Modeling Datalog Fact Assertion and Retraction in Linear Logic *

Edmund S. L. Lam
Carnegie Mellon University
sllam@qatar.cmu.edu

Iliano Cervesato
Carnegie Mellon University
iliano@cmu.edu

Abstract

Practical algorithms have been proposed to efficiently recompute the logical consequences of a Datalog program after a new fact has been asserted or retracted. This is essential in a dynamic setting where facts are frequently added and removed. Yet while assertion is logically well understood as incremental inference, the monotonic nature of first-order logic is ill-suited to model retraction. As such, the traditional logical interpretation of Datalog offers at most an abstract specification of Datalog systems, but has tenuous relations to the algorithms that perform efficient assertions and retractions in practical implementations. This paper proposes a logical interpretation of Datalog based on linear logic. It not only captures the meaning of Datalog updates, but also provides an operational model that underlies the dynamic changes of the set of inferable facts, all within the confines of logic. We prove the correctness of this interpretation with respect to its traditional counterpart.

1. Introduction

Datalog, a logic programming language defined in the early 1980s, was originally used to describe and implement deductive databases [8]. Given a set $\mathcal{B}$ that represents the data in the database, a Datalog program $\mathcal{P}$ specified recursive views that a user can query over $\mathcal{B}$. Logically, $\mathcal{B}$ is a set of ground atomic formulas (facts) and $\mathcal{P}$ a set of implicational formulas of a restricted form (Horn clauses). The semantics of such a Datalog system, as well as some actual implementations [4], was given by the set of the atomic logical consequences of $\mathcal{P}$ with respect to $\mathcal{B}$, in symbols $\{p(\vec{t}) \mid \mathcal{P}, \mathcal{B} \vdash p(\vec{t})\}$ — a beautifully pure use of logic.

In the last ten years, Datalog has had a resurgence in a variety of domains other than databases: to provide user authentication [13], to implement network protocols [10, 14], to program cyber-physical systems [1] and to support deductive spreadsheets [5], just to cite a few. In these applications, the contents of the fact base $\mathcal{B}$ is highly dynamic, at least compared to the relatively static databases of the 1980s. This requires efficient algorithms to assert new facts and to retract stale facts. While such algorithms have been proposed [1, 7, 11, 14, 16], the connection with logic has lagged behind, especially for retraction. Indeed, although it is easy to give a logical specification of assertion and retraction, as done in Figure 1, a direct implementation of these specifications is horribly inefficient as it involves recomputing logical consequences from scratch after each assertion or retraction.

* Funded by the Qatar National Research Fund as project NPRP 09-667-1-100 (Effective Programming for Large Distributed Ensembles)

PPDP'12, September 19–21, 2012, Leuven, Belgium.

$$\textbf{Assertion:} \quad \mathcal{P}(\mathcal{B}) \stackrel{+a}{\Longrightarrow}_{\mathcal{P}} \mathcal{P}(\mathcal{B}, a)$$
$$\textbf{Retraction:} \quad \mathcal{P}(\mathcal{B}, a) \stackrel{-a}{\Longrightarrow}_{\mathcal{P}} \mathcal{P}(\mathcal{B})$$

where $\mathcal{P}(\mathcal{B}) = \{p(\vec{t}) \mid \mathcal{P}, \mathcal{B} \vdash p(\vec{t})\}$.

Figure 1. Logical Specification of Datalog Updates

In this paper, we propose a logical interpretation of Datalog that supports updates internally. Like with the traditional reading, the semantics of a Datalog system $(\mathcal{P}, B)$ will be the set $\{p(\vec{t}) \mid \mathcal{P}, B \vdash p(\vec{t})\}$ of its logical consequences (plus some bookkeeping information). However, the assertion or retraction of a fact a will trigger a small number of logical inferences to determine the new set of logical consequences rather than performing a full recalculation. In this way, the proposed encoding not only constitutes a logical specification of the static and dynamic aspects of a Datalog system, but it can also be viewed as an efficient algorithm to compute updates. Because retraction results in the removal of inferred facts, we base our encoding on a fragment of first-order linear logic rather than on the traditional predicate calculus.

The main contributions of this paper are therefore 1) the definition of an encoding of Datalog which models the assertion and retraction of facts within logic, and 2) a proof of its correctness with respect to the traditional interpretation.

The rest of this paper is organized as follows: Section 2 reviews Datalog and linear logic. Section 3 illustrates our approach with an example. Section 4 introduces an initial linear logic interpretation of Datalog that focuses just on assertion. Section 5 extends it to capture retraction as well. We discuss related work in Section 6, while Section 7 outlines directions of future work.

2. Preliminaries

This section provides a brief review of Datalog and linear logic. We first recall some common mathematical notions that we will use throughout this paper.

2.1 Notations

In this paper, we will refer to both sets and multisets. We write $\{\!|a_1, ..., a_n|\!\}$ for a multiset with elements $a_1, ..., a_n$ (possibly repeated) and $S \uplus S'$ for the multiset union of S and S'. We use $\emptyset$ to denote both the empty set and the empty multiset. Given a multiset S, we write $set(S)$ for the set of the elements of S (without duplicates). We write $\bar{a}$ for a generic set or multiset, disambiguating as needed. Sequences will be written as $(a_1, ..., a_n)$ or $\vec{a}$, with

Terms:	t	::=	$x \mid c$
Facts:	a, b, h	::=	$p(\vec{t})$
Clause:	D	::=	$r : \forall \vec{x}.\ B \supset h$
Clause bodies:	B	::=	$b \wedge B \mid b$
Programs:	$\mathcal{P}$	::=	$D,\ \mathcal{P} \mid \cdot$
Fact bases:	$\mathcal{B}$	::=	$p(\vec{c}),\ \mathcal{B} \mid \cdot$

Figure 2. Syntax of Datalog

ϵ for the empty sequence and S, S' for the concatenation of two sequences S and S'. A substitution $[t_1/x_1, .., t_n/x_n]$, denoted θ, is a mapping of variables x_i to terms t_i. Given a collection S (set, multiset or sequence) of formulas and $\theta = [t_1/x_1, ..., t_n/x_n]$, we denote with $\theta(S)$ the simultaneous substitution of all free occurrences of x_i in S with t_i. We denote the set of free variables in S as $FV(S)$.

2.2 Datalog

Datalog is a declarative rule-based language, founded on traditional first-order logic. Its syntax is given in Figure 2. A *Datalog system* consists of a fact base, $\mathcal{B}$, and a program, $\mathcal{P}$. A *fact base* is a set of ground atomic formulas, called *facts*, of the form $p(\vec{c})$, where p is a predicate of fixed arity and $\vec{c}$ is a sequence of constants drawn from a fixed and finite signature Σ. A *Datalog program* is a set of *clauses* of the form $r : \forall \vec{x}.\ b_1 \wedge \ldots \wedge b_n \supset h$ where h and b_i are atomic formulas possibly mentioning variables among $\vec{x}$, and r is a unique label acting as the rule's name. We call h the *head* of r and $b_1 \wedge \ldots \wedge b_n$ its *body*. In this paper, we consider Datalog in its "purest" form: Datalog clauses are a restricted form of Horn clauses: the body cannot be empty, the arguments of each predicate must be either constants or variables (function symbols are disallowed), and each variable in h must appear in the antecedent $b_1 \wedge \ldots \wedge b_n$ (variables are *range-restricted*). We do not consider negated literals. Furthermore, to keep the presentation simple, we shall assume that no two body atoms have common instances.

Datalog classifies predicates p into *base predicates*, whose values are supplied by the user and can appear only in the body of a clause, and *inferred predicates* corresponding to views computed by the system. Clearly, all heads have inferred predicates. We classify the facts built from these predicates into *base facts* and *inferred facts*, respectively. The set of all facts that can be built from the signature Σ and the predicates mentioned in a program $\mathcal{P}$ is traditionally called the *Herbrand base* of $\mathcal{P}$. We denote it $herbrand(\mathcal{P})$. Furthermore, we write $herbrand_B(\mathcal{P})$ and $herbrand_I(\mathcal{P})$ for the portions of $herbrand(\mathcal{P})$ that contain base and inferred facts, respectively. The common assumption that both Σ and $\mathcal{P}$ are finite ensures that these sets are finite as well.

The semantics of a Datalog system $(\mathcal{P}, \mathcal{B})$ is defined as the set of all atomic logical consequences of $\mathcal{P}$ and $\mathcal{B}$, formally $\{p(\vec{t}) \mid \mathcal{P}, \mathcal{B} \vdash p(\vec{t})\}$. This set, the *inferential closure* of $\mathcal{P}$ and $\mathcal{B}$, consists of both base and inferred facts. We will denote it as $\mathcal{P}(\mathcal{B})$, thereby emphasizing that a fixed program $\mathcal{P}$ can be viewed as a function that maps a fact base $\mathcal{B}$ to the set of its logical consequences. This is particularly convenient given that this paper focuses on the effect of dynamic updates to the fact base of a Datalog system.

Given a fact base $\mathcal{B}$, the *assertion* of a fact a yields the updated fact base $\mathcal{B}' = \mathcal{B} \cup \{a\}$ (which we abbreviate "$\mathcal{B}, a$"). Similarly, the *retraction* of a from $\mathcal{B}$ produces $\mathcal{B}' = \mathcal{B} \setminus \{a\}$. For convenience, we will assume that $a \notin \mathcal{B}$ in the first case and $a \in \mathcal{B}$ in the second. The effect of either type of update is to change the set $\mathcal{P}(\mathcal{B})$ to $\mathcal{P}(\mathcal{B}')$. We denote this as $\mathcal{P}(\mathcal{B}) \stackrel{\alpha}{\Longrightarrow}_{\mathcal{P}} \mathcal{P}(\mathcal{B}')$ where the *action* α is $+a$ in the case of assertion and $-a$ for retraction. The logical meaning of both forms of update is summarized in Figure 1.

Although it is possible to compute $\mathcal{P}(\mathcal{B}')$ from scratch on the basis of $\mathcal{P}$ and $\mathcal{B}'$, it is generally much more efficient to do so starting from $\mathcal{P}(\mathcal{B})$. Although a number of algorithms have been proposed to do exactly this [1, 7, 11, 14, 16], they lack a logical justification, especially in the case of retraction. This paper provides such a justification.

2.3 Linear Logic: LV^{obs} and LV^{obs-}

Linear logic [9] is a substructural refinement of traditional logic. In traditional logic, assumptions grow monotonically during proof search. By restricting the use of the structural rules of contraction and weakening, linear logic enables the context to grow and shrink as inference rules are applied. This permits interpreting formulas as *resources* that are produced and consumed during the process of constructing a derivation. The result is a rich logic that is well-suited for modeling concurrent and dynamic systems [6].

In linear logic, the usage pattern of a formula is made explicit by replacing the connectives of traditional logic with a new set of operators. For instance, multiplicative conjunction or tensor ($A \otimes B$) denotes the simultaneous occurrence of A and B, and additive conjunction ($A \& B$) denotes the exclusive occurrence of either A or B. Hence $\otimes$ and $\&$ models two understandings of "conjunction" on resources. Traditional logic implication $A \supset B$ is replaced by $A \multimap B$ which preserves the idea of deducing B from A, but differs in that A is consumed upon the application of $A \multimap B$. An atom $p(\vec{t})$ is defined as in Figure 2. The fragment of linear logic used in this paper is shown in the top part of Figure 3. With the exception of the $\multimap_a$ operator (discussed below), it is a subset of the linear logic system, known as LV^{obs} [6], that has been used to faithfully encode concurrent languages and rewriting systems.

We use the judgment $\Gamma; \Delta \longrightarrow C$ to express that the linear logic formula C is derivable from the *unrestricted context* Γ (a set of formulas that can be used as often as needed) and the *linear context* Δ (a multiset of formulas that must be used exactly once) [6, 18]. The proof system defining this judgment is given in the bottom part of Figure 3. Rules (obs), $(\multimap_L)$ and $(\multimap_L^-)$ make use of the *tensorial closure* $\bigotimes \Delta$ of a context Δ, defined as follows:

$$\begin{array}{lcl} \bigotimes(.) & = & 1 \\ \bigotimes(A, \Delta) & = & A \otimes (\bigotimes \Delta) \end{array}$$

The language used in this paper, LV^{obs-}, adds one construct to LV^{obs} (highlighted in Figure 3): *guarded implication*, written $A \multimap_a B$, where A and B are any formulas and a is an atomic formula. Its left rule $(\multimap_L^-)$ differs from that of regular implication in that it is applicable only when the linear context *does not* contain the atomic formula a; moreover, once applied, a is inserted into the context alongside the consequent B. It can be thought of as an implication with *negation-as-absence* on an atomic formula a (i.e., $\neg a \otimes A \multimap B \otimes a$).

A formula of the form $A \multimap B \otimes A$ is called *reassertive* in that applying it consumes A from the linear context Δ but immediately restores it there. If $A \multimap B \otimes A$ is in the unrestricted context Γ, it can then be applied again and again, producing arbitrarily many copies of B. By contrast, the *guarded reassertive formula* $A \multimap_a B \otimes A$ can be used at most once and produce a single B, even if it is unrestricted. Indeed, it is enabled only if $a \notin \Delta$ but its application adds a to Δ thereby preventing additional uses. Therefore, a can act both as a guard and a witness of the application of this rule. We will make abundant use of guarded reassertive formulas and choose the witness a in such a way that distinct instances can be used at most once in a derivation.

Given unrestricted context Γ and linear context Δ, the pair (Γ, Δ) is *quiescent*, denoted $Quiescent(\Gamma, \Delta)$, if, whenever $\Gamma; \Delta \longrightarrow \bigotimes \Delta'$ is derivable, it must be the case that $\Delta = \Delta'$. In the sequel, we will be encoding Datalog clauses as guarded

$$
\begin{array}{rrcl}
\textit{Atomic formulas:} & a, b & ::= & p(\vec{t}) \\
\textit{Formulas:} & A, B, C & ::= & a \mid 1 \mid A \otimes B \mid A \& B \mid A \multimap B \mid \forall x.A \mid \boxed{A \multimap_a B} \\
\textit{Unrestricted contexts:} & \Gamma & ::= & . \mid \Gamma, A \\
\textit{Linear contexts:} & \Delta & ::= & . \mid \Delta, A
\end{array}
$$

$$
\frac{}{\Gamma; \Delta \longrightarrow \bigotimes \Delta}\,(\text{obs}) \qquad \frac{\Gamma, A; \Delta, A \longrightarrow C}{\Gamma, A; \Delta \longrightarrow C}\,(\text{clone}) \qquad \frac{\Gamma; \Delta \longrightarrow C}{\Gamma; \Delta, 1 \longrightarrow C}\,(1_L)
$$

$$
\frac{\Gamma; \Delta, A_1, A_2 \longrightarrow C}{\Gamma; \Delta, A_1 \otimes A_2 \longrightarrow C}\,(\otimes_L) \qquad \frac{\Gamma; \Delta, A_i \longrightarrow C}{\Gamma; \Delta, A_1 \& A_2 \longrightarrow C}\,(\&_{L_i}) \qquad \frac{t \in \Sigma \quad \Gamma; \Delta, [t/x]A \longrightarrow C}{\Gamma; \Delta, \forall x.A \longrightarrow C}\,(\forall_L)
$$

$$
\frac{\Gamma; \Delta, B \longrightarrow C}{\Gamma; \Delta, \Delta', (\bigotimes \Delta') \multimap B \longrightarrow C}\,(\multimap_L) \qquad \boxed{\frac{a \notin \Delta \qquad \Gamma; \Delta, B, a \longrightarrow C}{\Gamma; \Delta, \Delta', (\bigotimes \Delta') \multimap_a B \longrightarrow C}\,(\multimap_L^-)}
$$

Figure 3. LV^{obs} and LV^{obs-} Intuitionistic Linear Logic

$$
\mathcal{P} = \begin{cases} r_1 : \forall x, y.\ E(x,y) \supset P(x,y) \\ r_2 : \forall x, y, z.\ E(x,y) \wedge P(y,z) \supset P(x,z) \end{cases}
$$

Linear Logic Interpretation (Simplified):

$$
[\![\mathcal{P}]\!]' = \begin{cases} \forall x, y.\ E(x,y) \\ \quad \multimap \quad P(x,y) \otimes E(x,y) \otimes \\ \quad\quad \boxed{(\tilde{E}(x,y) \multimap \tilde{P}(x,y) \otimes \tilde{E}(x,y))} & -\mathcal{I}_1 \\ \forall x, y, z.\ E(x,y) \otimes P(y,z) \\ \quad \multimap \quad P(x,z) \otimes E(x,y) \otimes P(y,z) \otimes \\ \quad\quad \boxed{(\tilde{E}(x,y) \multimap \tilde{P}(x,z) \otimes \tilde{E}(x,y))} \,\&\, & -\mathcal{I}_2 \\ \quad\quad \boxed{(\tilde{P}(y,z) \multimap \tilde{P}(x,z) \otimes \tilde{P}(y,z))} \end{cases}
$$

$$
\mathcal{A}_{\mathcal{P}} = \begin{cases} \forall x, y.\ E(x,y) \otimes \tilde{E}(x,y) \multimap 1 & -\mathcal{A}_1 \\ \forall x, y.\ P(x,y) \otimes \tilde{P}(x,y) \multimap 1 & -\mathcal{A}_2 \end{cases}
$$

Figure 4. Example of Linear Logic Interpretation

reassertive implications in Γ and facts as atomic formulas in Δ. Because the rules in Figure 3 operate exclusively on the left-hand side of a sequent, the inferential closure of a fact base will be obtained when these contexts reach quiescence. The primary purpose of guarded implications is indeed to allow us to reach quiescence. This is crucial to our results in Section 4.

3. Example

Consider the Datalog program $\mathcal{P}$ shown in Figure 4, together with a simplified version $[\![\mathcal{P}]\!]'$ of its linear logic interpretation. This program computes the paths in a graph: the base predicate $E(x, y)$ means that there is an edge between nodes x and y, while the inferred predicate $P(x, y)$ indicates that there is a path between them. Each Datalog clause is represented by an unrestricted linear implication called an *inference rule* ($\mathcal{I}_1$ and $\mathcal{I}_2$). Intuitively, Datalog facts are not quite *linear* in that they should not be consumed when applied to inference rules, yet we do not want to model them as *unrestricted* formulas in Γ because they are subject to retraction. Hence we model Datalog facts as *linear* atomic formulas in Δ which can be consumed, and Datalog clauses as reassertive implications. In Figure 4, we model the retraction of a fact $E(x, y)$ by defining an equal and opposite fact $\tilde{E}(x, y)$ that will consume a copy of the atomic formula $E(x, y)$[1]. Each inference rule, modeling the inference of the head, embeds one or more scoped linear implications called a *retraction rule* (highlighted in boxes), that deals with the retraction of this same fact.

Embedded retraction rules prepare for the retraction of the head of the corresponding clause in the event of the retraction of one of its body elements. For instance, the retraction rule in $\mathcal{I}_1$ states that if $E(x, y)$ is retracted — represented by $\tilde{E}(x, y)$, we shall retract $P(x, y)$ — represented by $\tilde{P}(x, y)$. Similarly, inference rule $\mathcal{I}_2$ contains two embedded retraction rules, each preparing for the retraction of its consequent $\tilde{P}(x, z)$ in the event of the retraction of either of its antecedents ($\tilde{E}(x, y)$ or $\tilde{P}(y, z)$, respectively). These two retraction rules are connected by the & operator, thereby strictly enforcing the exclusive use of either of the rules and hence ensuring that we introduce at most a single $\tilde{P}(x, z)$ in the event of the retraction of either of $P(x, z)$'s antecedents. Retraction rules enter a derivation as *linear* formulas, which means that they disappear the moment they are used. The retraction of some base fact $E(a, b)$ will trigger a cascade of applications of retraction rules, effectively retracting all paths $P(c, d)$ that are causally dependent on $E(a, b)$. Notice that retraction rules are reassertive (their antecedents appear as in their consequent). This is because a single retraction atom $\tilde{E}(x, y)$ might be responsible for producing more than one retraction atoms (e.g. $\tilde{E}(1, 2)$ might need to produce $\tilde{P}(1, 2)$ and $\tilde{P}(1, 3)$) hence the actual consumption of a retraction atom $\tilde{E}(x, y)$ cannot be the responsibility of a retraction rule, but is pushed to an *absorption rule* (e.g., $\mathcal{A}_1$ and $\mathcal{A}_2$), which models the annihilation of equal and opposite atoms.

The interpretation $[\![\mathcal{P}]\!]'$ illustrated in Figure 4 is a simplified form of our approach. Indeed, some additional bookkeeping information needs to be included to guarantee quiescence and correctness. Sections 4 and 5 will describe refinements that will provide such guarantees.

4. Modeling Datalog Assertion

In this section, we define a linear logic interpretation of Datalog program which focuses only on assertions. We call this interpretation llD^{Assert}. It maps a Datalog assertions inference $\mathcal{P}(\mathcal{B}) \xrightarrow{+a}_{\mathcal{P}} \mathcal{P}(\mathcal{B}, a)$ to a derivation of an LV^{obs-} sequent $\Gamma; \Delta \longrightarrow \bigotimes \Delta'$ where Γ encodes $\mathcal{P}$, Δ encodes $\mathcal{P}(\mathcal{B})$ together with a, and Δ' encodes $\mathcal{P}(\mathcal{B}, a)$.

[1] $\tilde{E}$ and $\tilde{P}$ are ordinary predicates. The notation is meant to indicate that they are related to E and P, respectively.

$$\mathcal{P} \quad = \quad I_1 : \forall x, y.\, E(x,y) \supset P(x,y) \quad , \quad I_2 : \forall x, y, z.\, E(x,y) \wedge P(y,z) \supset P(x,z)$$

$$\begin{array}{lcl} \lceil \mathcal{P} \rceil & = & \forall x, y.\, \mathcal{I}_1^{(x,y)}, \forall x, y, z.\, \mathcal{I}_2^{(x,y,z)} \\ \mathcal{I}_1^{(x,y)} & = & E(x,y) \multimap_{I_1^\sharp(x,y)} P(x,y) \otimes E(x,y) \\ \mathcal{I}_2^{(x,y,z)} & = & E(x,y) \otimes P(y,z) \multimap_{I_2^\sharp(x,y,z)} P(x,z) \otimes E(x,y) \otimes P(y,z) \end{array}$$

Assertion of $E(1,2)$**:** $\emptyset \xrightarrow{+E(1,2)}{}^{LL}_{\lceil \mathcal{P} \rceil}\ \Delta$ **where** $\Delta = \wr P(1,2), E(1,2), I_1^\sharp(1,2) \int$

$$\mathcal{I}_1^{(1,2)}\ \dfrac{\dfrac{}{\lceil \mathcal{P} \rceil; \boxed{P(1,2), E(1,2), I_1^\sharp(1,2)} \longrightarrow \boxed{\bigotimes \Delta}}\,(obs)}{\boxed{\lceil \mathcal{P} \rceil}; \boxed{E(1,2)} \longrightarrow \bigotimes \Delta}\,(\mathcal{I}_L)$$

Assertion of $E(2,3)$**:** $\Delta \xrightarrow{+E(2,3)}{}^{LL}_{\lceil \mathcal{P} \rceil}\ \Delta'$ **where** $\Delta' = \wr P(1,2), P(2,3), P(1,3), E(1,2), E(2,3), I_1^\sharp(1,2), I_1^\sharp(2,3), I_2^\sharp(1,2,3) \int$

$$\mathcal{I}_2^{(1,2,3)}\ \dfrac{\dfrac{\dfrac{}{\lceil \mathcal{P} \rceil; \boxed{P(1,2), P(2,3), P(1,3), E(1,2), E(2,3), I_1^\sharp(1,2), I_1^\sharp(2,3), I_2^\sharp(1,2,3)} \longrightarrow \boxed{\bigotimes \Delta'}}\,(obs)}{\boxed{\lceil \mathcal{P} \rceil}; P(1,2), \boxed{P(2,3), E(1,2)}, E(2,3), I_1^\sharp(1,2), I_1^\sharp(2,3) \longrightarrow \bigotimes \Delta'}\,(\mathcal{I}_L)}{\boxed{\lceil \mathcal{P} \rceil}; P(1,2), E(1,2), \boxed{E(2,3)}, I_1^\sharp(1,2) \longrightarrow \bigotimes \Delta'}\,(\mathcal{I}_L)$$

Figure 5. LV^{obs-} derivations of an example llD^{Assert} interpretation

$$\begin{array}{rlcl} \text{Fact} & \lceil p(\vec{t}) \rceil & = & p(\vec{t}) \\ \text{Program} & \lceil D\ ,\ \mathcal{P} \rceil & = & \lceil D \rceil, \lceil \mathcal{P} \rceil \\ & \lceil . \rceil & = & . \\ \text{Conjunction} & \lceil b \wedge B \rceil & = & \lceil b \rceil \otimes \lceil B \rceil \\ \text{Clause} & \lceil r : \forall \vec{x}.\, B \supset h \rceil & = & \\ & & & \forall \vec{x}.\, \lceil B \rceil \multimap_{r^\sharp(\vec{x})} \lceil h \rceil \otimes \lceil B \rceil \\ \text{Fact base} & \lceil \mathcal{B}, a \rceil & = & \lceil \mathcal{B} \rceil, \lceil a \rceil \\ & \lceil \emptyset \rceil & = & . \end{array}$$

Figure 6. llD^{Assert} Interpretation of Datalog

4.1 Inference Rules and Interpretation

We model a Datalog clause as a guarded implication in the unrestricted context and a Datalog fact as a linear atomic formula. Given a Datalog clause $r : \forall \vec{x}.\, b_1 \wedge ... \wedge b_n \supset h$, we interpret it as a linear logic implication $\forall \vec{x}.\, b_1 \otimes ... \otimes b_n \multimap_{r^\sharp(\vec{x})} h \otimes b_1 \otimes ... \otimes b_n$ which we call an *inference rule* (denoted by $\mathcal{I}$). The witness $r^\sharp(\vec{x})$ serves the purpose of reaching quiescence — it is discussed in detail below. While inference rules are unrestricted formulas that can be applied as many times as needed (more precisely, as many times as the $(\multimap_L^-)$ rule permits), Datalog facts are linear atomic formulas. The responsibility of modeling their non-linearity is achieved by making inference rule reassertive. This choice is taken in anticipation to extending our logic interpretation with retraction in Section 5.

Figure 6 formalizes the encoding function $\lceil - \rceil$ which takes a Datalog program $\mathcal{P}$ and returns the corresponding linear logic interpretation $\lceil \mathcal{P} \rceil$. Datalog facts are mapped to atomic formulas, and multisets of facts are mapped to linear contexts of atomic formulas. A Datalog clause is mapped to a guarded reassertive implication.

$$\begin{array}{rcl} \text{Let} \quad \lceil \mathcal{P} \rceil & \equiv & \Gamma, \mathcal{I} \\ \mathcal{I} & \equiv & \forall \vec{x}.\, \bigotimes \Delta \multimap_{r^\sharp(\vec{x})} h \otimes \bigotimes \Delta \end{array}$$

$$\dfrac{r^\sharp(\vec{t}) \notin \Delta' \quad \lceil \mathcal{P} \rceil; [\vec{t}/\vec{x}]h, [\vec{t}/\vec{x}]\Delta, r^\sharp(\vec{t}), \Delta' \longrightarrow C}{\lceil \mathcal{P} \rceil; [\vec{t}/\vec{x}]\Delta, \Delta' \longrightarrow C}\,(\mathcal{I}_L)$$

which decomposes into:

$$\dfrac{\dfrac{r^\sharp(\vec{t}) \notin \Delta' \qquad \dfrac{\lceil \mathcal{P} \rceil; [\vec{t}/\vec{x}]h, [\vec{t}/\vec{x}]\Delta, r^\sharp(\vec{t}), \Delta' \longrightarrow C}{\lceil \mathcal{P} \rceil; [\vec{t}/\vec{x}](h \otimes \bigotimes \Delta), r^\sharp(\vec{t}), \Delta' \longrightarrow C}\,(\otimes_L)^*}{\lceil \mathcal{P} \rceil; [\vec{t}/\vec{x}]\mathcal{I}, [\vec{t}/\vec{x}]\Delta, \Delta' \longrightarrow C}\,(\multimap_L^-)}{\lceil \mathcal{P} \rceil; \mathcal{I}, [\vec{t}/\vec{x}]\Delta, \Delta' \longrightarrow C}\,(\forall_L)^*}{\lceil \mathcal{P} \rceil; [\vec{t}/\vec{x}]\Delta, \Delta' \longrightarrow C}\,(\text{clone})$$

Figure 7. llD^{Assert} Inference Macro Rule

Inference rules are guarded by an *atomic formula witness* $r^\sharp(\vec{x})$ where r is the unique name of the clause and $\vec{x}$ are all the variables of the clause. Note that $r^\sharp$ is not a Datalog predicate but an artifact of the encoding. In particular $r^\sharp(\vec{c}) \notin herbrand(\mathcal{P})$ for any $\vec{c}$. We write $\Delta^\sharp$ for a context fragment consisting only of rule witnesses. An llD^{Assert} state is a multiset of facts and witnesses. Specifically, given a Datalog program $\mathcal{P}$, a multiset of facts $F \subseteq herbrand(\mathcal{P})$ and a multiset of rule witnesses $\Delta^\sharp$, an llD^{Assert} *state* is defined as the set of linear formula $\lceil F \rceil, \Delta^\sharp$. Moreover, given base facts $\mathcal{B} \subseteq herbrand_B(\mathcal{P})$, the state $\lceil F \rceil, \Delta^\sharp$ is *reachable* from $\mathcal{B}$ if $\lceil \mathcal{P} \rceil; \lceil \mathcal{B} \rceil \longrightarrow \bigotimes \lceil F \rceil \otimes \bigotimes \Delta^\sharp$ is derivable. Note that $\lceil \mathcal{B} \rceil$ is always reachable from $\mathcal{B}$ — it is called an *initial state*.

To model the application of a Datalog clause D, we consider the sequence of linear logic derivation steps that corresponds to the cloning and instantiation of inference rule $\lceil D \rceil$, followed by the actual application of the inference rule instance and assembly of the tensor closure of consequents into atomic formulas. Figure 7

factors out the mapping of this sequence of linear logic derivation steps into a *macro rule* ($\mathcal{I}_L$). Lemma 1 in Section 4.3 entails that any derivation on LV^{obs-} states has an equivalent derivation comprised of only ($\mathcal{I}_L$) and (obs) derivation steps.

We informally define a transition $\Delta \overset{+a}{\Longrightarrow}{}^{LL}_{\lceil\mathcal{P}\rceil} \Delta'$ that maps a state Δ to another state Δ' such that Δ' has reached quiescence and $\mathcal{P}; \Delta, a \longrightarrow \bigotimes \Delta'$. We will extend it to retraction in Section 5 and formally define it in Section 5.3.

4.2 Quiescence and Complete Inference

Figure 5 illustrates an example LV^{obs-} derivation based on the llD^{Assert} interpretation of our example in Figure 4. It features two successive assertion steps from an empty state. For clarity, we highlight (using boxes) the formula fragments that are acted upon in each derivation step. The first derivation represents the assertion of $E(1,2)$ from the empty state, during which the inference rule instance $\mathcal{I}_1^{(1,2)}$ is applied and we produce the state Δ. Note that without the guards $I_1^\sharp(x,y)$ and $I_2^\sharp(x,y,z)$, the inference rules would allow the derivation of goal formulas with arbitrarily many copies of the head of the clauses that have been applied. Instead, from $E(1,2)$ we can only derive a single instance of $P(1,2)$ because an application of $\mathcal{I}_1^{(x,y)}$ introduces the atomic formula $I_1^\sharp(1,2)$ which, by rule $(\multimap_L^-)$, prevents further application of $\mathcal{I}_1^{(x,y)}$. In fact, derivations such as this will always reach quiescence. The second derivation represents the assertion of $E(2,3)$ from the state Δ obtained in the previous derivation. Here, we reach quiescence with Δ' after applying inference rule instances $\mathcal{I}_1^{(2,3)}$ and $\mathcal{I}_2^{(1,2,3)}$.

Given a Datalog program $\mathcal{P}$, a state Δ reachable from fact base $\mathcal{B}$ and some new fact a, to derive the inferential closure of $(\mathcal{B}, a)$, we construct the derivation of $\lceil\mathcal{P}\rceil; \Delta, a \longrightarrow \bigotimes \Delta'$ such that Δ' is quiescent with respect to $\lceil\mathcal{P}\rceil$.

4.3 Correctness

In this section, we argue for the soundness and completeness of the llD^{Assert} linear logic interpretation of Datalog program. Detailed proofs can be found in [12].

As mentioned earlier, for LV^{obs-} sequents of the form we are interested in, we can always find a normalized LV^{obs-} derivation that consist just of ($\mathcal{I}_L$) and (obs) rules. This simplifies the proof of the correctness of our interpretation.

LEMMA 1 (Normalized Derivations). *Given a Datalog program $\mathcal{P}$ and states $\lceil F_1\rceil, \Delta_1^\sharp$ and $\lceil F_2\rceil, \Delta_2^\sharp$, a sequent $\lceil\mathcal{P}\rceil; \lceil F_1\rceil, \Delta_1^\sharp \longrightarrow \bigotimes\lceil F_2\rceil \otimes \bigotimes \Delta_2^\sharp$ has a derivation consisting of only applications of ($\mathcal{I}_L$) and (obs) derivation steps.*

Lemma 2 states the redundancy of having multiple copies of facts, with respect to Datalog inference. Essentially a consequence of the invertibility of the left contraction rule, this lemma allows us to safely go back and forth between the multisets of facts of LV^{obs-} and the sets of facts in traditional logic, in our soundness proof (Theorem 3). In the reverse direction of the proof, it allows us to focus on traditional logic proofs that are sufficiently *saturated*, meaning that multiple applications of the same instance of Horn clauses do not provide more information. This is crucial for our completeness proof (Theorem 4), given that in the llD^{Assert} interpretation of Datalog clauses, inference rules are guarded implications that will reach quiescence, preventing multiple applications of same implication rule instances.

LEMMA 2 (Redundancy). *Given a Datalog program $\mathcal{P}$, states $\Delta_1 \equiv \lceil F_1\rceil, \Delta_1^\sharp$ and $\Delta_2 \equiv \lceil F_2\rceil, \Delta_2^\sharp$ and some fact $b \in herbrand(\mathcal{P})$, we have the following:*

1. *$\mathcal{P}, b, b, F_1 \vdash \bigwedge F_2$ if and only if $\mathcal{P}, b, F_1 \vdash \bigwedge F_2$*
2. *$\lceil\mathcal{P}\rceil; \lceil b\rceil, \lceil b\rceil, \Delta_1 \longrightarrow \lceil b\rceil \otimes \bigotimes \Delta_2$ if and only if $\lceil\mathcal{P}\rceil; \lceil b\rceil, \Delta_1 \longrightarrow \bigotimes \Delta_2$*

Given a Datalog program and a multiset of facts, every valid LV^{obs-} derivation built from the llD^{Assert} interpretation has a corresponding valid traditional logic derivation. Therefore, our linear logic interpretation is sound with respect to the traditional semantics of Datalog.

THEOREM 3 (Soundness). *Given a Datalog program $\mathcal{P}$, fact base $\mathcal{B}$ and reachable llD^{Assert} state $\lceil F\rceil, \Delta^\sharp$, if $\lceil\mathcal{P}\rceil; \lceil\mathcal{B}\rceil \longrightarrow \bigotimes\lceil F\rceil \otimes \bigotimes \Delta^\sharp$, then $\mathcal{P}, \mathcal{B} \vdash \bigwedge F$.*

Proof: Given Lemma 1, we can safely focus on normalized derivations. Then this proof proceeds by structural induction on normalized LV^{obs-} derivations: for any ($\mathcal{I}_L$) derivation step we can find a corresponding sequence of traditional logic derivation steps that mimics application of Horn clauses, while (obs) derivation steps corresponds to a combination of weakening and identity rules in traditional logic sequent calculus. □

Given a Datalog program and a set of facts, every valid traditional logic derivation built from the traditional Horn clause interpretation of Datalog, has a corresponding LV^{obs-} derivation built from its llD^{Assert} interpretation. Our interpretation is therefore also complete.

THEOREM 4 (Completeness). *Given a Datalog program $\mathcal{P}$, fact base $\mathcal{B}$ and $F \subseteq herbrand(\mathcal{P})$, if $\mathcal{P}, \mathcal{B} \vdash \bigwedge F$, then there is $\Delta^\sharp$ such that $\lceil\mathcal{P}\rceil; \lceil\mathcal{B}\rceil \longrightarrow \bigotimes\lceil F\rceil \otimes \bigotimes \Delta^\sharp$.*

Proof: With permutability properties of traditional logic derivation steps [15], we safely focus on normalized derivations, which comprise of a sequence of derivations modeling applications of Horn clauses, while weakening rules and identity rules are permuted upwards. The proof proceeds by structural induction on normalized traditional logic derivations. Thanks to Lemma 2, derivation steps which corresponds to application of Horn clauses can always be mapped to ($\mathcal{I}_L$) macro rules. The combination of weakening and identity rules are mapped to the (obs) rule. □

From Theorems 4 and 3, we have that, given a fact base $\mathcal{B}$, we can always derive a state $\lceil F\rceil, \lceil\Delta^\sharp\rceil$ such that the set interpretation of $\lceil F\rceil$ (denoted $set(F)$) is equal to $\mathcal{P}(\mathcal{B})$ and this state is quiescent. Furthermore, there can only be one such quiescent state.

THEOREM 5 (Correctness). *Let $\mathcal{P}$ be a Datalog program and $\mathcal{B}$ a collection of base facts.*

1. *There are $\lceil F\rceil$ and $\Delta^\sharp$ such that $\lceil\mathcal{P}\rceil; \lceil\mathcal{B}\rceil \longrightarrow \bigotimes\lceil F\rceil \otimes \bigotimes \Delta^\sharp$ and $Quiescent(\lceil\mathcal{P}\rceil, (\lceil F\rceil, \Delta^\sharp))$ and $set(F) = \mathcal{P}(\mathcal{B})$.*
2. *For any state Δ, if $\lceil\mathcal{P}\rceil; \lceil\mathcal{B}\rceil \longrightarrow \bigotimes\lceil\Delta\rceil$ and $Quiescent(\lceil\mathcal{P}\rceil, \Delta)$, then there is $\Delta^\sharp$ such that $\Delta = \lceil F\rceil, \Delta^\sharp$ and $set(F) = \mathcal{P}(\mathcal{B})$.*

Proof: (1) From Theorem 4, we can conclude that since $\mathcal{P}, \mathcal{B} \vdash \bigwedge \mathcal{P}(\mathcal{B})$, then we have $\lceil\mathcal{P}\rceil; \lceil\mathcal{B}\rceil \longrightarrow \bigotimes\lceil F\rceil \otimes \bigotimes \Delta^\sharp$ for some $\Delta^\sharp$ and $set(F) = \mathcal{P}(\mathcal{B})$. Furthermore we must have $Quiescent(\lceil\mathcal{P}\rceil, (\lceil F\rceil, \Delta^\sharp))$ since all inference rules must have applied to reach quiescence, and $\mathcal{P}(\mathcal{B})$ contains the inferential closure of $\mathcal{B}$.

(2) By contradiction, suppose that $Quiescent(\lceil\mathcal{P}\rceil, \Delta)$ and $set(F)$ contains some fact b not in $\mathcal{P}(\mathcal{B})$. Yet by Theorem 3, we have $\mathcal{P}, \mathcal{B} \vdash \bigwedge F$ with F containing facts that are not in $\mathcal{P}(\mathcal{B})$ hence contradicting that $\mathcal{P}(\mathcal{B})$ is the inferential closure of $\mathcal{B}$. □

$$\mathcal{P} \quad = \quad I : \forall x.\, A(x) \wedge B(x) \supset D(x)$$

$$\begin{array}{lcl} \lceil \mathcal{P} \rceil' & = & \forall x.\, \mathcal{I}_1^{(x)} \\ \mathcal{I}_1^{(x)} & = & A(x) \otimes B(x) \multimap_{I^\sharp(x)} D(x) \otimes A(x) \otimes B(x) \otimes \mathcal{R}_1^{(x)} \\ \mathcal{R}_1^{(x)} & = & (\tilde{A}(x) \otimes I^\sharp(x) \multimap \tilde{D}(x) \otimes \tilde{A}(x)) \,\&\, \\ & & (\tilde{B}(x) \otimes I^\sharp(x) \multimap \tilde{D}(x) \otimes \tilde{B}(x)) \end{array} \qquad \begin{array}{lcl} \mathcal{A}_{\mathcal{P}} & = & \forall x.\, \mathcal{A}_A^{(x)}, \forall x.\, \mathcal{A}_B^{(x)}, \forall x.\, \mathcal{A}_D^{(x)} \\ \mathcal{A}_A^{(x)} & = & A(x) \otimes \tilde{A}(x) \multimap 1 \\ \mathcal{A}_B^{(x)} & = & B(x) \otimes \tilde{B}(x) \multimap 1 \\ \mathcal{A}_D^{(x)} & = & D(x) \otimes \tilde{D}(x) \multimap 1 \end{array}$$

Assertion of $B(5)$

$$\wr A(5) \int \xrightarrow{+B(5)}{}^{LL}_{\lceil \mathcal{P} \rceil} \Delta' \text{ where } \Delta' = \wr A(5), B(5), D(5), \mathcal{R}_1^{(5)}, I^\sharp(5) \int$$

Retraction of $A(5)$

$$\Delta' \xrightarrow{-A(5)}{}^{LL}_{\lceil \mathcal{P} \rceil} \Delta'' \text{ where } \Delta'' = \wr B(5) \int$$

$$\mathcal{I}_1^{(5)} \dfrac{\dfrac{}{\lceil \mathcal{P} \rceil', \mathcal{A}_{\mathcal{P}}; \boxed{A(5), B(5), D(5), \mathcal{R}_1^{(5)}, I^\sharp(5)} \longrightarrow \boxed{\bigotimes \Delta'}}(obs)}{\lceil \mathcal{P} \rceil', \mathcal{A}_{\mathcal{P}}; \boxed{A(5), B(5)} \longrightarrow \bigotimes \Delta'}$$

$$\mathcal{A}_D^{(5)} \dfrac{\mathcal{A}_A^{(5)} \dfrac{\dfrac{}{\lceil \mathcal{P} \rceil', \mathcal{A}_{\mathcal{P}}; \boxed{B(5)} \longrightarrow \boxed{\bigotimes \Delta''}}(obs)}{\lceil \mathcal{P} \rceil', \boxed{\mathcal{A}_{\mathcal{P}}}; \boxed{A(5)}, B(5), \boxed{\tilde{A}(5)} \longrightarrow \bigotimes \Delta''}}{\lceil \mathcal{P} \rceil', \boxed{\mathcal{A}_{\mathcal{P}}}; A(5), B(5), \boxed{D(5)}, \tilde{A}(5), \boxed{\tilde{D}(5)} \longrightarrow \bigotimes \Delta''}$$

$$\mathcal{R}_1^{(5)} \dfrac{\lceil \mathcal{P} \rceil', \mathcal{A}_{\mathcal{P}}; A(5), B(5), D(5), \boxed{\mathcal{R}_1^{(5)}, I^\sharp(5), \tilde{A}(5)} \longrightarrow \bigotimes \Delta''}{\lceil \mathcal{P} \rceil', \mathcal{A}_{\mathcal{P}}; A(5), B(5), D(5), \boxed{\mathcal{R}_1^{(5)}, I^\sharp(5), \tilde{A}(5)} \longrightarrow \bigotimes \Delta''}$$

Figure 8. An Example of llD^{Full-} Interpretation with Assertion/Retraction Derivations

These results are incremental steps towards achieving the main goal of this paper, which is to formalize an interpretation of both assertion and retraction in Datalog, done in Section 5.

5. Modeling Datalog Retraction

In this section, we extend our linear logic interpretation of Datalog assertion to also account for the retraction of facts. We first illustrate our technique in Section 5.1 by informally introducing a simplified version called llD^{Full-}. Unfortunately, this simplified version admits LV^{obs-} sequent derivations that correspond to incomplete retraction of facts, meaning that when such a derivation reaches quiescence there is no guarantee that the state we obtain corresponds to a valid Datalog object. In Section, 5.2, we modify it into the llD^{Full} interpretation that provides guarantees of completeness when quiescence is achieved. Then in Section 5.3, we formally define the llD^{Full} interpretation that incorporates this modification and correctly models Datalog assertion and retraction.

5.1 Retraction and Absorption Rules

We model the removal of facts by means of *anti-predicates*. An anti-predicate $\tilde{p}$ is an ordinary predicate that is meant to be *equal and opposite* to p. While p applied to terms $\vec{t}$ forms a *fact*, $\tilde{p}$ applied to terms form a *retraction fact*. For instance, recalling the reachability example from Figure 4, the predicates E and P will have equal and opposite anti-predicates $\tilde{E}$ and $\tilde{P}$ respectively. The intent is that an atom $\tilde{p}(\vec{t})$ will ultimately absorb its equal and opposite atom $p(\vec{t})$. To model this property, we define *absorption rules* of the form $\forall \vec{x}.\ p(\vec{x}) \otimes \tilde{p}(\vec{x}) \multimap 1$. We illustrate this by means of an example, displayed in Figure 8, which shows a Datalog program $\mathcal{P}$ and its llD^{Full-} interpretation. We have three absorption rules $\mathcal{A}_-^{(x)}$, one for each predicate of this Datalog program. Also demonstrated here, we enrich the linear logic interpretation of Datalog programs with *retraction rules*: Each Datalog clause $r : \forall \vec{x}.\ p_1(\vec{t_1}) \wedge \ldots \wedge p_n(\vec{t_n}) \supset p_h(\vec{t_h})$ is modeled by an inference rule that contains an additional consequent, the additive conjunction of n retraction rules, each of which is uniquely associated with one of the antecedents $p_i(\vec{t_i})$. For instance, inference rule $\mathcal{I}_1^{(x)}$ has an embedded retraction rule $\mathcal{R}_1^{(x)}$ in its consequent and $\mathcal{R}_1^{(x)}$ contains two linear implications, joined by the & operator. Each of these linear implications produces a retraction of $D(x)$ (i.e. $\tilde{D}(x)$) in the event that we have a retraction of $B(x)$ or $C(x)$ respectively.

In the llD^{Full-} interpretation of Datalog programs, we can always find a valid LV^{obs-} derivation that represents each assertion or retraction of facts. Figure 8 illustrates two such derivations, starting from a state containing just the fact $A(5)$, the first derivation shows a derivation in the llD^{Full-} interpretation of this program, that models the assertion of a new fact $B(5)$. As such, we consider the derivation starting from the initial state, consisting of $A(5)$ and the newly asserted fact $B(5)$. We highlight (using boxes) the active fragments of the sequent which contributes to the application of the derivation step. Notice that when the inference rule instance $\mathcal{I}_1^{(5)}$ applies, we have not only the inferred fact $D(5)$, but we have the retraction rule instance $\mathcal{R}_1^{(5)}$ as well. This retraction rule is bookkeeping information that will facilitate the retraction of $D(5)$, in the event that either $A(5)$ or $B(5)$ is retracted. To illustrate this, the second derivation of this figure models the retraction of fact $A(5)$. We consider the state Δ' (obtained in the previous derivation) with the retraction atom $\tilde{A}(5)$ inserted. $\tilde{A}(5)$ and $I^\sharp(5)$ triggers the first additive conjunction component $\tilde{A}(5) \otimes I^\sharp(5) \multimap \tilde{D}(5) \otimes \tilde{A}(5)$ of the retraction rule instance $\mathcal{R}_1^{(5)}$ and produces $\tilde{D}(5)$ while retaining $\tilde{A}(5)$. The purpose of retaining $\tilde{A}(5)$ is that in the general case, $\tilde{A}(5)$ might be necessary for triggering the retraction of other atoms, yet we consume $I^\sharp(5)$ to allow future applications of inference rule instance $I^\sharp(5)$ should $A(5)$ be re-asserted by future updates. Joining the linear implications of retraction rule $\mathcal{R}_2^{(x)}$ with the & operator allows the creation of exactly *one* $\tilde{D}(5)$ atom from the retraction of $B(5)$ or $C(5)$. Hence we will not be left with a "dangling" copy of $\tilde{D}(5)$ should $C(5)$ be retracted as well (since $\mathcal{R}_1^{(5)}$ has completely been consumed by $\tilde{B}(5)$). We finally complete the derivation by applying the absorption of $D(5)$ and $A(5)$ by $\tilde{D}(5)$ and $\tilde{A}(5)$, via absorption rules $\mathcal{A}_D^{(5)}$ and $\mathcal{A}_A^{(5)}$ respectively. Note that Δ and Δ' are *quiescent* with respect to the llD^{Full-} interpretation of the Datalog program.

5.2 Quiescence and Complete Retraction

In this section we address a shortcoming of the llD^{Full-} interpretation: we cannot guarantee that reachable states derived by valid

$$\mathcal{P} \quad = \quad I : \forall x.\, A(x) \wedge B(x) \supset D(x) \qquad\qquad [\![\mathcal{P}]\!] \quad = \quad \forall x.\, \mathcal{I}_1^{(x)}$$

$$\mathcal{I}_1^{(x)} \quad = \quad A(x) \otimes B(x) \multimap_{I^\sharp(x)} \; D(x) \otimes A(x) \otimes B(x) \otimes \mathcal{R}_1^{(x)} \otimes \boxed{A^\sharp(x) \otimes B^\sharp(x)}$$

$$\mathcal{R}_1^{(x)} \quad = \quad (\tilde{A}(x) \otimes \boxed{A^\sharp(x) \otimes B^\sharp(x)} \multimap \tilde{D}(x) \otimes \tilde{A}(x)) \;\&\; (\tilde{B}(x) \otimes \boxed{A^\sharp(x) \otimes B^\sharp(x)} \multimap \tilde{D}(x) \otimes \tilde{B}(x))$$

$$\mathcal{A}_{\mathcal{P}} \quad = \quad \forall x.\, \mathcal{A}_A^{(x)}, \forall x.\, \mathcal{A}_B^{(x)}, \forall x.\, \mathcal{A}_D^{(x)}$$

$$\mathcal{A}_A^{(x)} \quad = \quad A(x) \otimes \tilde{A}(x) \boxed{\multimap_{A^\sharp(x)}} (A^\sharp(x) \multimap 1)$$

$$\mathcal{A}_B^{(x)} \quad = \quad B(x) \otimes \tilde{B}(x) \boxed{\multimap_{B^\sharp(x)}} (B^\sharp(x) \multimap 1)$$

$$\mathcal{A}_D^{(x)} \quad = \quad D(x) \otimes \tilde{D}(x) \boxed{\multimap_{D^\sharp(x)}} (D^\sharp(x) \multimap 1)$$

Assertion of $B(5)$**:** $\wr A(5) \int \xRightarrow{+B(5)}{}^{LL}_{[\![\mathcal{P}]\!]} \Delta'$ **where** $\Delta' = \wr A(5), B(5), D(5), \mathcal{R}_1^{(5)}, I^\sharp(5), A^\sharp(5), B^\sharp(5) \int$

$$\mathcal{I}_1^{(5)} \; \dfrac{\dfrac{}{[\![\mathcal{P}]\!], \mathcal{A}_{\mathcal{P}}; \boxed{A(5), B(5), D(5), \mathcal{R}_1^{(5)}, I^\sharp(5), A^\sharp(5), B^\sharp(5)} \longrightarrow \boxed{\bigotimes \Delta'}}\,(obs)}{\boxed{[\![\mathcal{P}]\!]}, \mathcal{A}_{\mathcal{P}}; \boxed{A(5), B(5)} \longrightarrow \bigotimes \Delta'}$$

Retraction of $A(5)$**:** $\Delta' \xRightarrow{-A(5)}{}^{LL}_{[\![\mathcal{P}]\!]} \Delta''$ **where** $\Delta'' = \wr B(5) \int$

$$\mathcal{R}_1^{(5)} \; \dfrac{\mathcal{A}_D^{(5)} \; \dfrac{\mathcal{A}_A^{(5)} \; \dfrac{\dfrac{}{[\![\mathcal{P}]\!], \mathcal{A}_{\mathcal{P}}; \boxed{B(5)} \longrightarrow \boxed{\bigotimes \Delta''}}\,(obs)}{[\![\mathcal{P}]\!], \boxed{\mathcal{A}_{\mathcal{P}}}; \boxed{A(5)}, B(5), \boxed{\tilde{A}(5)} \longrightarrow \bigotimes \Delta''}}{[\![\mathcal{P}]\!], \boxed{\mathcal{A}_{\mathcal{P}}}; A(5), B(5), \boxed{D(5)}, \tilde{A}(5), \boxed{\tilde{D}(5)} \longrightarrow \bigotimes \Delta''}}{[\![\mathcal{P}]\!], \mathcal{A}_{\mathcal{P}}; A(5), B(5), D(5), \boxed{\mathcal{R}_1^{(5)}, I^\sharp(5), \tilde{A}(5), A^\sharp(5), B^\sharp(5)} \longrightarrow \bigotimes \Delta''}$$

Figure 9. An Example of llD^{Full} Interpretation

$$\text{Fact} \quad [\![p(\vec{t})]\!] \quad = \quad p(\vec{t})$$

$$\text{Program} \begin{cases} [\![D,\ \mathcal{P}]\!] & = & [\![D]\!], [\![\mathcal{P}]\!] \\ [\![.]\!] & = & . \end{cases}$$

$$\text{Conjunction} \quad [\![b \wedge B]\!] \quad = \quad [\![b]\!] \otimes [\![B]\!]$$

$$\text{Clause} \begin{cases} [\![r : \forall \vec{x}.\, B \supset h]\!] = \\ \quad \forall \vec{x}.\, [\![B]\!] \multimap_{r^\sharp(\vec{x})} [\![h]\!] \otimes [\![B]\!] \otimes [\![B^\sharp]\!] \otimes \\ \qquad\qquad Ret(B, B, r^\sharp(\vec{x}), h) \end{cases}$$

$$\text{Fact base} \begin{cases} [\![\mathcal{B}, a]\!] & = & [\![\mathcal{B}]\!], [\![a]\!] \\ [\![\emptyset]\!] & = & . \end{cases}$$

where

$$Ret(b \wedge B', B, r^\sharp(\vec{x}), h) = (\![\tilde{b}]\!] \otimes r^\sharp(\vec{x}) \otimes [\![B^\sharp]\!] \multimap [\![\tilde{h}]\!] \otimes [\![\tilde{b}]\!]) \;\&\; Ret(B', B, r^\sharp(\vec{x}), h)$$

$$Ret(b, B, r^\sharp(\vec{x}), h) = (\![\tilde{b}]\!] \otimes r^\sharp(\vec{x}) \otimes [\![B^\sharp]\!] \multimap [\![\tilde{h}]\!] \otimes [\![\tilde{b}]\!])$$

Figure 10. llD^{Full} Interpretation of Datalog

LV^{obs-} derivations model complete retraction of Datalog facts when they reach quiescence.

Refer to Figure 8 again. In the second derivation, which corresponds to the retraction of $A(5)$ starting from state Δ', $\tilde{A}(5)$, rather than applying retraction rule $\mathcal{R}_1^{(5)}$, we could have instead immediately applied the absorption rule instance $\mathcal{A}_1^{(5)}$, yielding a state of quiescence and bypassing the retraction of $D(5)$. This state, however, would be an incorrect representation of the effect of retracting $A(5)$, hence such derivations should somehow be disallowed. In general, any pair of equal and opposite atoms $p(\vec{t})$ and $\tilde{p}(\vec{t})$ should be applied to (and consumed by) their corresponding absorption rule only when all retraction rules that can be applied to $p(\vec{t})$ have been applied. In other words, we must *inhibit* the application of absorption rules until all such retraction rules have been applied. To do so, we use the same technique with which we achieved quiescence in the presence of reassertive inference rules: guarded linear implications. Specifically we require that for each inference rule that is applied, for each antecedent $p(\vec{t})$ of this inference rule we produce a witness atom $p^\sharp(\vec{t})$; retraction rules additionally consume these witnesses $p^\sharp(\vec{t})$. We define absorption rules as guarded implications $p(\vec{x}) \otimes \tilde{p}(\vec{x}) \multimap_{p^\sharp(\vec{x})} (p^\sharp(\vec{x}) \multimap 1)$. The witnesses inhibit absorption rules until they are finally removed by retraction rules. Notice that an absorption rule produces a peculiar consequent $p^\sharp(\vec{x}) \multimap 1$. The purpose for this is to consume the atom $p^\sharp(\vec{x})$ produced as a consequent of applying the $(\multimap_L^-)$ rule.

Figure 9 illustrates this idea on our last example. The boxes highlight the components added on top of the llD^{Full-} interpretation example of Figure 8. We call it the llD^{Full} interpretation, implemented by the function $[\![-]\!]$, defined in Section 5.3. Specifically inference rule $\mathcal{I}_1^{(x)}$ has two additional consequents $A^\sharp(x)$ and $B^\sharp(x)$, which are witness atoms for antecedents $A(x)$ and $B(x)$ respectively. Absorption rules are now guarded linear implications with witness atoms as inhibitors. For instance, $\mathcal{A}_A^{(x)}$ can only apply if there are no witnesses $A^\sharp(x)$ in the context. Finally, on top of consuming inference witness $I^\sharp(x)$ and producing retraction atom $\tilde{D}(x)$, the retraction rules in $\mathcal{R}_1^{(x)}$ have the additional responsibility of consuming witness atoms $A^\sharp(x)$ and $B^\sharp(x)$.

Figure 9 shows derivations for the llD^{Full} interpretation of our last example (Figure 8). The first derivation corresponds to the assertion of $B(5)$ starting from a state containing a single $A(5)$

$$\text{Let} \quad \llbracket\mathcal{P}\rrbracket \equiv \Gamma, \mathcal{I} \qquad \mathcal{I} \equiv \forall\vec{x}.\ \bigotimes\Delta \multimap_{r^\sharp(\vec{x})} h \otimes \bigotimes\Delta \otimes \mathcal{R}_r \otimes \bigotimes\Delta^\sharp$$

$$\frac{\begin{array}{c} r^\sharp(\vec{t}) \notin \Delta' \\ \llbracket\mathcal{P}\rrbracket, \mathcal{A}_\mathcal{P}; [\vec{t}/\vec{x}](h, \Delta, \mathcal{R}_r, \Delta^\sharp), r^\sharp(\vec{t}), \Delta' \longrightarrow C \end{array}}{\llbracket\mathcal{P}\rrbracket, \mathcal{A}_\mathcal{P}; [\vec{t}/\vec{x}]\Delta, \Delta' \longrightarrow C}\ (\mathcal{I}_L)$$

which decomposes into:

$$\dfrac{r^\sharp(\vec{t}) \notin \Delta' \qquad \dfrac{\begin{array}{ll} \llbracket\mathcal{P}\rrbracket, & [\vec{t}/\vec{x}](h, \Delta, \mathcal{R}_r), \\ \mathcal{A}_\mathcal{P} \quad ; & r^\sharp(\vec{t}), [\vec{t}/\vec{x}]\Delta^\sharp, \Delta' \end{array} \longrightarrow C}{\begin{array}{ll} \llbracket\mathcal{P}\rrbracket, & [\vec{t}/\vec{x}](h \otimes \bigotimes\Delta \otimes \mathcal{R}_r \otimes \\ \mathcal{A}_\mathcal{P} \quad ; & \bigotimes\Delta^\sharp), r^\sharp(\vec{t}), \Delta' \end{array} \longrightarrow C}\ (\otimes_L)^*}{\dfrac{\llbracket\mathcal{P}\rrbracket, \mathcal{A}_\mathcal{P}; [\vec{t}/\vec{x}]\mathcal{I}, [\vec{t}/\vec{x}]\Delta, \Delta' \longrightarrow C}{\llbracket\mathcal{P}\rrbracket, \mathcal{A}_\mathcal{P}; \mathcal{I}, [\vec{t}/\vec{x}]\Delta, \Delta' \longrightarrow C}\ (\forall_L)^*}\ (\text{clone})}\ (\multimap_L^-)$$

Figure 11. llD^{Full} Inference Macro Rule

and is similar to that in Figure 8 except we additionally infer witness atoms $A^\sharp(5)$ and $B^\sharp(5)$. The next derivation illustrates the subsequent retraction of $A(5)$. Now absorption rule $\mathcal{A}_A^{(5)}$ cannot be applied to $\tilde{A}(5)$ and $A(5)$ as there is a witness atom $A^\sharp(5)$ inhibiting $\mathcal{A}_A^{(5)}$. The only possible way forward is the application of retraction rule $\mathcal{R}_1^{(5)}$, which on top of producing retraction atom $D(\tilde{5})$ and consuming inference witness $I^\sharp(5)$, consumes witness atoms $A^\sharp(5)$ and $B^\sharp(5)$, thereby unlocking the applicability of $\mathcal{A}_A^{(5)}$.

5.3 The llD^{Full} Interpretation

We now formalize the llD^{Full} interpretation of Datalog programs. For each atom representing a fact $p(\vec{t})$, we define its equal and opposite retraction atom $\tilde{p}(\vec{t})$. We will extend this notation to collections of atoms, hence for a multiset of atoms Δ, we denote a multiset of all equal and opposite retraction atoms of Δ by $\tilde{\Delta}$. Beside the rule witnesses $r^\sharp(\vec{x})$ of Section 4.1, for each atom $p(\vec{x})$ we have another type of witnesses, the *fact witness* $p^\sharp(\vec{x})$. While a rule witness $r^\sharp(\vec{t})$ witnesses the application of an inference rule instance of r, a fact $p^\sharp(\vec{t})$ witnesses a use of $p(\vec{t})$ to infer some other fact. Given a Datalog program $\mathcal{P}$, for each predicate p of some arity n, we define the set of all absorption rules $\mathcal{A}_\mathcal{P}$ as follows:

$$\mathcal{A}_\mathcal{P} = \{\!|\forall\vec{x}.p(\vec{x}) \otimes \tilde{p}(\vec{x}) \multimap_{p^\sharp(\vec{x})} (p^\sharp(\vec{x}) \multimap 1) \mid p \text{ in } \mathcal{P}|\!\}$$

Figure 10 formalizes the llD^{Full} interpretation of Datalog programs as the function $\llbracket - \rrbracket$. The main difference between this function and the function $\lceil - \rceil$ of llD^{Full-} is that Datalog clauses are inference rules that include two additional types of consequents. Specifically, a retraction rule which is an additive conjunction of linear implications defined by the function Ret and a tensor closure of inference fact witnesses derived from the antecedents of the inference rule. Altogether, given a Datalog clause $r : \forall\vec{x}.\ B \supset h$, its corresponding inference rule in llD^{Full} interpretation infers three types of bookkeeping information, namely, the retraction rule $\mathcal{R}_r$, the rule witness $r^\sharp(\vec{x})$ and fact witnesses $B^\sharp$.

Given Datalog program $\mathcal{P}$, multiset of facts $F \subseteq herbrand(\mathcal{P})$, multiset of retraction rules $\Delta^\mathcal{R}$ and multiset of witness atoms $\Delta^\sharp$, a llD^{Full} state of this Datalog program is an LV^{obs-} linear context

$$\frac{a \notin \Delta \quad \llbracket\mathcal{P}\rrbracket, \mathcal{A}_\mathcal{P}; \Delta, a \longrightarrow \bigotimes\Delta' \qquad Quiescent((\llbracket\mathcal{P}\rrbracket, \mathcal{A}_\mathcal{P}), \Delta')}{\Delta \overset{+a}{\Longrightarrow}{}^{LL}_{\llbracket\mathcal{P}\rrbracket} \Delta'}\ (Infer)$$

$$\frac{a \in \Delta \quad \llbracket\mathcal{P}\rrbracket, \mathcal{A}_\mathcal{P}; \Delta, \tilde{a} \longrightarrow \bigotimes\Delta' \qquad Quiescent((\llbracket\mathcal{P}\rrbracket, \mathcal{A}_\mathcal{P}), \Delta')}{\Delta \overset{-a}{\Longrightarrow}{}^{LL}_{\llbracket\mathcal{P}\rrbracket} \Delta'}\ (Retract)$$

Figure 12. Assertion and Retraction, in llD^{Full}

$\llbracket F\rrbracket, \Delta^\mathcal{R}, \Delta^\sharp$. Given base facts $\mathcal{B} \subseteq herbrand_B(\mathcal{P})$, the state $\lceil F\rceil, \Delta^\mathcal{R}, \Delta^\sharp$ is *reachable* from $\mathcal{B}$ if $\llbracket\mathcal{P}\rrbracket; \lceil\mathcal{B}\rceil \longrightarrow \bigotimes\llbracket F\rrbracket \otimes \bigotimes\Delta^\mathcal{R} \otimes \bigotimes\Delta^\sharp$ is derivable. Note that $\llbracket\mathcal{B}\rrbracket$ is always reachable from $\mathcal{B}$ — it is again called an *initial state*.

Similarly to Figure 7, Figure 11 defines the inference macro derivation rule for the llD^{Full} interpretation.

Figure 13 introduces the retraction macro step, which models the retraction of some fact h. As defined in Figure 10 by the function Ret, a retraction rule $\mathcal{R}_r$ is an additive conjunction of linear implications of the form $\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes\Delta^\sharp \multimap \tilde{a}_i \otimes \tilde{h}$, which are reassertive (antecedent $\tilde{a}_i$ appears as consequent) and the only component that varies among the linear implications is the antecedent $\tilde{a}_i$. As such, retraction rule $\mathcal{R}_r$ can be viewed as a formula that reacts to any $\tilde{a}_i$ which is an antecedent of one of its linear implications and, when applied, consumes rule witness $r(\vec{t})$ and fact witnesses $\Delta^\sharp$, and produces $\tilde{h}$.

Figure 14 defines the last of the three macro derivation steps of the llD^{Full} interpretation. It models the absorption of an atom $p(\vec{t})$ and the equal and opposite $\tilde{p}(\vec{t})$, but only in the absence of any witness $p^\sharp(\vec{t})$.

Figure 12 illustrates the top-level formulation of the main goal of this paper, defining $\Longrightarrow^{LL}_{\llbracket\mathcal{P}\rrbracket}$, our linear logic specification of Datalog assertion and retraction in terms of LV^{obs-} derivations. In contrast to the logical specification $\Longrightarrow_\mathcal{P}$ in Figure 1 which defines assertion and retraction based on inferential closures of base facts, $\Longrightarrow^{LL}_{\llbracket\mathcal{P}\rrbracket}$ is defined as follows: the assertion of a new base fact a to a state Δ maps to the derivation with fact a inserted into the original state, while the retraction an existing base fact a from state Δ maps to a derivation with $\tilde{a}$ inserted. In both cases, we consider only resulting states which are quiescent. Theorem 12 in Section 5.5 states the correspondence of the llD^{Full} interpretation that formally prove the correctness this logical specification.

5.4 Cycles and Re-assertion

In this section, we analyze two aspects of the llD^{Full} interpretation. Specifically, we discuss *cycles* in Datalog inference and the need for *re-assertion* when doing retraction.

Quiescence in Cycles: One important issue to consider for the llD^{Full} interpretation, is whether quiescence can be achieved for Datalog programs with cycles. Figure 15 shows one such example modified from the graph example of Figure 5, along with its linear logic interpretation (we omit the absorption rules for brevity). This example attempts to generate reflexive paths $P(x, y)$ from base facts $E(x, y)$ that represent edges. Figure 15 also illustrates an LV^{obs-} derivation which represents the assertion of $E(4, 5)$ which in fact, achieves quiescence: We infer the facts $P(4, 5)$ and $P(5, 4)$ through inference rule instances $\mathcal{I}_1^{(4,5)}$ and $\mathcal{I}_2^{(4,5)}$, as well as an additional copy of $P(4, 5)$ through $\mathcal{I}_2^{(5,4)}$, before we reach quiescence. We omit fact witnesses and retraction rules as they are inessential in this discussion. In general, as long as Datalog clauses

$$\mathcal{P} \quad = \quad I_1 : \forall x, y.\ E(x,y) \supset P(x,y) \quad , \quad I_2 : \forall x, y.\ P(x,y) \supset P(y,x)$$

$$\begin{array}{lcl}
[\![\mathcal{P}]\!] & = & \forall x, y.\ \mathcal{I}_1^{(x,y)},\ \forall x, y.\ \mathcal{I}_2^{(x,y)} \\
\mathcal{I}_1^{(x,y)} & = & E(x,y) \multimap_{I_1^\sharp(x,y)} P(x,y) \otimes E(x,y) \otimes E^\sharp(x,y) \otimes \mathcal{R}_1^{(x,y)} \\
\mathcal{R}_1^{(x,y)} & = & \tilde{E}(x,y) \otimes I_1^\sharp(x,y) \otimes E^\sharp(x,y) \multimap \tilde{P}(x,y) \otimes \tilde{E}(x,y) \\
\mathcal{I}_2^{(x,y)} & = & P(x,y) \multimap_{I_2^\sharp(x,y)} P(y,x) \otimes P(x,y) \otimes P^\sharp(x,y) \otimes \mathcal{R}_2^{(x,y)} \\
\mathcal{R}_2^{(x,y)} & = & \tilde{P}(x,y) \otimes I_2^\sharp(x,y) \otimes P^\sharp(x,y) \multimap \tilde{P}(y,x) \otimes \tilde{P}(x,y)
\end{array}$$

Assertion of $E(4,5)$**:** $\emptyset \xrightarrow{+E(4,5)}_{\mathcal{P}} \Delta$ **where** $\Delta = \wr P(4,5), P(5,4), P(4,5), E(4,5), I_1^\sharp(4,5), I_2^\sharp(4,5), I_2^\sharp(5,4), ..\int$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \boxed{P(4,5), P(5,4), P(4,5), E(4,5), I_1^\sharp(4,5), I_2^\sharp(4,5), I_2^\sharp(5,4), ..} \longrightarrow \boxed{\bigotimes \Delta}}\,(obs)}{\mathcal{I}_2^{(5,4)}\ [\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \boxed{P(5,4)}, P(4,5), E(4,5), I_1^\sharp(4,5), I_2^\sharp(4,5), .. \longrightarrow \bigotimes \Delta}\,(\mathcal{I}_L)}{\mathcal{I}_2^{(4,5)}\ \boxed{[\![\mathcal{P}]\!]}, \mathcal{A}_\mathcal{P}; \boxed{P(4,5)}, E(4,5), I_1^\sharp(4,5), .. \longrightarrow \bigotimes \Delta}\,(\mathcal{I}_L)}{\mathcal{I}_1^{(4,5)}\ \boxed{[\![\mathcal{P}]\!]}, \mathcal{A}_\mathcal{P}; \boxed{E(4,5)} \longrightarrow \bigotimes \Delta}\,(\mathcal{I}_L)$$

Figure 15. Example of Quiescence in the Presence of Cycles

Let $\quad \mathcal{R}_i \equiv \tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes \Delta^\sharp \multimap \tilde{a}_i \otimes \tilde{h}$

$$\dfrac{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \tilde{a}_k, \tilde{h}, \Delta \longrightarrow C}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \tilde{a}_k, \&_{i=1}^n \mathcal{R}_i, r(\vec{t}), \Delta^\sharp, \Delta \longrightarrow C}\,(\mathcal{R}_L)$$

for some $k \geq 1$ and $k \leq n$, which decomposes into:

$$\dfrac{\dfrac{\dfrac{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \tilde{a}_k, \tilde{h}, \Delta \longrightarrow C}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \tilde{a}_k \otimes \tilde{h}, \Delta \longrightarrow C}\,(\otimes_L)}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \tilde{a}_k, \mathcal{R}_k, r(\vec{t}), \Delta^\sharp, \Delta \longrightarrow C}\,(\multimap_L)}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \tilde{a}_k, \&_{i=1}^n \mathcal{R}_i, r(\vec{t}), \Delta^\sharp, \Delta \longrightarrow C}\,(\&_L)^*$$

Figure 13. llD^{Full} Retraction Macro Rule

Let $\quad \forall \vec{x}.\ p(\vec{x}) \otimes \tilde{p}(\vec{x}) \multimap_{p^\sharp(\vec{x})} (p^\sharp(\vec{x}) \multimap 1) \in \mathcal{A}_\mathcal{P}$

$$\dfrac{p^\sharp(\vec{t}) \notin \Delta \quad [\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \Delta \longrightarrow C}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; p(\vec{t}), \tilde{p}(\vec{t}), \Delta \longrightarrow C}\,(\mathcal{A}_L)$$

which decomposes into:

$$\dfrac{\dfrac{p^\sharp(\vec{t}) \notin \Delta \quad \dfrac{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \Delta \longrightarrow C}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; 1, \Delta \longrightarrow C}\,(1_L)}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; (p^\sharp(\vec{t}) \multimap 1), p^\sharp(\vec{t}), \Delta \longrightarrow C}\,(\multimap_L)}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P};\ \begin{array}{l}(p(\vec{t}) \otimes \tilde{p}(\vec{t}) \multimap_{p^\sharp(\vec{t})} (p^\sharp(\vec{t}) \multimap 1)) \\ , p(\vec{t}), \tilde{p}(\vec{t}), \Delta\end{array} \longrightarrow C}\,(\multimap_L^-)$$

$$\dfrac{\dfrac{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \mathcal{A}_p, p(\vec{t}), \tilde{p}(\vec{t}), \Delta \longrightarrow C}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; p(\vec{t}), \tilde{p}(\vec{t}), \Delta \longrightarrow C}\,(clone)}{}\,(\forall_L)^*$$

Figure 14. llD^{Full} Absorption Macro Rule

are well-formed and the set Σ of all possible constants is finite, the llD^{Full} interpretation of Datalog programs will reach quiescence under LV^{obs-} derivations. This is because the possible combinations of rule instances are finite, and since for each rule instance we can apply it at most once in a derivation, all LV^{obs-} derivations of well-formed Datalog programs reach quiescence.

Re-assertion during Retraction: In some cases, re-assertion of facts occurs during retraction while working towards quiescence. Figure 16 illustrates a Datalog program which exhibits such a behavior. From either base facts $A(x)$ or $B(x)$, we can infer $C(x)$. We consider an initial state Δ where $A(2)$ and $B(2)$ has been asserted. Hence, inference rule instances $\mathcal{I}_1^{(2)}, \mathcal{I}_2^{(2)}$ and $\mathcal{I}_3^{(2)}$ has been applied before reaching quiescence at state Δ. Note that while we have two copies of $C(2)$ (from $\mathcal{I}_1^{(2)}$ and $\mathcal{I}_2^{(2)}$), we only have one copy of $D(2)$, since rule instance $\mathcal{I}_3^{(2)}$ is only permitted to be applied once. Next we consider the situation when $A(2)$ is retracted, triggering the application of retraction rules $\mathcal{R}_1^{(2)}$ and $\mathcal{R}_3^{(2)}$ which ultimately absorbs $A(2)$, $C(2)$ and $D(2)$. This leads us to a seemingly inconsistent state, where $B(2)$ and $C(2)$ exists, yet $D(2)$ has been completely retracted as part of the retraction closure of $A(2)$. However, quiescence is only achieved when inference rule instance $\mathcal{I}_3^{(2)}$ is re-applied, re-asserting another copy of $D(2)$. While re-assertion guarantees the re-completion of assertion closures of the llD^{Full} interpretation, it arguably adds overheads. One of our future objectives is to refine the llD^{Full} interpretation to overcome the need for re-assertion during retraction.

5.5 Correctness

In this section, we highlight the proofs of correspondence of the llD^{Full} linear logic interpretation of Datalog program. Details of the proofs can be found in the full version of this paper [12].

For any LV^{obs-} sequents constructed from the llD^{Full} interpretations of Datalog assertion, we can always find a normalized LV^{obs-} derivation that consists only of $(\mathcal{I}_L)$ and (obs) derivation rules. Similarly, for sequents from llD^{Full} interpretations of Datalog retraction, we can find normalized derivations that consist strictly only of $(\mathcal{I}_L)$, $(\mathcal{R}_L)$, $(\mathcal{A}_L)$ and (obs) derivation rules. This will simplify the proof of the correctness of our interpretation.

LEMMA 6 (Normalized Derivations). *Given Datalog program $\mathcal{P}$, llD^{Full} states $[\![F_1]\!], \Delta_1^\mathcal{R}, \Delta_1^\sharp$ and $[\![F_2]\!], \Delta_2^\mathcal{R}, \Delta_2^\sharp$, a LV^{obs-}*

$$\begin{array}{lcl} \mathcal{P} & = & I_1 : \forall x.\, A(x) \supset C(x) \quad I_2 : \forall x.\, B(x) \supset C(x) \quad I_3 : \forall x.\, C(x) \supset D(x) \\ [\![\mathcal{P}]\!] & = & \forall x.\, \mathcal{I}_1^{(x)},\ \forall x.\, \mathcal{I}_2^{(x)},\ \forall x.\, \mathcal{I}_3^{(x)} \end{array}$$

$$\begin{array}{lcllcl} \mathcal{I}_1^{(x)} & = & A(x) \multimap_{I_1^\sharp(x)} C(x) \otimes A(x) \otimes A^\sharp(x) \otimes \mathcal{R}_1^{(x)} & \mathcal{R}_1^{(x)} & = & \tilde{A}(x) \otimes I_1^\sharp(x) \otimes A^\sharp(x) \multimap \tilde{C}(x) \otimes \tilde{A}(x) \\ \mathcal{I}_2^{(x)} & = & B(x) \multimap_{I_2^\sharp(x)} C(x) \otimes B(x) \otimes B^\sharp(x) \otimes \mathcal{R}_2^{(x)} & \mathcal{R}_2^{(x)} & = & \tilde{B}(x) \otimes I_2^\sharp(x) \otimes B^\sharp(x) \multimap \tilde{C}(x) \otimes \tilde{B}(x) \\ \mathcal{I}_3^{(x)} & = & C(x) \multimap_{I_3^\sharp(x)} D(x) \otimes C(x) \otimes C^\sharp(x) \otimes \mathcal{R}_3^{(x)} & \mathcal{R}_3^{(x)} & = & \tilde{C}(x) \otimes I_3^\sharp(x) \otimes C^\sharp(x) \multimap \tilde{D}(x) \otimes \tilde{C}(x) \end{array}$$

Retraction of $A(2)$**:** $\Delta \xrightarrow{-A(2)}{}^{LL}_{[\![\mathcal{P}]\!]} \Delta'$ **where** $\Delta' = \wr D(2), B(2), C(2), I_3^\sharp(2), I_2^\sharp(2), C^\sharp(2), B^\sharp(2), \mathcal{R}_3^{(2)}, \mathcal{R}_2^{(2)} \int$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \boxed{D(2), B(2), C(2), I_3^\sharp(2), I_2^\sharp(2), C^\sharp(2), B^\sharp(2), \mathcal{R}_3^{(2)}, \mathcal{R}_2^{(2)}} \longrightarrow \boxed{\bigotimes \Delta'}}\ (\mathcal{I}_L)}{\boxed{[\![\mathcal{P}]\!]}, \mathcal{A}_\mathcal{P}; B(2), \boxed{C(2)}, I_2^\sharp(2), B^\sharp(2), \mathcal{R}_2^{(2)} \longrightarrow \bigotimes \Delta'}\ (\mathcal{A}_L)}{[\![\mathcal{P}]\!], \boxed{\mathcal{A}_\mathcal{P}}; \boxed{\tilde{C}(2)}, B(2), \boxed{C(2)}, C(2), I_2^\sharp(2), B^\sharp(2), \mathcal{R}_2^{(2)} \longrightarrow \bigotimes \Delta'}\ (\mathcal{A}_L)}{[\![\mathcal{P}]\!], \boxed{\mathcal{A}_\mathcal{P}}; \boxed{\tilde{D}(2)}, \tilde{C}(2), B(2), C(2), C(2), \boxed{D(2)}, I_2^\sharp(2), B^\sharp(2), \mathcal{R}_2^{(2)} \longrightarrow \bigotimes \Delta'}\ (\mathcal{R}_L)}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \boxed{\tilde{C}(2)}, B(2), C(2), C(2), D(2), I_2^\sharp(2), \boxed{I_3^\sharp(2)}, B^\sharp(2), \boxed{C^\sharp(2)}, \mathcal{R}_2^{(2)}, \boxed{\mathcal{R}_3^{(2)}} \longrightarrow \bigotimes \Delta'}\ (\mathcal{A}_L)}{[\![\mathcal{P}]\!], \boxed{\mathcal{A}_\mathcal{P}}; \tilde{C}(2), \boxed{\tilde{A}(2), A(2)}, B(2), C(2), C(2), D(2), I_2^\sharp(2), I_3^\sharp(2), B^\sharp(2), C^\sharp(2), \mathcal{R}_2^{(2)}, \mathcal{R}_3^{(2)} \longrightarrow \bigotimes \Delta'}\ (\mathcal{R}_L)}{[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \begin{array}{l} \boxed{\tilde{A}(2)}, A(2), B(2), C(2), C(2), D(2), \boxed{I_1^\sharp(2)}, I_2^\sharp(2), I_3^\sharp(2), \\ \boxed{A^\sharp(2)}, B^\sharp(2), C^\sharp(2), \boxed{\mathcal{R}_1^{(2)}}, \mathcal{R}_2^{(2)}, \mathcal{R}_3^{(2)} \end{array} \longrightarrow \bigotimes \Delta'}$$

Figure 16. Example of re-assertion during retraction

derivation $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F_1]\!], \Delta_1^\mathcal{R}, \Delta_1^\sharp \longrightarrow \bigotimes [\![F_1]\!] \otimes \bigotimes \Delta_2^\mathcal{R} \otimes \bigotimes \Delta_2^\sharp$ *has a proof consisting of only applications of* $(\mathcal{I}_L)$*,* $(\mathcal{R}_L)$*,* $(\mathcal{A}_L)$ *and (obs) derivation rules.*

For every llD^{Full} derivation that models the assertion of facts (excluding retraction), we have a corresponding llD^{Assert} derivation and vice versa. Both directions depend on the fact that witness facts and retraction rules have no effects on the assertion of new facts.

LEMMA 7 (Soundness and Completeness of Assertion). *Given a Datalog Program* $\mathcal{P}$*, for any fact base* $\mathcal{B}$*, multiset of facts* $F \subseteq herbrand(\mathcal{P})$*, multiset of rule witnesses* $\Delta_R^\sharp$*, multiset of fact witnesses* $\Delta_F^\sharp$ *and multiset of retraction rules* $\Delta^\mathcal{R}$*,*

$$[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![\mathcal{B}]\!] \longrightarrow \bigotimes [\![F]\!] \otimes \bigotimes \Delta^\mathcal{R} \otimes \bigotimes \Delta_R^\sharp \otimes \bigotimes \Delta_F^\sharp$$
if and only if $$\lceil \mathcal{P} \rceil; \lceil \mathcal{B} \rceil \longrightarrow \bigotimes \lceil F \rceil \otimes \bigotimes \Delta_R^\sharp$$

Similarly to Theorem 5, for any fact base $\mathcal{B}$, we can derive a quiescent state which is an inferential closure of $\mathcal{B}$. Further more this there can only be one such quiescent state.

THEOREM 8 (Correctness of Assertion). *Let* $\mathcal{P}$ *be a Datalog program and* $\mathcal{B}$ *a collection of base facts,*

1. *There is* $[\![F]\!], \Delta^\mathcal{R}, \Delta^\sharp$ *such that* $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![\mathcal{B}]\!] \longrightarrow \bigotimes [\![F]\!] \otimes \bigotimes \Delta^\mathcal{R} \otimes \bigotimes \Delta^\sharp$ *and* $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), ([\![F]\!], \Delta^\mathcal{R}, \Delta^\sharp))$ *and* $set(F) = \mathcal{P}(\mathcal{B})$.
2. *For any state* Δ*, if we have* $[\![\mathcal{P}]\!]; [\![\mathcal{B}]\!] \longrightarrow \bigotimes [\![\Delta]\!]$ *and* $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), \Delta)$*, then there is* $\Delta^\mathcal{R}, \Delta^\sharp$ *such that* $\Delta = \lceil F \rceil, \Delta^\mathcal{R}, \Delta^\sharp$ *and* $set(F) = \mathcal{P}(\mathcal{B})$.

Proof: This follows from Lemma 7, which states the correspondence between llD^{Full} and llD^{Assert} assertions. As such, since Theorem 5 holds, then we must have Theorem 8 as well. □

While $(\mathcal{I}_L)$ macro derivation rules in llD^{Full} corresponds to llD^{Assert}, applications of the $(\mathcal{I}_L)$ macro rules in llD^{Full} that models assertion of facts provides certain properties that facilitates the retraction of facts. Lemma 9 states these properties of assertion. Similarly, retraction rules and absorption rules are defined in a manner which provides us some important properties that help to guarantee the correctness of retraction. Lemma 10 formally asserts these properties.

LEMMA 9 (Properties of Assertion). *Given a Datalog Program* $\mathcal{P}$*, for any* llD^{Full} *state* $\Delta, \Delta^\mathcal{R}, \Delta^\sharp$*, and for some* Δ'*, if* $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \Delta, \Delta^\mathcal{R}, \Delta^\sharp \longrightarrow \bigotimes \Delta'$ *and* $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), \Delta')$ *we have the following properties:*

1. *For each* $a \in \Delta'$ *such that* $a \in herbrand_I(\mathcal{P})$ *(i.e.* a *is inferred), there must exist some retraction rule* $\&_{i=1}^n(\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h}) \in \Delta'$ *such that* $a = h$.
2. *For each retraction rule* $\&_{i=1}^n(\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h}) \in \Delta'$*, we must have for each* $\tilde{a}_i$ *such that equal and opposite* $a_i \in \Delta'$.
3. *For each retraction rule* $\mathcal{R} = \&_{i=1}^n(\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h})$*,* $\mathcal{R} \in \Delta'$ *if and only if* $B^\sharp \in \Delta'$ *and* $r(\vec{t}) \in \Delta'$ *as well.*
4. *Always exists some* Δ' *such that* $Quiescent((\mathcal{P}, \mathcal{A}_\mathcal{P}), \Delta')$.

Proof: The proof of these properties rely entirely on the definition of inference rules in $[\![\mathcal{P}]\!]$ (Figure 11). Recall that they are of the form: $\forall \vec{x}.\ \bigotimes B \multimap_{r(\vec{x})} h \otimes \bigotimes B \otimes \bigotimes B^\sharp \otimes \mathcal{R}$ such that $\mathcal{R} = \&_{i=1}^n(\tilde{a}_i \otimes r(\vec{x}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h})$. Details of the proof for each property follow.

1. For any fact $a \in \Delta'$, if a is not a base fact then it must be the product of an application of an inference rule instance from $[\![\mathcal{P}]\!]$. As such, by the definition of inference rule, for every $h = a$ fact inferred, we have a corresponding retraction rule $\mathcal{R} = \&_{i=1}^{n}(\tilde{a}_i \otimes r(\vec{x}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{a})$ as a by-product. Hence we have property (1).
2. Since, by definition, retraction rules are only introduced by inference rules, each inference rule $\forall \vec{x}.\ \bigotimes B \multimap_{r(\vec{x})} h \otimes \bigotimes B \otimes \bigotimes B^\sharp \otimes \mathcal{R}$ introduces retraction rule $\mathcal{R}$ such that each $\tilde{a}_i$ that appears in each implication of the retraction rule is the equal and opposite of a unique atom in B. Hence we have property (2).
3. Similarly, since retraction rules are only introduced by inference rules, by the definition of inference rule, each retraction rule $\mathcal{R} = \&_{i=1}^{n}(\tilde{a}_i \otimes r(\vec{x}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h})$ is introduced explicitly with $B^\sharp$ as accompanying consequents, while the formulation of $(\multimap_L^-)$ rule dictates the introduction of inhibiting atom $r(\vec{t})$. Hence we have property (3).
4. We argue that all inference rules are guarded implication rules inhibited by a unique rule instance $r(\vec{t})$. Since all well-formed Datalog program we consider have finite number of clauses and finite constants, there are a finite number of rule instances $r(\vec{t})$. As such, we can apply a finite number of $(\mathcal{I}_L)$ macro derivation steps and hence all derivations modeling assertion will reach quiescence.

This concludes the proof of this lemma. □

LEMMA 10 (Properties of Retraction). *Given a Datalog program $\mathcal{P}$, for any llD^{Full} state that represent an intermediate state of retraction $\Delta, \Delta^{\mathcal{R}}, \Delta^\sharp$ such that $\Delta = [\![F]\!], [\![a]\!], [\![F']\!], [\![\tilde{F'}]\!], [\![\tilde{a}]\!]$ and $F, F' \subseteq herbrand(\mathcal{P})$ and $a \in herbrand(\mathcal{P})$, and for some Δ', if we have the sequent $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; \Delta, \Delta^{\mathcal{R}}, \Delta^\sharp \longrightarrow \bigotimes \Delta'$, we have the following properties:*

1. *For each retraction rule $\mathcal{R} = \&_{i=1}^{n}(\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h})$, $\mathcal{R} \in \Delta'$ if and only if $B^\sharp \in \Delta'$ and $r(\vec{t}) \in \Delta'$.*
2. *If $\tilde{a} \notin \Delta'$, then there does not exist any retraction rule $\&_{i=1}^{n}(\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h}) \in \Delta'$, such that $\tilde{a} = \tilde{a}_i$ for some $i \geq 1$ and $i \leq n$.*
3. *There exists some Δ' such that $Quiescent((\mathcal{P}, \mathcal{A}_\mathcal{P}), \Delta')$.*
4. *If $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), \Delta')$, then there not exists any $\tilde{b} \in \Delta'$.*

Proof: The proofs of these properties rely on guarantees provided by assertions (Lemma 9) and the definition of retraction rules and absorption rules (Figure 13 and 14 respectively). Details of the proof for each property follow.

1. Since all retraction rules are only introduced by inference rules and $\Delta, \Delta^{\mathcal{R}}, \Delta^\sharp$ is a reachable state, property 3 of Lemma 9 guarantees that initially, each retraction rule in $\&_{i=1}^{n}(\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h}) \in \Delta^{\mathcal{R}}$ has a complete set of witnesses in $\Delta^\sharp$ (i.e. $r(\vec{t}), B^\sharp \subseteq \Delta^\sharp$). During retraction, each application of a retraction $\mathcal{R}$ consumes the retraction rule itself as well as the exact set of witnesses $B^\sharp$ and $r(\vec{t})$ (and nothing more) which was introduced together with $\mathcal{R}$ in the same inference rule instance. Hence, the property that any remaining retraction rule has its full set of witnesses in the context is preserved throughout intermediate retraction states.
2. If $\tilde{a} \notin \Delta'$, absorption rule $a \otimes \tilde{a} \multimap_{a^\sharp} (a^\sharp \multimap 1)$ must have been applied, since it is the only way to consume an $\tilde{a}$. In order for the absorption rule to apply, the context must not have had any occurrence of $a^\sharp$, and the only way to have achieved this is that all retraction rules $\&_{i=1}^{n}(\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h})$ such that $\tilde{a} = \tilde{a}_i$ for some $i \geq 1$ and $i \leq n$ and $a^\sharp \in B^\sharp$ were applied to consume all instance of $a^\sharp$, hence allowing the absorption rule to be applied.
3. We argue that retraction rules are linear, hence there always exist a derivation which involves an exhaustive and finite application of the retraction macro steps $(\mathcal{R}_L)$. Since we can apply a finite number of $(\mathcal{R}_L)$ rules, we can generate a finite numbers of retraction atoms $\tilde{a}$. As such, we can apply a finite number of absorption $(\mathcal{A}_L)$ macro derivation steps as well. Yet in retraction, we may need to reassert facts via $(\mathcal{I}_L)$. By property (4) of Lemma 9 we can conclude that there are also finite a number of reassertion steps. Hence we can always reach quiescence.
4. From 1, we argue that all retraction rules at any intermediate reachable states always have their accompanying witnesses $B^\sharp$ and $r(\vec{t})$. As such we can exhaustively apply retraction rules to remove all witnesses inhibiting absorption rules from being applicable, after which all $\tilde{b} \in \Delta$ will be consumed. Hence if Δ' is quiescent, we must have removed all retraction atom $\tilde{b}$.

This concludes the proof of this lemma. □

Inserting a set of retraction atoms $\tilde{\mathcal{B}}$ in a llD^{Full} state $[\![F]\!], [\![\mathcal{B}]\!], \Delta^{\mathcal{R}}, \Delta^\sharp$ will ultimately result in the derivation of the state Δ_1 which excludes $[\![\mathcal{B}]\!]$. Recall that we disallow the insertion of a retraction atom $\tilde{b}$ if b does not exist in the state.

THEOREM 11 (Correctness of Retraction). *Given a Datalog Program $\mathcal{P}$ and a base fact $b \in herbrand_B(\mathcal{P})$ and llD^{Full} states Δ_1,*

1. *Exists some state $[\![F]\!], \Delta^{\mathcal{R}}, \Delta^\sharp$, and Δ_1,Δ_2, such that*
   $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F]\!], [\![b]\!], \Delta^{\mathcal{R}}, \Delta^\sharp \longrightarrow \bigotimes \Delta_1 \otimes \bigotimes \Delta_2$ *and*
   $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), (\Delta_1, \Delta_2))$ *and*
   $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F]\!], \Delta^{\mathcal{R}}, \Delta^\sharp \longrightarrow \bigotimes \Delta_1$ *and*
   $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), \Delta_1)$ *and*
   $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F]\!], [\![b]\!], \Delta^{\mathcal{R}}, \Delta^\sharp, [\![\tilde{b}]\!] \longrightarrow \bigotimes \Delta_1$
2. *For all states $[\![F]\!], \Delta^{\mathcal{R}}, \Delta^\sharp$, there exists some Δ_1,Δ_2, such that*
   if $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F]\!], [\![b]\!], \Delta^{\mathcal{R}}, \Delta^\sharp, [\![\tilde{b}]\!] \longrightarrow \bigotimes \Delta_1$
   and $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), \Delta_1)$
   then $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F]\!], [\![b]\!], \Delta^{\mathcal{R}}, \Delta^\sharp \longrightarrow \bigotimes \Delta_1 \otimes \bigotimes \Delta_2$
   and $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), (\Delta_1, \Delta_2))$
   and $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F]\!], \Delta^{\mathcal{R}}, \Delta^\sharp \longrightarrow \bigotimes \Delta_1$

Proof: First, by relying on property 4 of Lemma 9, we can conclude that there exists an assertion state Δ_1, Δ_2 such that $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), (\Delta_1, \Delta_2))$ and by property 3 of Lemma 10 we can conclude that there exists a retraction state Δ_1 such that $Quiescent((\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}), \Delta_1)$. We now need to show that there exist such quiescent assertion and retraction states where $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F]\!], [\![b]\!], \Delta^{\mathcal{R}}, \Delta^\sharp, [\![\tilde{b}]\!] \longrightarrow \bigotimes \Delta_1$. We assume that $[\![\mathcal{P}]\!], \mathcal{A}_\mathcal{P}; [\![F]\!], [\![b]\!], \Delta^{\mathcal{R}}, \Delta^\sharp, [\![\tilde{b}]\!] \longrightarrow \bigotimes \Delta_3$ for some Δ_3, then proceed to show that if $\Delta_1 = \Delta_3$, our assumption still holds. Property 4 of Lemma 10 states that quiescent retraction states have no retraction atoms, hence we know for sure Δ_3 contains no retraction atoms. By property 2 of Lemma 10, since $\tilde{b} \notin \Delta_3$, we can safely assert that no retraction rule $\mathcal{R} = \&_{i=1}^{n}(\tilde{a}_i \otimes r(\vec{t}) \otimes \bigotimes B^\sharp \multimap \tilde{a}_i \otimes \tilde{h}) \in \Delta^{\mathcal{R}}$ such that $\tilde{a}_i = \tilde{b}$ for some i is in Δ_3 as well. With all such retraction rule removed, property 1 of Lemma 10 further states that corresponding witnesses of these retraction rules are not present in Δ_3 either. Finally, we know that to remove all retraction atoms $\tilde{a}$ from the context, Δ_3 was the result of several $(\mathcal{A}_L)$

macro derivation steps that removed b and its consequent facts. Hence, we can conclude that Δ_3 can be intermediately obtained by removing b and all its consequents (facts, retraction rules and witnesses) from $[\![F]\!], \Delta^{\mathcal{R}}, \Delta^{\sharp}$, yielding some Δ_4 which contains no retraction atoms. From Δ_4 we can apply ($\mathcal{I}_L$) macro steps to reassert facts common to b's consequents and by Theorem 5 we can guarantee that we can find a unique quiescent state Δ_3 since this quiescent state must be unique, hence $\Delta_1 = \Delta_3$. □

Corollary 12 formally asserts the bridge between our llD^{Full} interpretation of Datalog in linear logic LV^{obs-} and the traditional interpretation (Figure 1 and 12). It is directly derived from Theorems 8 and 11.

COROLLARY 12 (Correctness). *Given a Datalog Program $\mathcal{P}$, for reachable states $\Delta_1, \Delta_1^{\mathcal{R}}, \Delta_1^{\sharp}$ and $\Delta_2, \Delta_2^{\mathcal{R}}, \Delta_2^{\sharp}$ such that $\Delta_1 = [\![\mathcal{P}(\mathcal{B}_1)]\!]$ and $\Delta_2 = [\![\mathcal{P}(\mathcal{B}_2)]\!]$, then we have the following:*

$$(\Delta_1, \Delta_1^{\mathcal{R}}, \Delta_1^{\sharp}) \overset{\alpha}{\Longrightarrow}{}^{LL}_{[\![\mathcal{P}]\!]} (\Delta_2, \Delta_2^{\mathcal{R}}, \Delta_2^{\sharp}) \text{ iff } \mathcal{P}(\mathcal{B}_1) \overset{\alpha}{\Longrightarrow}_{\mathcal{P}} \mathcal{P}(\mathcal{B}_2)$$

6. Related Work

Early work on Datalog focused on efficient support for querying the logical consequences of a static fact base [4] although some of it also supported assertions. Examples include the semi-naive strategy [2], QSQ [19], and magic sets [3]. Recently, there has been a trend in the parallel and distributed programming research community to develop high-level declarative programming languages which are based on (or inspired by) Datalog. For instance, NetLog [10] and Network Datalog (*NDLog*) [14] target the development of networking protocols, Meld [1] for programming of large ensembles of embedded robotic devices, and the system in [13] enables distributed user authentication. To varying extents, all of these works credit the declarative nature of Datalog as a main motivation, mitigating the notorious complexities of parallel and distributed programming. These new application areas are characterized by frequent assertions *and* retractions. For example, the Nexcel language of [5] uses a dialect of Datalog to compute derived data in a spreadsheet as the user types.

These works contributed efficient algorithms and implementations to efficiently compute inferential closures incrementally as facts are dynamically added or removed [1, 7, 11, 14, 16]. However we are unaware of any work that bridges these algorithms back to formal logic. As such, our efforts here complement these developments.

7. Conclusions and Future Work

In this paper, we have given a logical interpretation of Datalog in linear logic, which captures assertion and retraction of facts within the confines of logic. It directly maps assertion and retraction steps to derivations in linear logic, thereby functioning not only as an abstract specification, but as an operational blueprint for practical implementations of Datalog systems. Indeed, we prove the llD^{Full} interpretation corresponds with the traditional logic interpretation of Datalog programs.

In the future, we plan to implement a Datalog system based on llD^{Full} and exploiting ideas in [6]. We believe that this can be a foundationally strong approach for deriving expressive and powerful concurrent logic programming languages to be applied to highly parallel systems such as Claytronics [1]. We also wish to further investigate re-assertion during retraction, briefly discussed in Section 5.4. Specifically, we are interested in exploring the possibility of deriving a linear logic interpretation of Datalog programs that removes the need for re-assertion during fact retraction, thereby providing a more streamlined and efficient operational semantics for practical implementation. We intend to explore the use of other variants of linear logic, in the attempt to streamline our interpretation of Datalog. (E.g. representing Datalog facts as consumable subexponentials [17] and using polarized presentations of linear logic).

References

[1] M. P. Ashley-Rollman, P. Lee, S. C. Goldstein, P. Pillai, and J. D. Campbell. A language for large ensembles of independently executing nodes. In *Proc. of ICLP'09*, volume 5649/2009, pages 265–280. Springer-Verlag, 2009.

[2] I. Balbin and K. Ramamohanarao. A generalization of the differential approach to recursive query evaluation. *J. Log. Program.*, 4(3):259–262, 1987.

[3] F. Bancilhon, D. Maier, Y. Sagiv, and J. D. Ullman. Magic sets and other strange ways to implement logic programs (extended abstract). In *Proc. of PODS'86*, pages 1–15. ACM, 1986.

[4] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. on Knowl. and Data Eng.*, 1(1):146–166, March 1989.

[5] I. Cervesato. NEXCEL, a Deductive Spreadsheet. *The Knowledge Engineering Review*, 22:221–236, 2007.

[6] I. Cervesato and A. Scedrov. Relating state-based and process-based concurrency through linear logic. *Inf. Comput.*, 207(10):1044–1077, 2009.

[7] F. Cruz, M. P. Ashley-Rollman, S. C. Goldstein, Ricardo Rocha, and F. Pfenning. Bottom-up logic programming for multicores. In Vítor Santos Costa, editor, *Proc. of DAMP 2012 - Short Papers*. ACM Digital Library, January 2012.

[8] H. Gallaire, J. Minker, and J. M. Nicolas. Logic and databases: A deductive approach. *ACM Comput. Surv.*, 16(2):153–185, June 1984.

[9] J. Y. Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

[10] S. Grumbach and F. Wang. Netlog, a rule-based language for distributed programming. In *Proc. of PADL'10*, volume 5937/2010, pages 88–103. Springer-Verlag, 2010.

[11] A. Gupta, I. S. Mumick, and V. S. Subrahmanian. Maintaining views incrementally. In *Proc. of SIGMOD'93*, pages 157–166. ACM, 1993.

[12] E. S. L. Lam and I. Cervesato. Modeling datalog assertion and retraction in linear logic (full-version). Technical Report CMU-CS-QTR-113/CMU-CS-12-126, Carnegie Mellon University, Jun 2012.

[13] Ninghui Li and John C. Mitchell. Datalog with constraints: A foundation for trust-management languages. In *Proc. of PADL'03*, volume 2562/2003, pages 58–73. Springer-Verlag, 2003.

[14] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative networking: language, execution and optimization. In *Proc. of SIGMOD '06*, pages 97–108. ACM, 2006.

[15] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Proc. of APAL'91*, 51:125–157, 1991.

[16] V. Nigam, L. Jia, B. T. Loo, and A. Scedrov. Maintaining distributed logic programs incrementally. In *Proc. of PPDP'11*, pages 125–136. ACM, 2011.

[17] V. Nigam and D. Miller. Algorithmic specifications in linear logic with subexponentials. In *Proc. of PPDP'09*, pages 129–140. ACM, 2009.

[18] F. Pfenning. Structural cut elimination. In *Proc. of LICS'95*, pages 156–166. IEEE Press, 1995.

[19] L. Vieille. Recursive axioms in deductive databases: The query/subquery approach. In *Expert Database Conf.*, pages 253–267. Benjamin Cummings, 1986.