma^B$-part$[I_{i,j}^o \cap I_{i,j}^n]$ and property (D.19) ensure $S_k(c_{k-1}, c_k) \in X_{i,j}$. Since S_k is an unfolding of itself, the required property holds.

Since the sequence of predicates $R_{1,1}, \ldots, R_{1,m_1}, \ldots, R_{n,1}, \ldots R_{n,m_n}$ is an unfolding of R and all the facts belong to $X_{i,j}$, there exists $q(u, v) \in \mathsf{close}^R[X_{i,j}]$ such that $q \in F^R$ holds. Then, property (D.18) ensures $q(u, v) \in F^R$-part$[I_{i,j}^q]$; together with $R(u, v) \in \Delta_{i,j}$ and line 142, this ensures $R(u, v) \in J_{i,j}$, as required. $\square$

Appendix E. Proof of Theorem 33

Theorem 33. *Functions* $\mathsf{Add}^{seq(P,R,<)}$, $\mathsf{Del}^{seq(P,R,<)}$, *and* $\mathsf{Red}^{seq(P,R,<)}$ *are correct.*

Consider an arbitrary unary predicate P, binary predicates R and $<$, program Π, stratification λ of Π, stratum index s such that $seq(P, R, <) \subseteq \Pi^s$, sequence of datasets $E_0, \ldots, E_m$ of datasets, and a call history H for $seq(P, R, <)$ that is compatible with Π, λ, s and $E_0, \ldots, E_m$. We assume that H is of the form as specified in Definition 12. We also assume that the implicit assumptions from Definition 32 are satisfied; this clearly ensures that R and $<$ do not occur in a rule head in $\Pi \setminus seq(P, R, <)$. Also, for λ to be a valid stratification, predicates P and $<$ must be assigned to a stratum with index less than s. For each $0 \leq i \leq m$ and each $1 \leq j \leq h_i$, let $S_{i,j}^P$ be the values of S^P after call $C_{i,j}$. Furthermore, we define $S_{0,0}^P = \emptyset$ and, for $1 \leq i \leq m$, we let $S_{i,0}^P = S_{i-1,h_{i-1}}^P$. Finally, to simplify the notation, let $M = seq(P, R, <)$ for the rest of this section.

We prove Theorem 33 by showing that, for each $0 \leq i \leq m$ and each $1 \leq j \leq h_i$, call $C_{i,j}$ in H satisfies properties (N1)–(N3).

(N1) If call $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Add}^M[I_{i,j} : \Delta_{i,j}^-, \Delta_{i,j}^+ : \Delta_{i,j}^m]$, then the following property holds.

$$S_{i,j}^P = \{b \mid P(b) \in \Pi_\infty[E_i]\} \tag{E.1}$$

(N2) If call $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Del}^M[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}^-, \Delta_{i,j}^+ : \Delta_{i,j}^m]$ or $J_{i,j} := \mathsf{Red}^M[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}]$, then the following property holds.

$$S_{i,j}^P = \{b \mid P(b) \in \Pi_\infty[E_{i-1}] \cap \Pi_\infty[E_i]\} \tag{E.2}$$

(N3) Call $C_{i,j}$ is correct.

We prove this claim by double induction on i and j, where we consider each call $C_{i,j}$ of type shown in Table 2. The base case involves a call of type A1, and the inductive step involves all remaining calls. We structure our proof into five claims.

Claim 47. *If call $C_{i,j}$ is of type A1, or of type A3 and call $C_{i,j-1}$ satisfies properties (N1)–(N3), then $C_{i,j}$ satisfies properties (N1) and (N3).*

Proof. Assume that $C_{i,j}$ is of the form $J_{i,j} := \mathsf{Add}^M[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+ \vdots \Delta_{i,j}^{\mathsf{m}}]$. We next define set E' and show that properties (E.3) and (E.5) are established either at the beginning of the algorithm or by the preceding call $C_{i,j-1}$.

$$S_{i,j-1}^P = \{b \mid P(b) \in \Pi_\infty[E'] \cap \Pi_\infty[E_i]\} \tag{E.3}$$

$$I_{i,j} = \Pi_\infty[E'] \tag{E.4}$$

$$[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \cap \mathsf{O}^{<s} = \Pi_\infty[E_i]^{<s} \tag{E.5}$$

Towards this goal, call $C_{i,j}$ can be of one of the following three types.

- Assume that $C_{i,j}$ is of type A1. We define $E' = \emptyset$. Properties (E.3) and (E.4) hold trivially, and property (E.5) is implied by property (A1.a).
- Assume that $C_{i,j}$ is of type A3. We define $E' = E_{i-1}$. Call $C_{i,j-1}$ is of the form $J_{i,j-1} := \mathsf{Red}^M[I_{i,j-1}^{\mathsf{o}}, I_{i,j-1}^{\mathsf{n}} \vdots \Delta_{i,j-1}]$, so property (E.2) holds for i and $j-1$ by the inductive assumption, and it implies property (E.3). Moreover, properties (A3.b), (A3.c), and (A3.d) imply properties (E.4) and (E.5).

We are now ready to prove that call $C_{i,j}$ satisfies properties (N1) and (N3).

For the $\subseteq$ direction of property (E.1), consider an arbitrary constant $b \in S_{i,j}^P$. If $b \in S_{i,j-1}^P$ holds already, then property (E.3) ensures $P(b) \in \Pi_\infty[E'] \cap \Pi_\infty[E_i] \subseteq \Pi_\infty[E_i]$, as required. Otherwise, we have $b \in S_{i,j}^P \setminus S_{i,j-1}^P$, so b is added to $S_{i,j}^P$ in line 144, which ensures $P(b) \in \Delta_{i,j}^+$; but then, property (E.5) and $P(b) \in \mathsf{O}^{<s}$ ensure $P(b) \in \Pi_\infty[E_i]$. For the $\supseteq$ direction of property (E.1), consider an arbitrary constant b such that $P(b) \in \Pi_\infty[E_i]$. If $P(b) \in \Pi_\infty[E']$, then property (E.3) ensures $b \in S_{i,j-1}^P \subseteq S_{i,j}^P$, as required. Otherwise, we have $P(b) \in \Pi_\infty[E_i] \setminus \Pi_\infty[E']$; property (E.4) ensures $I_{i,j} = \Pi_\infty[E']$; and property (E.5) ensures $P(b) \in (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$; together, these observations imply $P(b) \in \Delta_{i,j}^+$, in which case line 144 ensures $b \in S_{i,j}^P$, as required.

We next prove $J_{i,j} = M_{\mathsf{A}}[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+]$, which ensures that call $C_{i,j}$ is correct. For the $\subseteq$ direction, consider an arbitrary fact $R(e, f) \in J_{i,j}$, and let r' be the instance of rule (50) where x is mapped to e and y is mapped to f.

- Assume $R(e, f)$ is added to $J_{i,j}$ in line 146. Then, lines 145 and 146 ensure $P(f) \in \Delta_{i,j}^+$ and $e \in S_{i,j}^P$; moreover, properties (E.1) and (E.5) ensure $P(e) \in (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$. Thus, we have $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models P(e) \wedge P(f) \wedge (e < f)$. Moreover, by the condition in line 145, constant e is the immediate predecessor of f in $S_{i,j}^P$, so properties (E.1) and (E.5) ensure $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models \mathsf{not}\, \exists z.(P(z) \wedge (e < z) \wedge (z < f))$. Hence, we have $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models r'$. Moreover, $P(f) \in \Delta_{i,j}^+$ and $\Delta_{i,j}^+ \cap I_{i,j} = \emptyset$ ensure $P(f) \notin I_{i,j}$, which in turn ensures $I_{i,j} \not\models r'$. Finally, line 150 ensures $R(e, f) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, so $R(e, f) \in M_{\mathsf{A}}[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+]$ holds, as required.
- The case when $R(e, f)$ is added to $J_{i,j}$ in line 147 is analogous to above one, so we omit the details for the sake of brevity.
- Assume $R(e, f)$ is added to $J_{i,j}$ in line 149. Then, there exists a constant g with $e < g < f$ such that $\{e, f\} \subseteq S_{i,j}^P$ and $P(g) \in \Delta_{i,j}^-$ hold. Properties (E.1) and (E.5) then ensure $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models P(e) \wedge P(f) \wedge (e < f)$. Moreover, $P(g) \in \Delta_{i,j}^-$, and $\Delta_{i,j}^- \subseteq I_{i,j}$, and $\Delta_{i,j}^+ \cap I_{i,j} = \emptyset$ ensure $P(g) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$; together with properties (E.1) and (E.5), we have $g \notin S_{i,j}^P$. But then, e and f are adjacent in $S_{i,j}^P$ by line 149, so $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models \mathsf{not}\, \exists z.(P(z) \wedge (e < z) \wedge (z < f))$ holds by properties (E.1) and (E.5). Thus, $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models r'$ holds. Furthermore, $P(g) \in \Delta_{i,j}^- \subseteq I_{i,j}$ and $e < g < f$ ensure $I_{i,j} \not\models r'$; and line 150 ensures $R(e, f) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$. Thus, $R(e, f) \in M_{\mathsf{A}}[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+]$ holds, as required.

For the $\supseteq$ direction, consider an arbitrary fact $R(e, f) \in M_{\mathsf{A}}[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+]$, and let r' be the instance of rule (50) where x is mapped to e and y is mapped to f. Thus, we have $I_{i,j} \not\models r'$, $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models r'$, and $R(e, f) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$. But then, properties (E.1) and (E.5) and $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models r'$ ensure $\{P(e), P(f)\} \subseteq (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$, and e and f are adjacent in $S_{i,j}^P$ with $e < f$. Moreover, $I_{i,j} \not\models r'$ ensures that one of the following two cases holds.

- Assume $I_{i,j} \not\models P(e) \wedge P(f) \wedge (e < f)$. Together with $\{P(e), P(f)\} \subseteq (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ this ensures $\{P(e), P(f)\} \cap \Delta_{i,j}^+ \neq \emptyset$, so we have $P(e) \in \Delta_{i,j}^+$ or $P(f) \in \Delta_{i,j}^+$. For the former case, $P(e)$ is considered in line 145. Since f is the immediate successor of e in $S_{i,j}^P$, fact $R(e, f)$ is added to J in line 147. Then, line 150 and $R(e, f) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ ensure $R(e, f) \in J_{i,j}$, as required. The case where $P(f) \in \Delta_{i,j}^+$ is analogous, so we omit the details for the sake of brevity.

- Assume $I_{i,j} \not\models \operatorname{not} \exists z.(P(z) \wedge (e < z) \wedge (z < f))$. Thus, there exists a fact $P(g) \in I_{i,j}$ such that $e < g < f$ holds. In addition, $(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+ \models r'$ ensures $P(g) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$. Hence, we have $P(g) \in \Delta_{i,j}^-$, so $P(g)$ is considered in line 148. Since e and f are adjacent in $S_{i,j}^P$, fact $R(e, f)$ is added to J in line 149. Then, line 150 and $R(e, f) \notin (I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+$ ensure $R(e, f) \in J_{i,j}$, as required.

This completes our proof for $J_{i,j} = M_{\mathsf{A}}[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+] \subseteq M[(I_{i,j} \setminus \Delta_{i,j}^-) \cup \Delta_{i,j}^+] \subseteq \Pi_\infty[E_i]^s$, where first inclusion follows from Definition 8, and the second inclusion follows from property (E.5) and the fact that all body atoms of rule (50) belong to a stratum with index less than s and that the head atom of the rule belongs to stratum s. Hence, call $C_{i,j}$ is correct. $\square$

Claim 48. *If call $C_{i,j}$ is of type A2 and call $C_{i,j-1}$ satisfies properties (N1)–(N3), then $C_{i,j}$ satisfies properties (N1) and (N3).*

Proof. The previous call $C_{i,j-1}$ involves the Add^M function, so property (E.1) holds for i and $j - 1$ by the inductive assumption. Moreover, $\Delta_{i,j}^- = \emptyset$ holds in all calls of type A2, and condition (A2.c) and the fact that P belongs to a stratum with index less s than ensure that $\Delta_{i,j}^+$ contains no P-facts. Thus, property (E.1) remains preserved for i and j; and Algorithm 13 ensures $J_{i,j} = M_{\mathsf{A}}[I_{i,j} \vdots \Delta_{i,j}^-, \Delta_{i,j}^+] = \emptyset$, so call $C_{i,j}$ is correct. $\square$

Claim 49. *If call $C_{i,j}$ is of type D1 and call $C_{i-1,h_{i-1}}$ satisfies properties (N1)–(N3), then $C_{i,j}$ satisfies properties (N2) and (N3).*

Proof. Assume that $C_{i,j}$ is of the form $J_{i,1} := \mathsf{Del}^M[I_{i,1}^o, I_{i,1}^n \vdots \Delta_{i,1}^-, \Delta_{i,1}^+ \vdots \Delta_{i,1}^m]$ with $i \geq 1$, so condition (D1.d) ensures (E.6). Moreover, call $C_{i-1,h_{i-1}}$ involves the Add^M function, so the inductive assumption ensures that property (E.1) holds for $i - 1$ and h_{i-1} and can be rewritten as (E.7).

$$I_{i,1}^o = I_{i,1}^n = \Pi_\infty[E_{i-1}] \tag{E.6}$$

$$S_{i,0}^P = \{b \mid P(b) \in I_{i,1}^o\} \tag{E.7}$$

We are now ready to prove that call $C_{i,1}$ satisfies properties (N2) and (N3).

We first consider property (E.2). In particular, line 157 ensures $S_{i,1}^P = S_{i,0}^P \setminus \{b \mid P(b) \in \Delta_{i,1}^-\}$, and condition (D1.e) ensures $\Delta_{i,1}^- \cap O^{<s} = \Pi_\infty[E_{i-1}]^{<s} \setminus \Pi_\infty[E_i]^{<s}$; thus, since predicate P belongs to a stratum with index less than s, property (E.2) holds.

We next prove $J_{i,1} = M_{\mathsf{D}}[I_{i,1}^o, I_{i,1}^n \vdots \Delta_{i,1}^-, \Delta_{i,1}^+]$, which ensures that call $C_{i,1}$ is correct. For the $\subseteq$ direction, consider an arbitrary fact $R(e, f) \in J_{i,1}$, and let r' be the instance of rule (50) where x is mapped to e and y is mapped to f.

- Assume $R(e, f)$ is added to $J_{i,1}$ in line 153. Then, lines 152 and 153 ensure $e \in S_{i,0}^P$ and $P(f) \in \Delta_{i,1}^-$. Properties (E.6) and (E.7) ensure $P(e) \in I_{i,1}^o = I_{i,1}^n$, and $\Delta_{i,1}^- \subseteq I_{i,1}^n$ implies $P(f) \in I_{i,1}^o$; thus, $I_{i,1}^o \models P(e) \wedge P(f) \wedge (e < f)$ holds. Furthermore, line 153 ensures that there exists no $P(g) \in I_{i,1}^o$ with $e < g < f$, or e would not be the immediate predecessor of f in $S_{i,0}^P$; hence, $I_{i,1}^o \models \operatorname{not} \exists z.(P(z) \wedge (e < z) \wedge (z < f))$ holds as well. Thus, $I_{i,1}^o \models r'$ and $I_{i,1}^n \models r'$ hold. Moreover, $P(f) \in \Delta_{i,1}^-$ implies $P(f) \notin \Delta_{i,1}^+$, so $(I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+ \not\models r'$ holds. Furthermore, line 158 ensures $R(e, f) \in (I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+$. Consequently, we have $R(e, f) \in M_{\mathsf{D}}[I_{i,1}^o, I_{i,1}^n \vdots \Delta_{i,1}^-, \Delta_{i,1}^+]$, as required.
- The case when $R(e, f)$ is added to $J_{i,1}$ in line 154 is analogous to above one, so we omit the details for the sake of brevity.
- Assume $R(e, f)$ is added to $J_{i,1}$ in line 156. Then, there exists a constant g with $e < g < f$ such that $\{e, f\} \subseteq S_{i,0}^P$, and $P(g) \in \Delta_{i,1}^+$ all hold. Properties (E.6) and (E.7) ensure $I_{i,1}^o \models P(e) \wedge P(f) \wedge (e < f)$. Also, $P(g) \in \Delta_{i,1}^+$ and $\Delta_{i,1}^+ \cap I_{i,1}^o = \emptyset$ ensure $P(g) \notin I_{i,1}^o$, which together with property (E.7) ensures $g \notin S_{i,0}^P$. But then, e and f are adjacent in $S_{i,0}^P$ by line 156, so $I_{i,1}^o \models \operatorname{not} \exists z.(P(z) \wedge (e < z) \wedge (z < f))$ holds. Thus, $I_{i,1}^o \models r'$ and $I_{i,1}^n \models r'$ hold. Furthermore, $P(g) \in \Delta_{i,1}^+$ clearly ensures $(I_{i,1}^n \setminus \Delta_{i,1}^-) \cap \Delta_{i,1}^+ \not\models r'$; and line 158 ensures $R(e, f) \in I_{i,1}^n \setminus \Delta_{i,1}^- \subseteq (I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+$. Thus, $R(e, f) \in M_{\mathsf{D}}[I_{i,1}^o, I_{i,1}^n \vdots \Delta_{i,1}^-, \Delta_{i,1}^+]$ holds, as required.

For the $\supseteq$ direction, consider an arbitrary fact $R(e, f) \in M_{\mathsf{D}}[I_{i,1}^o, I_{i,1}^n \vdots \Delta_{i,1}^-, \Delta_{i,1}^+]$, and let r' be the instance of rule (50) where x is mapped to e and y is mapped to f. Thus, we have $I_{i,1}^o \models r'$, $I_{i,1}^n \models r'$, $(I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+ \not\models r'$, and $R(e, f) \in (I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+$. Consequently, there does not exist a fact $P(g) \in I_{i,1}^o$ with $e < g < f$. Then, property (E.7) en-

sures that e and f are adjacent elements in $S_{i,0}^P$ with $e < f$. Moreover, $(I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+ \not\models r'$ ensures that one of the following two cases holds.

- Assume $(I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+ \not\models P(e) \wedge P(f) \wedge (e < f)$. Together with $I_{i,1}^n \models r'$ this ensures $P(e) \in \Delta_{i,1}^-$ or $P(f) \in \Delta_{i,1}^-$. In the former case, $P(e)$ is considered in line 152. Since f is the immediate successor of e in $S_{i,0}^P$, fact $R(e, f)$ is added to J in line 154. Then, $R(e, f) \in (I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+$ and condition (D1.f) ensure $R(e, f) \in I_{i,1}^n \setminus \Delta_{i,1}^-$, so line 158 ensures $R(e, f) \in J_{i,1}$, as required. The case where $P(f) \in \Delta_{i,1}^-$ is analogous, so we omit the details for the sake of brevity.
- Assume $(I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+ \not\models \text{not} \exists z.(P(z) \wedge (e < z) \wedge (z < f))$. Thus, there exists a fact $P(g) \in (I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+$ such that $e < g < f$ holds. In addition, $I_{i,1}^n \models r'$ ensures $P(g) \in \Delta_{i,1}^+$, so $P(g)$ is considered in line 155. Since e and f are adjacent in $S_{i,0}^P$, constant e is the immediate predecessor of g in $S_{i,0}^P$ and constant f is the immediate successor of g in $S_{i,0}^P$. Consequently, fact $R(e, f)$ is added to J in line 156. In the same way as in the previous case, we have $R(e, f) \in J_{i,1}$, as required.

This completes our proof for $J_{i,1} = M_D[I_{i,1}^o, I_{i,1}^n : \Delta_{i,1}^-, \Delta_{i,1}^+] \subseteq (I_{i,1}^n \setminus \Delta_{i,1}^-) \cup \Delta_{i,1}^+$, where the last inclusion follows from Definition 9. Hence, call $C_{i,1}$ is correct. □

Claim 50. *If call $C_{i,j}$ is of type D2 and call $C_{i,j-1}$ satisfies properties (N1)–(N3), then $C_{i,j}$ satisfies properties (N2) and (N3).*

Proof. The previous call $C_{i,j-1}$ involves the Del^M function, so property (E.2) holds for i and $j - 1$ by the inductive assumption. Moreover, $\Delta_{i,j}^+ = \emptyset$ holds in all calls of type D2, and condition (D2.d) and the fact that P belongs to a stratum with index less than s ensure that $\Delta_{i,j}^-$ contains no P-facts. Thus, property (E.2) remains preserved for i and j; and Algorithm 14 ensures $J_{i,j} = M_D[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}^-, \Delta_{i,j}^+] = \emptyset$, so call $C_{i,j}$ is correct. □

Claim 51. *If call $C_{i,j}$ is of type (R) and call $C_{i,j-1}$ satisfies properties (N1)–(N3), then $C_{i,j}$ satisfies properties (N2) and (N3).*

Proof. Assume that $C_{i,j}$ is of the form $J_{i,j} := \text{Red}^M[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}]$. The previous call $C_{i,j-1}$ involves the Del^M function, so the inductive assumption ensures that property (E.2) holds for i and $j - 1$. Since Algorithm 15 does not change S^P, property (E.2) clearly holds for i and j. In addition, condition (R.c) ensures property (E.8), and condition (R.e), ensures property (E.9).

$$I_{i,j}^o = \Pi_\infty[E_{i-1}] \tag{E.8}$$

$$I_{i,j}^n \cap O^{<s} = \Pi_\infty[E_{i-1}]^{<s} \tag{E.9}$$

We next prove $J_{i,j} = M_R[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}]$, which ensures that call $C_{i,j}$ is correct. For the $\subseteq$ direction, consider an arbitrary fact $R(a, b) \in J_{i,j}$, and let r' be the instance of rule (50) where x is mapped to a and y is mapped to b. Line 161 then ensures $R(a, b) \in \Delta_{i,j}$, and that b is the immediate successor of a in $S_{i,j}^P$; thus, $a < b$ holds; moreover property (E.2) ensures $\{P(a), P(b)\} \subseteq \Pi_\infty[E_{i-1}] \cap \Pi_\infty[E_i]$, and it also ensures that there does not exist a fact $P(c) \in \Pi_\infty[E_{i-1}] \cap \Pi_\infty[E_i]$ with $a < c < b$. Together with properties (E.8) and (E.9), we clearly have $I_{i,j}^o \models r'$ and $I_{i,j}^n \models r'$, so $R(a, b) \in M_R[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}]$, as required. For the $\supseteq$ direction, consider an arbitrary fact $R(a, b) \in M_R[I_{i,j}^o, I_{i,j}^n : \Delta_{i,j}]$, and let r' be the instance of rule (50) where x is mapped to a and y is mapped to b. Thus, we have $I_{i,j}^o \models r'$, $I_{i,j}^n \models r'$, and $R(a, b) \in \Delta_{i,j}$. Then, there does not exist a fact $P(c) \in I_{i,j}^o \cap I_{i,j}^n$ with $a < c < b$. By properties (E.2), (E.8), and (E.9), then there does not exist a constant c in $S_{i,j}^P$ such that $a < c < b$. But then, line 161 ensures $R(a, b) \in J_{i,j}$, as required. Hence, call $C_{i,j}$ is correct. □

References

[1] Serge Abiteboul, Richard Hull, Victor Vianu, Foundations of Databases, Addison-Wesley, 1995.

[2] Foto N. Afrati, Christos H. Papadimitriou, The parallel complexity of simple logic programs, J. ACM 40 (4) (1993) 891–916.

[3] Josh Alman, Virginia Vassilevska Williams, A refined laser method and faster matrix multiplication, in: Proc. of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2021), Alexandria, VA, USA, January 10–13, 2020, pp. 522–539.

[4] Güneş Aluç, Olaf Hartig, M. Tamer Özsu, Khuzaima Daudjee, Diversified stress testing of rdf data management systems, in: International Semantic Web Conference, Springer, 2014, pp. 197–212.

[5] Peter Alvaro, Tyson Condie, Neil Conway, Khaled Elmeleegy, Joseph M. Hellerstein, Russell Sears, Boom analytics: exploring data-centric, declarative programming for the cloud, in: Proc. of the 5th European Conf. on Computer systems (EuroSys 2010), Paris, France, ACM, April 13–16, 2010, pp. 223–236.

[6] Peter Alvaro, Neil Conway, Joseph M. Hellerstein, William R. Marczak, Consistency analysis in bloom: a CALM and collected approach, in: Proc. of the 5th Biennial Conference on Innovative Data Systems Research (CIDR 2011), Asilomar, CA, USA, January 9–12, 2011, pp. 249–260.

[7] Molham Aref, Balder ten Cate, Todd J. Green, Benny Kimelfeld, Dan Olteanu, Emir Pasalic, Todd L. Veldhuizen, Geoffrey Washburn, Design and implementation of the LogicBlox system, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2015), Melbourne, Vic, Australia, ACM, May 31–June 4, 2015, pp. 1371–1382.

[8] Faiz Arni, KayLiang Ong, Shalom Tsur, Haixun Wang, Carlo Zaniolo, The deductive database system LDL++, Theory Pract. Log. Program. 3 (1) (2003) 61–94, https://doi.org/10.1017/S1471068402001515.

[9] Pablo Barceló Baeza, Querying graph databases, in: Richard Hull, Wenfei Fan (Eds.), Proc. of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2013), New York, NY, USA, ACM, June 22–27, 2013, pp. 175–188.

[10] François Bancilhon, Naive evaluation of recursively defined relations, in: On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies. Book Resulting from the Islamorada Workshop, Islamorada, FL, USA, Springer, 1985, pp. 165–178.

[11] Luigi Bellomarini, Emanuel Sallinger, Georg Gottlob, The vadalog system: Datalog-based reasoning for knowledge graphs, Proc. VLDB Endow. 11 (9) (2018) 975–987.

[12] Barry Bishop, Atanas Kiryakov, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, Ruslan Velkov, OWLIM: a family of scalable semantic repositories, Semant. Web 2 (1) (2011) 33–42.

[13] Barry Bishop, Atanas Kiryakov, Damyan Ognyanov, Ivan Peikov, Zdravko Tashev, Ruslan Velkov, Factforge: a fast track to the web of data, Semant. Web 2 (2) (2011) 157–166.

[14] José A. Blakeley, Per-Åke Larson, Frank Wm. Tompa, Efficiently updating materialized views, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1986), Washington, D.C., USA, ACM Press, May 28–30, 1986, pp. 61–71.

[15] Martin Bravenboer, Yannis Smaragdakis, Strictly declarative specification of sophisticated points-to analyses, in: Proc. of the 24th Annual ACM SIG-PLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2009), Orlando, FL, USA, ACM, October 25–29, 2009, pp. 243–262.

[16] David Carral, Irina Dragoste, Larry González, Ceriel J.H. Jacobs, Markus Krötzsch, Jacopo Urbani, VLog: a rule engine for knowledge graphs, in: ISWC, in: LNCS, vol. 11779, Springer, 2019, pp. 19–35.

[17] S. Ceri, J. Widom, Deriving production rules for incremental view maintenance, in: G.M. Lohman, A. Sernadas, R. Camps (Eds.), Proc. of the 17th Int. Conf. on Very Large Data Bases (VLDB 1991), Barcelona, Spain, Morgan Kaufmann, September 3–6, 1991, pp. 577–589.

[18] Stefano Ceri, Georg Gottlob, Letizia Tanca, What you always wanted to know about Datalog (and never dared to ask), IEEE Trans. Knowl. Data Eng. 1 (1) (1989) 146–166, https://doi.org/10.1109/69.43410.

[19] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, Andrei Voronkov, Complexity and expressive power of logic programming, ACM Comput. Surv. 33 (3) (2001) 374–425, https://doi.org/10.1145/502807.502810.

[20] Datomic, Datomic, https://www.datomic.com, 2021. (Accessed 27 September 2021).

[21] Camil Demetrescu, Giuseppe F. Italiano, Fully dynamic transitive closure: breaking through the o (n/sup 2/) barrier, in: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, 2000, IEEE, 2000, pp. 381–389.

[22] John DeTreville, Binder, a logic-based security language, in: Proc. of the 2002 IEEE Symposium on Security and Privacy (S&P 2002), Berkeley, CA, USA, IEEE Computer Society, May 12–15, 2002, pp. 105–113.

[23] Hasanat M. Dewan, David Ohsie, Salvatore J. Stolfo, Ouri Wolfson, Sushil Da Silva, Incremental database rule processing in PARADISER, J. Intell. Inf. Syst. 1 (2) (1992) 177–209, https://doi.org/10.1007/BF00962282.

[24] Guozhu Dong, Jianwen Su, Rodney W. Topor, Nonrecursive incremental evaluation of Datalog queries, Ann. Math. Artif. Intell. 14 (2–4) (1995) 187–223.

[25] Guozhu Dong, Leonid Libkin, Jianwen Su, Limsoon Wong, Maintaining transitive closure of graphs in SQL, Int. J. Inf. Technol. 5 (1999) 46–78.

[26] Jason Eisner, Nathaniel Wesley Filardo, Dyna: extending Datalog for modern AI, in: Datalog Reloaded—First International Workshop, Datalog 2010, Revised Selected Papers, Oxford, UK, Springer, March 16–19, 2010, pp. 181–220.

[27] T. Eiter, G. Gottlob, H. Mannila, Disjunctive Datalog, ACM Trans. Database Syst. 22 (3) (1997) 364–418.

[28] Wolfgang Faber, Gerald Pfeifer, Nicola Leone, Semantics and complexity of recursive aggregates in answer set programming, Artif. Intell. 175 (1) (2011) 278–298, https://doi.org/10.1016/j.artint.2010.04.002.

[29] Sumit Ganguly, Abraham Silberschatz, Shalom Tsur, A framework for the parallel processing of Datalog queries, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1990), Atlantic City, NJ, USA, ACM Press, May 23–25, 1990, pp. 143–152.

[30] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: Proc. of the 5th Int. Conf. on Logic Programming (ICLP '88), Seattler, WA, USA, MIT Press, August 15–19, 1988, pp. 1070–1080.

[31] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Gener. Comput. 9 (3–4) (1991) 365–386.

[32] Georg Gottlob, Christoph Koch, Robert Baumgartner, Marcus Herzog, Sergio Flesca, The lixto data extraction project—back and forth between theory and practice, in: Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2004), Paris, France, ACM, June 14–16, 2004, pp. 1–12.

[33] GraphDB, Graphdb, graphdb.ontotext.com, 2021. (Accessed 27 September 2021).

[34] Todd J. Green, Shan Shan Huang, Boon Thau Loo, Wenchao Zhou, Datalog and recursive query processing, Found. Trends® Databases 5 (2) (2013) 105–195, https://doi.org/10.1561/1900000017.

[35] Timothy Griffin, Leonid Libkin, Incremental maintenance of views with duplicates, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1995), San Jose, CA, USA, ACM Press, May 22–25, 1995, pp. 328–339.

[36] Timothy Griffin, Leonid Libkin, Howard Trickey, An improved algorithm for the incremental recomputation of active relational expressions, IEEE Trans. Knowl. Data Eng. 9 (3) (1997) 508–511, https://doi.org/10.1109/69.599937.

[37] Yuanbo Guo, Zhengxiang Pan, Jeff Heflin, Lubm: a benchmark for owl knowledge base systems, Web Semant. Sci. Serv. Agents World Wide Web 3 (2–3) (2005) 158–182.

[38] Ashish Gupta, Dinesh Katiyar, Inderpal Singh Mumick, Counting solutions to the view maintenance problem, in: Proc. of the Workshop on Deductive Databases Held in Conjunction with the Joint Int. Conference and Symposium on Logic Programming, Washington, D.C., USA, November 14, 1992, pp. 185–194.

[39] Ashish Gupta, Inderpal Singh Mumick, Venkatramanan Siva Subrahmanian, Maintaining views incrementally, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1993), Washington, D.C., USA, ACM, May 26–28, 1993, pp. 157–166.

[40] Eric N. Hanson, A performance analysis of view materialization strategies, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1987), San Francisco, CA, USA, ACM Press, May 27–29, 1987, pp. 440–453.

[41] Ian Horrocks, Ulrike Sattler, Decidability of shiq with complex role inclusion axioms, Artif. Intell. 160 (1–2) (2004) 79–104.

[42] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al., SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, 2004.

[43] Pan Hu, Boris Motik, Ian Horrocks, Optimised maintenance of Datalog materialisations, in: AAAI, 2018, pp. 1871–1879.

[44] Pan Hu, Boris Motik, Ian Horrocks, Modular materialisation of Datalog programs, in: Proc. of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019), Honolulu, HI, USA, January 27–February 1, 2019, pp. 2859–2866.

[45] Pan Hu, Jacopo Urbani, Boris Motik, Ian Horrocks, Datalog reasoning over compressed RDF knowledge bases, in: Proc. of the 28th ACM Int. Conf. on Information and Knowledge Management (CIKM 2019), Beijing, China, ACM, November 3–7, 2019, pp. 2065–2068.

[46] Toshihide Ibaraki, Naoki Katoh, On-line computation of transitive closures of graphs, Inf. Process. Lett. 16 (2) (1983) 95–97.

[47] Trevor Jim, SD3: a trust management system with certified evaluation, in: Proc. of the 2001 IEEE Symposium on Security and Privacy (S&P 2001), Oakland, CA, USA, IEEE Computer Society, May 14–16, 2001, pp. 106–115.

[48] Yevgeny Kazakov, An extension of complex role inclusion axioms in the description logic SROIQ, in: Proc. of the 5th Int. Joint Conf. on Automated Reasoning (IJCAR 2010), Edinburgh, UK, July 16–19, 2010, pp. 472–486.

[49] David B. Kemp, Peter J. Stuckey, Semantics of logic programs with aggregates, in: Vijay A. Saraswat, Kazunori Ueda (Eds.), Proc. of the 1991 Int. Symposium on Logic Programming, San Diego, CA, USA, MIT Press, October 28–November 1, 1991, pp. 387–401.

[50] 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, 1999, IEEE, 1999, pp. 81–89.

[51] Johannes A. La Poutre, Jan van Leeuwen, Maintenance of transitive closures and transitive reductions of graphs, in: International Workshop on Graph-Theoretic Concepts in Computer Science, Springer, 1987, pp. 106–120.

[52] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al., DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia, Semant. Web 6 (2) (2015) 167–195.

[53] Leonid Libkin, Limsoon Wong, Incremental recomputation of recursive queries with nested sets and aggregate functions, in: Sophie Cluet, Richard Hull (Eds.), Proc. of the 6th Int. Workshop on Database Programming Languages (DBPL-6), Estes Park, CO, USA, in: LNCS, vol. 1369, Springer, August 18–20, 1997, pp. 222–238.

[54] Boon Thau Loo, Tyson Condie, Joseph M. Hellerstein, Petros Maniatis, Timothy Roscoe, Ion Stoica, Implementing declarative overlays, in: Proc. of the 20th ACM Symposium on Operating Systems Principles (SOSP 2005), Brighton, UK, ACM, October 23–26, 2005, pp. 75–90.

[55] Boon Thau Loo, Joseph M. Hellerstein, Ion Stoica, Raghu Ramakrishnan, Declarative routing: extensible routing with declarative queries, in: Proc. of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2005, pp. 289–300.

[56] Boon Thau Loo, Tyson Condie, Minos N. Garofalakis, David E. Gay, Joseph M. Hellerstein, Petros Maniatis, Raghu Ramakrishnan, Timothy Roscoe, Ion Stoica, Declarative networking: language, execution and optimization, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2006), Chicago, IL, USA, ACM, June 27–29, 2006, pp. 97–108.

[57] Hermann A. Maurer, Arto Salomaa, Derick Wood, Pure grammars, Inf. Control 44 (1) (1980) 47–72.

[58] Alistair Miles, Sean Bechhofer, Skos simple knowledge organization system reference, W3C recommendation, 18:W3C, 2009.

[59] Boris Motik, Peter Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, et al., OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax, W3C, 2009.

[60] Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax, 27 October 2009, W3C Recommendation.

[61] Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, Dan Olteanu, Parallel materialisation of Datalog programs in centralised, main-memory rdf systems, in: AAAI, 2014, pp. 129–137.

[62] Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, Combining rewriting and incremental materialisation maintenance for Datalog programs with equality, in: Proc. of the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25–31, 2015, pp. 3127–3133.

[63] Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, Handling owl:sameAs via rewriting, in: Proc. of the AAAI Conference on Artificial Intelligence (AAAI 2015), Austin, TX, USA, January 25–30, 2015, pp. 231–237.

[64] Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, Incremental update of Datalog materialisation: the backward/forward algorithm, in: Proc. of the AAAI Conference on Artificial Intelligence (AAAI 2015), Austin, TX, USA, January 25–30, 2015, pp. 1560–1568.

[65] Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, Maintenance of Datalog materialisations revisited, Artif. Intell. 269 (2019) 76–136.

[66] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, Jay Banerjee, RDFox: a highly-scalable RDF store, in: ISWC, 2015, pp. 3–20.

[67] Jean-Marie Nicolas, Kioumars Yazdanian, An outline of BDGEN: a deductive DBMS, in: Proc. of the 9th World Computer Congress on Information Processing (IFIP 1983), Paris, France, September 19–23, 1983, pp. 711–717.

[68] Esko Nuutila, Efficient Transitive Closure Computation in Large Digraphs, PhD thesis, Helsinki University, 1995.

[69] Sushant Patnaik, Neil Immerman, Dyn-FO: a parallel, dynamic complexity class, J. Comput. Syst. Sci. 55 (2) (1997) 199–209.

[70] Robert Piro, Yavor Nenov, Boris Motik, Ian Horrocks, Peter Hendler, Scott Kimberly, Michael Rossman, Semantic technologies for data analysis in health care, in: ISWC, 2016, pp. 400–417.

[71] Teodor C. Przymusinski, On the declarative and procedural semantics of logic programs, J. Autom. Reason. 5 (2) (1989) 167–205.

[72] Xiaolei Qian, Gio Wiederhold, Incremental recomputation of active relational expressions, IEEE Trans. Knowl. Data Eng. 3 (3) (1991) 337–341, https://doi.org/10.1109/69.91063.

[73] Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan, Rule ordering in bottom-up fixpoint evaluation of logic programs, IEEE Trans. Knowl. Data Eng. 6 (4) (1994) 501–517, https://doi.org/10.1109/69.298169.

[74] Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan, Praveen Seshadri, The CORAL deductive system, VLDB J. 3 (2) (1994) 161–210.

[75] Kenneth A. Ross, Yehoshua Sagiv, Monotonic aggregation in deductive databases, J. Comput. Syst. Sci. 54 (1) (1997) 79–97.

[76] Warren Shen, AnHai Doan, Jeffrey F. Naughton, Raghu Ramakrishnan, Declarative information extraction using Datalog with embedded extraction predicates, in: Proc. of the 33rd Int. Conf. on Very Large Data Bases (VLDB 2007), Vienna, Austria, ACM, September 23–27, 2007, pp. 1033–1044.

[77] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J. Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J. Mungall, et al., The obo foundry: coordinated evolution of ontologies to support biomedical data integration, Nat. Biotechnol. 25 (11) (2007) 1251–1255.

[78] M. Staudt, M. Jarke, Incremental maintenance of externally materialized views, in: VLDB, 1996, pp. 75–86.

[79] Robert Stevens, Nicolas Matentzoglu, Uli Sattler, Margaret Stevens, A family history knowledge base in owl 2, in: ORE, 2014, pp. 71–76.

[80] Julien Subercaze, Christophe Gravier, Jules Chevalier, Frédérique Laforest, Inferray: fast in-memory RDF inference, Proc. VLDB Endow. 9 (6) (2016) 468–479.

[81] Efthymia Tsamoura, David Carral, Enrico Malizia, Jacopo Urbani, Materializing knowledge bases via trigger graphs, Proc. VLDB Endow. 14 (6) (2021) 943–956.

[82] Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank Van Harmelen, Henri Bal, WebPIE: a web-scale parallel inference engine using MapReduce, J. Web Semant. 10 (2012) 59–75.

[83] Jacopo Urbani, Ceriel J.H. Jacobs, Markus Krötzsch, Column-oriented Datalog materialization for large knowledge graphs, in: AAAI, 2016, pp. 258–264.

[84] A. van Gelder, K. Ross, J.S. Schlipf, The well-founded semantics for general logic programs, J. ACM 38 (3) (1991) 620–650.

[85] Dimitra Vista, Optimizing Incremental View Maintenance Expressions in Relational Databases, PhD thesis, University of Toronto, Ont., Canada, 1997.

[86] Dimitra Vista, Integration of incremental view maintenance into query optimizers, in: Proc. of the 6th Int. Conf. on Extending Database Technology (EDBT 1998), Valencia, Spain, Springer, March 23–27, 1998, pp. 374–388.

[87] John Whaley, Dzintars Avots, Michael Carbin, Monica S. Lam, Using Datalog with binary decision diagrams for program analysis, in: Proc. of the 3rd Asian Symposium on Programming Languages and Systems (APLAS 2005), Tsukuba, Japan, Springer, November 2–5, 2005, pp. 97–118.

[88] Ouri Wolfson, Hasanat M. Dewan, Salvatore J. Stolfo, Yechiam Yemini, Incremental evaluation of rules and its relationship to parallelism, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1991), Denver, CO, USA, ACM Press, May 29–31, 1991, pp. 78–87.

[89] Zhe Wu, George Eadon, Souripriya Das, Eugene Inseok Chong, Vladimir Kolovski, Melliyal Annamalai, Jagannathan Srinivasan, Implementing an inference engine for RDFS/OWL constructs and user-defined rules in Oracle, in: ICDE, IEEE, 2008, pp. 1239–1248.

[90] Weining Zhang, Ke Wang, Siu-Cheung Chau, Data partition and parallel evaluation of Datalog programs, IEEE Trans. Knowl. Data Eng. 7 (1) (1995) 163–176, https://doi.org/10.1109/69.368511.