Revisiting Semiring Provenance for Datalog
Camille Bourgaux, Pierre Bourhis, Liat Peterfreund, Michaël Thomazo

To cite this version:

HAL Id: hal-03771031
https://hal.science/hal-03771031
Submitted on 7 Sep 2022
Revisiting Semiring Provenance for Datalog

Camille Bourgaux¹, Pierre Bourhis², Liat Peterfreund³, Michaël Thomazo¹

¹DI ENS, ENS, CNRS, PSL University & Inria, Paris, France
²CRIStAL, CNRS, University of Lille, Inria, Lille, France
³LIGM, CNRS, Université Gustave Eiffel, ENPC, Paris, France

camille.bourgaux@ens.fr, pierre.bourhis@inria.fr, liat.peterfreund@univ-eiffel.fr, michael.thomazo@inria.fr

Abstract

Data provenance consists in bookkeeping meta information during query evaluation, in order to enrich query results with their trust level, likelihood, evaluation cost, and more. The framework of semiring provenance abstracts from the specific kind of meta information that annotates the data. While the definition of semiring provenance is uncontroversial for unions of conjunctive queries, the picture is less clear for Datalog. Indeed, the original definition might include infinite computations, and is not consistent with other proposals for Datalog semantics over annotated data. In this work, we propose and investigate several provenance semantics, based on different approaches for defining classical Datalog semantics. We study the relationship between these semantics, and introduce properties that allow us to analyze and compare them.

1 Introduction

Datalog is a rule language widely studied both in the database community, where it is seen as a query language, and in the KR community, as an ontology language.

In relational databases, the framework of semiring provenance was introduced to generalize computations over annotated databases, e.g., the semantics of probabilistic databases (Senellart 2017), the bag semantics, lineage or why-provenance (Cheney, Chiticariu, and Tan 2009). In this framework, the semantics of positive relational algebra queries over databases annotated with elements of any commutative semiring is inductively defined on the structure of the query (Green, Karvounarakis, and Tannen 2007; Green and Tannen 2017). Provenance semirings are expressions (such as polynomials) built from variables associated to each tuple of the database (Green 2009). A provenance expression provides a general representation of how tuples have been used to derive a query result, and can be faithfully evaluated in any semiring in which the considered provenance semiring can be homomorphically embedded.

Semiring provenance has also been studied for Datalog queries, for which it was defined based on the set of all derivation trees for the query (Green, Karvounarakis, and Tannen 2007; Deucht et al. 2014; Deucht, Gilad, and Moskovitch 2018). However, this definition seems less axiomatic than in the case of relational databases. Indeed, there may be infinitely many derivation trees, leading to infinite provenance expressions, while Datalog programs have finite models that can be computed efficiently (Abiteboul, Hull, and Vianu 1995). A consequence is that this definition is valid only for a restricted class of semirings, namely ω-continuous. Recently, Dannert et al. (2021) restrict the semiring even further by considering fully-chain complete semirings in order to extend provenance definition to logical languages featuring negation and fixed-point. Even if numerous useful semirings are ω-continuous, or can be extended to a such semiring, infinite provenance expressions may be considered unintuitive in some cases. Consider, for example, the counting semiring (i.e., natural numbers with standard operations) for which provenance of positive relational algebra queries corresponds to their bag semantics. This semiring can be extended to an ω-continuous one by adding ∞ to the natural numbers, hence providing a way to capture the bag semantics for Datalog queries (Mumick, Pirahesh, and Ramakrishnan 1990; Green, Karvounarakis, and Tannen 2007). However, query answers having infinite multiplicities may not seem very natural or informative. Moreover, alternative bag semantics for languages close to Datalog have been defined, and would not lead to such infinite multiplicities when applied to Datalog. This is in particular the case of the bag semantics for ontology-based data access (Nikolaou et al. 2017; Nikolaou et al. 2019), which corresponds to one of the two semantics proposed for source-to-target tuple generating dependencies in the context of data exchange (Hernich and Kolaitis 2017). Interestingly, these bag semantics are not based on derivation trees but are model-theoretic semantics: they define annotated interpretations, and conditions for rules satisfaction over such interpretations. Such model-theoretic semantics have also been used in other contexts to evaluate Datalog and variants over annotated databases, such as fuzzy Datalog (Ach and Kiss 1995) or description logic knowledge bases annotated with provenance tokens (Calvanese et al. 2019; Bourgaux et al. 2020). Finally, yet other semantics definitions have been proposed for some use cases. For instance, Zhao, Subotic, and Scholz (2020) consider minimal depth proof trees, which correspond to a Datalog evaluation algorithm, with the intended use of understanding the computation of the result, and guiding debugging.

The fact that the above semantics are not encompassed by the definition of semiring provenance for Datalog, along with the need of handling infinite computations which are entailed by this definition, motivate us to investigate alterna-
tive natural semantics that might be a better fit in different contexts.

In this paper we introduce several natural provenance semantics for Datalog over annotated data. Our definitions are based on different classical approaches: model-theoretic, execution-based and proof tree-based. They capture the semantics mentioned previously, and are inspired by practical needs. For instance, our semantics definition based on minimal depth derivation trees capture the behavior of Datalog engines, such as Soufflé (Soufflé 2020), that store only minimal depth derivation trees instead of storing them all (which might be impossible in case there are infinitely many); Our semantics based on non-recursive derivation trees (in which a fact is not derived from itself) resembles the approach taken by some graph query languages, e.g., SPARQL and Cypher, to handle queries with possibly infinite outputs by allowing to explicitly restrict the output to include only simple paths. In addition, some of the semantics suggested in the paper are closely related to paradigms for weighted reasoning in the context of words and trees (Esparza and Luttenberger 2011; Stüber and Vogler 2008).

After defining these different semantics, we study under which conditions they coincide and investigate their connections. We then provide a general framework for defining such provenance semantics, and present several properties relevant for provenance semantics that allow us to compare them. We briefly discuss some complexity issues in conclusion. Proofs and additional discussion are available in the appendix of (Bourgaux et al. 2022).

2 Preliminaries

2.1 Datalog

We use the standard Datalog settings (cf. (Abiteboul, Hull, and Vianu 1995) part D).

Syntax Let P, C, and V be mutually disjoint, possibly infinite sets of predicates, constants, and variables respectively. Elements of C ∪ V are called terms. An atom has the form p(t_1, ..., t_m) where p ∈ P is an n-ary predicate, and t_i’s are terms. A fact (or ground atom) is a variable-free atom. A (Datalog) rule is an expression: ∀x∀y(φ(x,y) → ψ(x)) where x and y are tuples of variables and φ(x,y) and ψ(x) are conjunctions of atoms whose variables are x ⨆ y and x respectively. We call φ(x,y) and ψ(x) the body and head of the rule, respectively. From now on, we assume that rules are in normalized form, i.e., the head consists of a single atom H(x), and quantifiers are implicit. The domain D(A) of a set A of atoms is the set of terms that appear in its atoms.

A database D is a finite set of facts, and a Datalog program (or ontology) Σ is a finite set of Datalog rules. The schema of D (resp. Σ) denoted S(D) (resp. S(Σ)) is the set of predicates that appear in its atoms.¹

Semantics The semantics of Datalog can classically be defined in three ways: through models, fixpoints or derivation trees. All three definitions rely on the notion of homomorphism: a homomorphism from a set A of atoms to a set B of atoms is a function h : D(A) → D(B) such that h(t) = t for all t ∈ C, and p(t_1, ..., t_m) ∈ A implies h(p(t_1, ..., t_m)) := h(p(t_1, ..., t_m)) ∈ B. We denote by h(A) the set {h(p(t_1, ..., t_m)) | p(t_1, ..., t_m) ∈ A}. The homomorphism definition is extended to conjunctions of atoms by viewing them as the sets of atoms they contain.

A set I of facts is a model of a rule r := φ(x,y) → ψ(x), denoted by I |= r, if every homomorphism h from φ(x,y) to I is also a homomorphism from ψ(x) to I; it is a model of a Datalog program Σ if I |= r for every r ∈ Σ; it is a model of a database D if D ⊆ I. A fact α is entailed by D and Σ, denoted Σ, D |= α, if α ∈ I for every model I of Σ and D.

Example 1. Let Σ contain the rules B(x) → A(x), R(n, x) ∧ A(x) → B(x), and R(n, y) → R(y, x), and D := {(B(a), B(b), R(a, b), R(b, a))}. Each model of D and Σ contains all facts in D as well as A(a) and A(b), which are thus entailed by D, Σ.

An equivalent way to define the entailment of a fact α by D and Σ is to check if there is a homomorphism from α to a specific model, defined as the least fixpoint containing D of the immediate consequence operator: An immediate consequence for D and Σ is either α ∈ D, or α such that there exists a rule r := φ(x,y) → ψ(x) and a homomorphism h from φ(x,y) to D such that h(ψ(x)) = α.

Finally, a third definition relies on derivation trees.

Definition 1 (Derivation Tree). A derivation tree t of a fact α w.r.t. a database D and a program Σ is a finite tree whose leaves are labeled by facts from D and non-leaf nodes are labeled by triples (p(t_1, ..., t_m), r, h) where

• p(t_1, ..., t_m) is a fact over the schema S(Σ);
• r is a rule from Σ of the form φ(x,y) → p(x);
• h is a homomorphism from φ(x,y) to the facts of the labels of the node children, such that h(p(φ(x,y))) = p(t_1, ..., t_m);
• there is a bijection f between the node children and the atoms of φ(x,y), such that for every q(z) ∈ φ(x,y), f(q(z)) is of the form (h(q(z)), r', h') or is a leaf labeled by h(φ(x,y)).

Moreover, if (p(t_1, ..., t_m), r, h) or p(t_1, ..., t_m) is the root of t, then p(t_1, ..., t_m) = α.

Example 2. Let Σ contain r_1 := R(x, y) → H(x, y), r_2 := R(x, y) → H(x, y), and r_3 := S(x, y, z) ∧ S(x, z, y) → H(x). If D = {R(a, a), S(a, b, c), S(a, c, b)}, then the fact α := H(a, a) has the following derivation trees

(α, r_1, h) (α, r_2, h) (α, r_3, h_a) (α, r_3, h_b)

R(a, a) R(a, a) S(a, b, c)S(a, c, b)S(a, c, b)S(a, b, c)

where h(x) = h(y) = a, h_3(x) = a, h_3(y) = b, h_3(z) = c

and h_3'(x) = a, h_3'(y) = c, h_3'(z) = b.

Note that when the program at hand is recursive (i.e., the dependency graph of its predicates contains cycles) a fact may have infinitely many derivation trees. Figure 1 depicts some of the infinitely many derivation trees of A(a) from Example 1. In this example, and from this point on, we omit rules and homomorphisms from trees where there is no ambiguity.
A (a) A (a) A (a)  
B (a) B (a) B (a)  
R (a, b) A (b) R (a, b) A (b)  
B (b) R (b, a) B (b)  

Figure 1: Some derivation trees of \( A (a) \) in Example 1.

Queries A conjunctive query (CQ) is an existentially quantified formula \( \exists \vec{y} \phi (\vec{x}, \vec{y}) \) where \( \phi (\vec{x}, \vec{y}) \) is a conjunction of atoms with variables in \( \vec{x} \cup \vec{y} \): a union of conjunctive queries (UCQ) is a disjunction of CQs (over the same free variables). A query is Boolean if it has no free-variables. A set of facts \( I \) satisfies a Boolean CQ (BCQ) \( q := \exists \vec{y} \phi (\vec{y}) \), written \( I \models q \), if and only if there is a homomorphism from \( \phi (\vec{y}) \) to \( I \). A BCQ \( q \) is entailed by a Datalog program \( \Sigma \) and database \( D \), written \( \Sigma, D \models q \), if and only if \( \Sigma \models q \) for every model \( I \) of \( \Sigma \) and \( D \). Note that \( \Sigma, D \models q \) if and only if \( \Sigma \cup \{ \phi (\vec{y}) \to \text{goal} \}, D \models \text{goal} \), where goal is a nullary predicate such that goal \( \not\in \Sigma \cup \{ \phi (\vec{y}) \to \text{goal} \} \). A tuple of constants \( \vec{a} \) is an answer to a BCQ \( q (\vec{x}) := \exists \vec{y} \phi (\vec{x}, \vec{y}) \) over \( \Sigma \) and \( D \) if \( \vec{a} \) and \( \vec{x} \) have the same arity and \( \Sigma, D \models \phi (\vec{a}) \) where \( \phi (\vec{a}) \) is the BCQ obtained by replacing the variables from \( \vec{x} \) with the corresponding constants from \( \vec{a} \). When \( \Sigma = \emptyset \), it amounts to the existence of a homomorphism from \( \phi (\vec{a}) \) to \( D \), which corresponds to the semantics of CQs over relational databases.

2.2 Annotated Databases

To equip databases with extra information, their facts might be annotated with, e.g., trust levels, clearance degree required to access them, or identifiers to track how they are used.

In the framework of semiring provenance, annotations are elements of algebraic structures known as commutative semirings. A semiring \( \mathbb{K} = (K, +_K, \times_K, 0_K, 1_K) \) is a set \( K \) with distinguished elements \( 0_K \) and \( 1_K \), equipped with two binary operators: \( +_K \), called the addition, which is an associative and commutative operator with identity \( 0_K \), and \( \times_K \), called the multiplication, which is an associative operator with identity \( 1_K \). It also holds that \( \times_K \) distributes over \( +_K \), and \( 0_K \) is annihilating for \( \times_K \). When multiplication is commutative, the semiring is said to be commutative. We use the convention according to which multiplication is applied before addition to omit parentheses. We omit the subscript of operators and distinguished elements when there is no ambiguity.

Definition 2. An annotated database is a triple \( (D, \mathbb{K}, \lambda) \) where \( D \) is a database, \( \mathbb{K} = (K, +_K, \times_K, 0_K, 1_K) \) is a semiring, and \( \lambda : D \to K \setminus \{0_K\} \) maps facts into semiring elements different from \( 0_K \).

Example 3 (Ex. 1 cont’d). The semiring \( \mathbb{N} = (\mathbb{N}, +, \times, 0, 1) \) of the natural numbers equipped with the usual operations is used for bag semantics. The tropical semiring \( \mathbb{T} = (\mathbb{R}^\infty_+, \min, +, \infty, 0) \) is used to compute minimal-cost paths. We define \( \lambda_\mathbb{N} : D \to \mathbb{N} \setminus \{0\} \) by \( \lambda_\mathbb{N}(B(a)) = 3 \), \( \lambda_\mathbb{N}(R(b, a)) = 1, \lambda_\mathbb{N}(R(b, a)) = 2 \), and \( \lambda_\mathbb{T}(R(b, a)) = 1 \). And \( \lambda_\mathbb{T}(R(b, a)) = 5 \), \( \lambda_\mathbb{T}(R(b, a)) = 2 \).

We next list some possible properties of semirings. A semiring is +-idempotent (resp. \( \times \)-idempotent) if for every \( a \in K, a + a = a \) (resp. \( a \times a = a \)). It is absorptive if for every \( a, b \in K, a \times b + a = a \). It is positive if for every \( a, b \in K, a + b = 0 \) if and only if \( a = 0 \) or \( b = 0 \), and \( a + b = 0 \) if and only if \( a = b = 0 \). Finally, an important class is that of \( \omega \)-continuous commutative semirings in which infinite sums are well-defined. Given a semiring, we define the binary relation \( \sqsubseteq \) such that \( a \sqsubseteq b \) if and only if there exists \( c \in K \) such that \( a + c = b \). A commutative semiring is \( \omega \)-continuous if \( \sqsubseteq \) is a partial order, every \( \omega \)-chain \( a_0 \sqsubseteq a_1 \sqsubseteq a_2 \ldots \) has a least upper bound \( \sup((a_i)_{i \in \mathbb{N}}) \), and for every \( a, a + \sup((a_i)_{i \in \mathbb{N}}) = \sup((a + a_i)_{i \in \mathbb{N}}) \) and \( a \times \sup((a_i)_{i \in \mathbb{N}}) = \sup((a \times a_i)_{i \in \mathbb{N}}) \).

The semantics of queries from the positive relational algebra, and in particular of UCQs, over annotated databases is defined inductively on the structure of the query (Green, Karvounarakis, and Tannen 2007). Intuitively, joint use of data (conjunction) corresponds to multiplication, and alternative use of data (union or projection) corresponds to addition.

Example 4 (Ex. 3 cont’d). The BCQ \( \exists \vec{y} \big( R(x, y) \land B(y) \big) \) is entailed from \( (D, \mathbb{N}, \lambda_\mathbb{N}) \) with multiplicity \( \lambda_\mathbb{N}(R(a, b)) \times \lambda_\mathbb{N}(B(b)) + \lambda_\mathbb{N}(R(b, a)) \times \lambda_\mathbb{N}(B(a)) = 5 \), and from \( (D, 1, \lambda_1) \) with minimal cost \( \min(\lambda_1(R(a, b)) + \lambda_1(T(R(b, a))) + \lambda_1(T(B(a)))) = 6 \).

A semantics of Datalog over annotated databases has been defined by Green, Karvounarakis, and Tannen (2007) using derivation trees, that we shall name the all-tree semantics. It associates to each fact \( \alpha \) entailed by \( \Sigma \) and \( D \) the following sum, where \( \mathcal{P}_D^\Sigma(\alpha) \) is the set of all derivation trees for \( \alpha \) w.r.t. \( \Sigma \) and \( D \) and \( \Lambda(t) \) := \( \prod_t \mathcal{P}_t(\alpha) \) is the \( \mathbb{K} \)-annotation of the derivation tree \( t \) since \( \mathbb{K} \) is commutative, the result of the product is well-defined.

\[
\mathcal{P}_D^\Sigma(\alpha) := \sum_{t \in \mathcal{T}_D^\Sigma(\alpha)} \Lambda(t).
\]

Since \( \mathcal{P}_D^\Sigma(\alpha) \) may be infinite, \( \mathcal{P}_D^\Sigma \) is well-defined for all \( \Sigma, (D, \mathbb{K}, \lambda) \) and \( \alpha \) only in the case where \( \mathbb{K} \) is \( \omega \)-continuous.

Example 5 (Ex. 3 cont’d). The fact \( \alpha := A(a) \) is entailed with minimal cost: \( \mathcal{P}_D^\Sigma(\alpha) := \min_{t \in \mathcal{T}_D^\Sigma(\alpha)} \sum_t \mathcal{P}_t(\alpha) \) is a leaf of \( \mathcal{P}_t(\alpha) \) is 3. Since \( \mathbb{N} \) is not \( \omega \)-continuous, \( \mathcal{P}_D^\Sigma(\alpha) \) is not defined.

2.3 Provenance Semirings

Provenance semirings have been introduced to abstract from a particular semiring by associating a unique provenance token to each fact of the database, and building expressions that trace their use. Given a set \( X \) of variables that annotate the database, a provenance semiring \( \mathbb{P}(X) \) is a semiring over a space of provenance expressions with variables from \( X \).

Various such semirings were introduced in the context of relational databases (Green 2009): The most expressive annotations are provided by the provenance polynomials semiring \( \mathbb{N}[X] := (\mathbb{N}[X], +, \times, 0, 1) \) of polynomials with coefficients from \( \mathbb{N} \) and variables from \( X \), and the usual operations. Less general provenance semirings include, for example, the semiring \( \mathbb{B}[X] := (\mathbb{B}[X], +, \times, 0, 1) \) of
polynomials with Boolean coefficients, and the semiring \( PosBool(X) := (PosBool(X), \lor, \land, \text{false}, \text{true}) \) of positive Boolean expressions.

In the Datalog context, it is important to allow for infinite provenance expressions, as there can be infinitely many derivation trees. A formal power series with variables from \( X \) and coefficients from \( K \) is a mapping that associates to each monomial over \( X \) a coefficient in \( K \). A formal power series \( S \) can be written as a possibly infinite sum \( S = \sum_{m \in \text{mon}(X)} S(m)m \) where \( \text{mon}(X) \) is the set of monomials over \( X \) and \( S(m) \) is the coefficient of the monomial \( m \). The set of formal power series with variables from \( X \) and coefficients from \( K \) is denoted \( K[[X]] \). Green, Karvounarakis, and Tannen (2007) define the Datalog provenance semiring as the semiring \( N^\infty[X] \) of formal power series with coefficients from \( N^\infty = N \cup \{\infty\} \).

A semiring homomorphism from \( K = (K, +_K, \times_K, 0_K, 1_K) \) to \( K' = (K', +_{K'}, \times_{K'}, 0_{K'}, 1_{K'}) \) is a mapping \( h : K \to K' \) such that \( h(0_K) = 0_{K'} \), \( h(1_K) = 1_{K'} \), and for all \( a, b \in K \), \( h(a +_K b) = h(a) +_{K'} h(b) \) and \( h(a \times_K b) = h(a) \times_{K'} h(b) \). A semiring homomorphism between \( \omega \)-continuous semirings is \( \omega \)-continuous if it preserves lower bounds: \( h(\sup((a_i)_{i \in I})) = \sup(h((a_i)_{i \in I})) \).

Following Deutch et al. (2014), we say that a provenance semiring \( Prov(X) \) specializes correctly to a semiring \( K \), if any valuation \( \nu : X \to K \) extends uniquely to a \( \omega \)-continuous if \( Prov(X) \) and \( K \) are \( \omega \)-continuous) semiring homomorphism \( h : Prov(X) \to K \), allowing the computations for \( K \) to factor through the computations for \( Prov(X) \). A provenance semiring \( Prov(X) \) is universal for a set of semirings if it specializes correctly to each semiring of this set. Green, Karvounarakis, and Tannen (2007) showed that \( N[X] \) is universal for commutative semirings, and \( N^\infty[X] \) is universal for commutative \( \omega \)-continuous semirings.

### 3 Alternative Semantics

In this section we propose several natural ways of defining the semantics of Datalog over annotated databases, and investigate their connections. We have seen that the semantics of Datalog can equivalently be defined through models, fixpoints or derivation trees. The semantics we propose also fall into these three approaches. For presentation purposes, we see each semantics as a partial function \( \mathcal{P} \) that associates to a Datalog program \( \Sigma \), an annotated database \( (D, \mathcal{K}, \lambda) \), and fact \( \alpha \), a semiring element \( \mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \alpha) \).

#### 3.1 Model-Based Semantics

We first investigate two provenance semantics based on Datalog’s model-theoretic semantics. In both cases, we will define interpretations \( (I, \mu^I) \) where \( I \) is a set of facts and \( \mu^I \) is a function that annotates facts of \( I \), and formulate requirements for them to be models of \( \Sigma \) and \( (D, \mathcal{K}, \lambda) \), extending standard models of \( \Sigma \) and \( D \) with fact annotations.

**Annotated Model-based** Hernich and Kolaitis (2017) define two bag semantics in the context of data exchange: the **incognizant** and **cognizant** semantics. The difference between them arise from the two different semantics of bag union: the {\textit{incognizant}} semantics uses the maximum-based union, while the {\textit{cognizant}} semantics uses the sum-based union.

In more details, both semantics are based on the following semantics for source-to-target tuple generating dependencies (s-t tgds): a pair \((I, J)\) of source and target instances satisfies an s-t tgd \( q_1(\vec{x}) \to q_2(\vec{x}) \) if for every answer \( \vec{a} \) to \( q_1 \) over \( I \), \( \vec{a} \) is an answer to \( q_2 \) over \( J \) with at least the same multiplicity. Given a set of s-t tgds \( \Sigma \) and a source \( I \), a target \( J \) is an {\textit{incognizant solution}} for \( I \) w.r.t. \( \Sigma \) if \( (I, J) \) satisfies every s-t tgd in \( \Sigma \). It is a {\textit{cognizant solution}} if for every \( r \in \Sigma \), there is a target instance \( J_r \) such that \( (I, J_r) \) satisfies \( r \) and \( \mu_{J_r} \subseteq J \), where \( \mu \) denotes the sum-union of bags (i.e., the multiplicity of each element of the sum-union is equal to the sum of its multiplicities). The incognizant (resp. cognizant) {\textit{certain answers}} to a query \( q \) w.r.t. \( \Sigma \) on \( I \) are defined using bag intersection of the answers over the incognizant (resp. cognizant) solutions for \( I \) w.r.t. \( \Sigma \), i.e., the multiplicity of an answer is the minimum of its multiplicities over the solutions. Note that for BCQs, the only possible certain answer is the empty tuple.

For example, consider \( \Sigma = \{B(x) \to A(x), C(x) \to A(x)\} \) and \( D = \{(B(\alpha), 1), (C(\alpha), 1)\} \). Under the incognizant semantics, the multiplicity of the certain answer of the Boolean query \( A(a) \) w.r.t. \( \Sigma \) and \( D \) is 1 while under the cognizant semantics it is 2. Indeed, \( J = \{(A(a), 1)\} \) is an incognizant solution for \( D \) w.r.t. \( \Sigma \) as it satisfies both s-t tgds, but is not a cognizant solution as the sum of multiplicities that arise from the two rules is 2.

It is easy to show that the cognizant semantics is equivalent to \( P^{\text{AT}} \) on the counting semiring \( N = (N, +, \times, 0, 1) \), and thus coincides with the classical bag semantics for Datalog. However, we have seen that the incognizant and cognizant semantics differ. Moreover, note that in the field of ontology-based data access, the bag semantics defined by Nikolaou et al. for DL-Lite\( _R \) (2017; 2019) coincides with the incognizant semantics, thus disagrees with the classical Datalog bag semantics (Mumick, Pirahesh, and Ramakrishnan 1990; Green, Karvounarakis, and Tannen 2007).

We hence define a provenance semantics that coincides with these semantics when used with the counting semiring. Since it is based on greater lower bounds, it is defined on a restricted class of semirings.

Let \( \mathcal{K} = (K, +, \times, 0, 1) \) be a commutative \( \omega \)-continuous semiring such that for every \( K' \subset K \), the greatest lower bound \( \inf(K') \) of \( K' \) is well defined (i.e., there exists a unique \( z \in K \) such that \( z \lessdot x \) for every \( x \in K' \) and \( \forall z' \) such that \( z' \lessdot x \) for every \( x \in K' \) is such that \( z' \lessdot z \)). \( \Sigma \) be a Datalog program, and \( (D, \mathcal{K}, \lambda) \) be an annotated database. We define \( \mathcal{K} \)-annotated interpretations as pairs \((I, \mu^I)\) where \( I \) is a set of facts, and \( \mu^I \) is a function from \( I \) to \( K \). We say that a \( \mathcal{K} \)-annotated interpretation \( (I, \mu^I) \) is a model of \( \Sigma \) and \( (D, \mathcal{K}, \lambda) \), denoted by \( (I, \mu^I) \models (\Sigma, D, \mathcal{K}, \lambda) \), if

1. \( D \subseteq I \), and for every \( \alpha \in D \), \( \lambda(\alpha) \subseteq \mu^I(\alpha) \);
2. for every \( \phi(\vec{x}, \vec{y}) \to H(\vec{x}) \) in \( \Sigma \), whenever there is a homomorphism \( h : \phi(\vec{x}, \vec{y}) \to I \), then \( h(H(\vec{x})) \in I \) and \( \sum_{h' : \phi(\vec{x}, \vec{y}) \to I, h'(\vec{x}) = h(\vec{x})} \prod_{\beta \in h'(\phi(\vec{x}, \vec{y}))} \mu^I(\beta) \subseteq \mu^I(h(H(\vec{x}))) \).

The annotated model-based provenance semantics \( P^{\text{AM}} \) is
defined by
\[ \mathcal{P}^\text{an} (\Sigma, D, \mathbb{K}, \lambda, \alpha) := \inf \{ \mu^I(\alpha)(I, \mu^I) \mid (\Sigma, D, \mathbb{K}, \lambda) \} \].

**Proposition 1.** If the Datalog rules in \( \Sigma \) are (1) s-t tgds, or (2) formulated in DL-Lite \( R \), then for every \( \mathcal{C}Q \), \( \mathcal{P}^\text{an} (\Sigma \cup \{ q \rightarrow \text{goal} \}, D, \mathbb{N}, \lambda_q, \text{goal}) = \inf \{ \mu^I(\alpha)(I, \mu^I) \mid (\Sigma, D, \mathbb{K}, \lambda) \} \).

**Set-Annotated Model-based** We adapt the work on provenance for the description logics DL-Lite \( R \) and \( \mathcal{C}L^\omega \) (Calvanese et al. 2019; Bourgaux et al. 2020), where the semiring is assumed to be a \( \tau \times \text{idempotent} \) \( \omega \)-continuous semiring. 

Example 7. Let \( \Sigma = \{ R(x, y) \rightarrow \text{goal} \} \), \( D = \{ R(a, b), R(a, c) \}, \lambda_q(R(a, b)) = 2 \) and \( \lambda_q(R(a, c)) = 2 \). 

**Proposition 3.** If \( \mathbb{K} \) is a commutative \( + \)-idempotent \( \omega \)-continuous semiring, then for every \( \Sigma \), \( D, \mathbb{K}, \lambda, \alpha \), and \( \alpha \), \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{K}, \lambda, \alpha) = \mathcal{P}^\text{at} (\Sigma, D, \mathbb{K}, \lambda, \alpha) \).

Example 8. Let \( \lambda_X \) be an injective function from \( D \) to \( X \).

- A monomial occurs in \( \mathcal{P}^\text{at} (\Sigma, D, \mathbb{N}^\infty \times X, \lambda_X, \alpha) \) if and only if it occurs in \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}^\infty \times X, \lambda_X, \alpha) \).
- \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}^\infty \times X, \lambda_X, \alpha) \) is obtained by setting all non-zero coefficients to 1 in \( \mathcal{P}^\text{at} (\Sigma, D, \mathbb{N}^\infty \times X, \lambda_X, \alpha) \).

Another example where \( \mathcal{P}^\text{an} \) and \( \mathcal{P}^\text{at} \) differ on \( \mathbb{N}^\infty \times X \) is the following: Let \( \Sigma \) contain \( A(x) \rightarrow B(x) \rightarrow A(x) \), \( D = \{ A(a), B(a) \} \), \( \lambda_X(A(a)) = x \) and \( \lambda_X(B(a)) = y \). Since there are infinitely many derivation trees for \( A(a) \), \( \mathcal{P}^\text{at} (\Sigma, D, \mathbb{K}, \lambda_X, A(a)) = \infty \) while for \( \mathcal{P}^\text{an} \), \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}^\infty \times X, \lambda_X, A(a)) = x \), as \( \{ A(a), B(a) \} \) with both facts annotated with \( x \) (resp. \( y \)) is a (resp. set-)annotated model for \( \Sigma \) and \( D, \mathbb{N}^\infty \times X, \lambda_X \).

Note that \( \mathcal{P}^\text{at} \) and \( \mathcal{P}^\text{an} \) can still lead to infinite provenance expressions. Let \( \Sigma = \{ A(x) \land B(x) \rightarrow A(x) \} \), \( D = \{ A(a), B(a) \} \), \( \lambda_X(A(a)) = x \) and \( \lambda_X(B(a)) = y \). For \( \mathcal{P}^\text{an} \), \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}^\infty \times X, \lambda_X, A(a)) = x + xy + xy^2 + xy^3 + \ldots \).

3.2 Execution- and Tree-Based Semantics

We saw that when annotations are present there is more than one way to define a model-based semantics for Datalog and that it differs from the all-tree semantics. We now investigate definitions based on classical Datalog evaluation algorithms.

We extend the notion of immediate consequence operator describing the application of rules onto facts, with the computation of annotation. To this end, we introduce the annotation aware immediate consequence operator \( T_{\Sigma} \). Applying \( T_{\Sigma} \) on a set of annotated facts \( \{ I, \mathbb{K}, \lambda \} \) results in \( (T_{\Sigma}, I, \mathbb{K}, \lambda) \) where \( T_{\Sigma} \) is the result of applying the immediate consequence operator to \( \Sigma \) and \( I \), and \( \lambda_{T_{\Sigma}} \) annotates facts in \( T_{\Sigma} \) with the relational provenance (over \( \{ I, \mathbb{K}, \lambda \} \) of the UCQ formed by the bodies of the rules that create them. Formally, \( T_{\Sigma} := \{ H(\bar{a}) \mid I \models \exists \bar{y} \phi(\bar{a}, \bar{y}), \phi(\bar{x}, \bar{y}) \rightarrow H(\bar{x}) \in \Sigma \} \)

**Proposition 2.** The following holds:
\[ \mathcal{P}^\text{an} \subseteq \mathcal{P}^\text{at} \text{ and } \mathcal{P}^\text{an} \subseteq \mathcal{P}^\text{at} \].

Next examples show that \( \mathcal{P}^\text{an} \) and \( \mathcal{P}^\text{at} \) are incomparable.

**Example 6.** Let \( \Sigma = \{ A(x) \rightarrow \text{goal}, B(x) \rightarrow \text{goal} \} \), \( D = \{ A(a), B(a) \} \), \( \lambda_A(A(a)) = 2 \) and \( \lambda_B(B(a)) = 3 \). 

- Annotated models of \( \Sigma \) and \( D, \lambda_B \) are such that \( \mu^\text{at}(\text{goal}) \geq 3 \), so \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}, \lambda_B, \text{goal}) \).
- Set-annotated models of \( \Sigma \) and \( D, \lambda_B \) are such that \( \{ 2, 3 \} \subseteq \mu^\text{at}(\text{goal}) \), so \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}, \lambda_B, \text{goal}) = 5 \).

- Hence \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}, \lambda_B, \text{goal}) < \mathcal{P}^\text{at} (\Sigma, D, \mathbb{N}, \lambda_B, \text{goal}) \).

In the example above, we have \( \lambda_T \equiv \lambda \) and \( \lambda_T \equiv \lambda \) for every \( \lambda \).

### Example 7
Let \( \Sigma = \{ R(x, y) \rightarrow \text{goal} \} \), \( D = \{ R(a, b), R(a, c) \} \), \( \lambda_R(R(a, b)) = 2 \) and \( \lambda_R(R(a, c)) = 2 \).

- Annotated models of \( \Sigma \) and \( D, \lambda_R \) are such that \( \mu^\text{at}(\text{goal}) \geq 4 \), so \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}, \lambda_R, \text{goal}) \).
- Set-annotated models of \( \Sigma \) and \( D, \lambda_R \) are such that \( \{ 2 \} \subseteq \mu^\text{at}(\text{goal}) \), so \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}, \lambda_R, \text{goal}) = 2 \).

Hence \( \mathcal{P}^\text{an} (\Sigma, D, \mathbb{N}, \lambda_R, \text{goal}) > \mathcal{P}^\text{at} (\Sigma, D, \mathbb{N}, \lambda_R, \text{goal}) \).

Despite of their inherently different approaches, \( \mathcal{P}^\text{an} \), \( \mathcal{P}^\text{at} \) and \( \mathcal{P}^\text{at} \) coincide on a large class of semirings.
We define a union operator for annotated databases (over the same semiring): $(I, \mathbb{K}, \lambda) \cup (I', \mathbb{K}, \lambda') := (I \cup I', \mathbb{K}, \lambda'')$ where $\lambda''(\alpha) := \lambda(\alpha) + \lambda'(\alpha)$ where we slightly abuse notation by setting $\lambda(\alpha) = 0$ if $\alpha \notin I$, and $\lambda'(\alpha) = 0$ if $\alpha \notin I'$.

**Naive Evaluation / All Trees** In the naive evaluation algorithm, all rules are applied in parallel until a fixpoint is reached. The ‘annotation aware’ version of it is as follows:

We set $I_0^0(\Sigma, D, \mathbb{K}, \lambda) := (D, \mathbb{K}, \lambda)$, and define inductively $I_{n+1}^0(\Sigma, D, \mathbb{K}, \lambda) := T_\Sigma(I_n^0(\Sigma, D, \mathbb{K}, \lambda)) \cup (D, \mathbb{K}, \lambda)$. Note that the subscript $n$ of $I_n$ is an abbreviation for ‘naive’, and the superscript $i$ indicates how many times $T_\Sigma$ was applied.

Let $(I_n^0, \mathbb{K}, \lambda_n^i)$ denote $I_n^0(\Sigma, D, \mathbb{K}, \lambda)$. We say that $I_n^0(\Sigma, D, \mathbb{K}, \lambda)$ converges if there is some $k$ such that $I_n^k = I_n^{k+1}$ for every $\ell \geq k$, and sup$(\lambda_\ell^i(\alpha))$ exists for every $\alpha \in I_n^k$.

**Proposition 5.** For every $\Sigma, D, \mathbb{K}, \lambda$, if $\mathbb{K}$ is $\omega$-continuous then $I_n^\omega := I_n^k$ and $\lambda_n^\omega := \sup_{\ell \to \infty} \lambda_\ell^i$. The naive execution provenance semantics $\mathcal{P}^{\text{NE}}$ is defined by

$$\mathcal{P}^{\text{NE}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \begin{cases} \lambda_n^\omega(\alpha) & \alpha \in I_n^\omega \\ 0 & \text{otherwise} \end{cases}$$

and is equivalent to the all-tree semantics.

**Proposition 6.** It holds that $\mathcal{P}^{\text{NE}} = \mathcal{P}^{\text{AT}}$.

**Optimized Naive Evaluation / Minimal Depth Trees** We consider an optimized version of the naive algorithm that stops as soon as the desired fact is derived. We define the ‘annotation aware’ version of this algorithm by $I_{n+1}^0(\Sigma, D, \mathbb{K}, \lambda) := (D, \mathbb{K}, \lambda)$, and $I_{n+1}^0(\Sigma, D, \mathbb{K}, \lambda)$ :=

$$\begin{cases} T_\Sigma(I_n^0(\Sigma, D, \mathbb{K}, \lambda)) \cup (D, \mathbb{K}, \lambda) & \alpha \notin I_n^0 \\\n I_n^0(\Sigma, D, \mathbb{K}, \lambda) & \text{otherwise} \end{cases}$$

where $I_n^0$ is such that $I_n^0 := (I_n^0, \mathbb{K}, I_n^0)$.

**Proposition 7.** For every $\Sigma, D, \mathbb{K}, \lambda$, and $\alpha$ such that $\Sigma, D \models \alpha$, there exists $k \geq 0$ such that $I_n^k(\Sigma, D, \mathbb{K}, \lambda) = I_n^k(\Sigma, D, \mathbb{K}, \lambda)$ for every $\ell \geq k$.

With $k$ as provided by Proposition 7, we define the optimized execution provenance semantics $\mathcal{P}^{\text{OE}}$ by:

$$\mathcal{P}^{\text{OE}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \begin{cases} \lambda_n^k(\alpha) & \alpha \in I_n^k \\ 0 & \text{otherwise} \end{cases}$$

We show that an equivalent tree-based semantics can be obtained by considering only minimal depth trees for the desired fact. This approach has been considered useful, for example to present a ‘small proof’ for debugging (Zhao, Stabic, and Scholz 2020). Formally, let depth$(t)$ denote the depth of tree $t$. We say that $t \in T_\beta(\alpha)$ is of minimal depth if for every $t' \in T_\beta(\alpha)$ it holds that depth$(t) \leq$ depth$(t')$. The minimal depth tree provenance semantics $\mathcal{P}^{\text{MDT}}$ defined by

$$\mathcal{P}^{\text{MDT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \sum_{t \in T_\beta(\alpha)} \Lambda(t)$$

and is equivalent to the optimized naive execution.

**Proposition 8.** It holds that $\mathcal{P}^{\text{OE}} = \mathcal{P}^{\text{MDT}}$.

**Semenaive Evaluation / Hereditary Minimal Depth Trees** In the semenaive evaluation algorithm, facts are derived only once. We introduce a new consequence operator $\Delta_\Sigma$ that derives only new facts and is defined as follows: $\Delta_\Sigma(I, \mathbb{K}, \lambda) := (\Delta_\Sigma(I, \mathbb{K}, \lambda), \lambda_\Delta)$ where $\Delta_\Sigma := (I_{\Delta_\Sigma}, \mathbb{K}, \lambda_{\Delta_\Sigma})$, $I_{\Delta_\Sigma} := I_{\Delta_\Sigma} \setminus I$, and $\lambda_{\Delta_\Sigma}$ is the restriction of $\lambda_{\Delta_\Sigma}$ to $I_{\Delta_\Sigma}$.

We can now define the annotation aware version of the semenaive evaluation: $I_n^{\text{OE}}(\Sigma, D, \mathbb{K}, \lambda) := I_n^{\text{OE}}(\Sigma, D, \mathbb{K}, \lambda) \cup (\Delta_\Sigma(I_n^{\text{OE}}(\Sigma, D, \mathbb{K}, \lambda), \lambda_\Delta)).$

**Proposition 9.** For every $\Sigma, D, \mathbb{K}, \lambda$, there exists $k \geq 0$ such that $I_n^{\text{OE}}(\Sigma, D, \mathbb{K}, \lambda) = I_n^{\text{OE}}(\Sigma, D, \mathbb{K}, \lambda)$ for every $\ell \geq k$.

Note that, unlike in Proposition 5, we do not require $\mathbb{K}$ to be $\omega$-continuous. With $k$ provided by Proposition 9, the semenaive execution provenance semantics $\mathcal{P}^{\text{SNE}}$ is defined by

$$\mathcal{P}^{\text{SNE}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \begin{cases} \lambda_n^k(\alpha) & \alpha \in I_n^k \\ 0 & \text{otherwise} \end{cases}$$

To capture this with the tree-based approach we need to further restrict all subtrees to be of minimal depth. Formally, a derivation tree $t \in T_\beta(\alpha)$ is a hereditary minimal-depth (derivation) tree if for every node $n$ of $t$ labeled by $(\beta, r, h)$, the subtree $t_\beta$ with root $n$ is a minimal-depth derivation tree for $\beta$. The hereditary minimal depth tree provenance semantics $\mathcal{P}^{\text{HMDT}}$ is defined by

$$\mathcal{P}^{\text{HMDT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \sum_{t \in T_\beta(\alpha)} \Lambda(t)$$

and is equivalent to the semenaive execution.

**Proposition 10.** It holds that $\mathcal{P}^{\text{SNE}} = \mathcal{P}^{\text{HMDT}}$.

**3.3 Non-Recursive Tree-Based Semantics** Both execution-based semantics $\mathcal{P}^{\text{OE}}$ and $\mathcal{P}^{\text{SNE}}$ take into account finite subsets of derivation trees (and hence converge). Is there more informative tree-based semantics (i.e., one that takes into account a bigger subset of derivation trees) that still converges? We present such a semantics based on the intuition that deriving a fact from itself is redundant.

Formally, a non-recursive (derivation) tree is a derivation tree that does not contain two nodes labeled with the same fact and such that one is the descendant of the other. The non-recursive tree provenance semantics $\mathcal{P}^{\text{NRT}}$ is defined by

$$\mathcal{P}^{\text{NRT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \sum_{t \in T_\beta(\alpha)} \Lambda(t)$$

and is non-recursive.

**Connections between semantics** Next proposition follows from the fact that hereditary minimal-depth trees are of minimal-depth and non-recursive. The sets of minimal depth trees and non-recursive trees are incomparable, so that $\mathcal{P}^{\text{NRT}} \not\subseteq \mathcal{P}^{\text{MDT}}$ and $\mathcal{P}^{\text{MDT}} \not\subseteq \mathcal{P}^{\text{NRT}}$.

**Proposition 11.** The following hold:

$$\mathcal{P}^{\text{NRT}} \not\subseteq \mathcal{P}^{\text{MDT}} \text{ and } \mathcal{P}^{\text{MDT}} \not\subseteq \mathcal{P}^{\text{NRT}}$$

Moreover $\mathcal{P}^{\text{MDT}}$ and $\mathcal{P}^{\text{AT}}$ coincide on specific semirings.
Proposition 12. For every \( \Sigma, D, K, \lambda \) and \( \alpha \), if \( K \) is a commutative absorptive \( \omega \)-continuous semiring, then \( \mathcal{P}_{\text{RRT}}(\Sigma, D, K, \lambda, \alpha) = \mathcal{P}_{\text{NT}}(\Sigma, D, K, \lambda, \alpha) \).

If \( K \) is not absorptive, there exists \( \Sigma, (D, K, \lambda) \) and \( \alpha \) such that \( \mathcal{P}_{\text{RRT}}(\Sigma, D, K, \lambda, \alpha) \neq \mathcal{P}_{\text{NT}}(\Sigma, D, K, \lambda, \alpha) \), even in the case where \( \mathcal{P} \) is +-idempotent and \( \times \)-idempotent: Let \( \Sigma \) consist of the rule \( A(x) \land B(x) \rightarrow A(x) \) and \( D = \{ (A(b), B(a)) \} \). Then \( \mathcal{P}_{\text{RRT}}(\Sigma, D, K, \lambda, A(a)) = \lambda(A(a)) \) while \( \mathcal{P}_{\text{NT}}(\Sigma, D, K, \lambda, A(a)) = \lambda(A(a)) \times \lambda(B(a)) \).

The other semantics differ even under strong restrictions.

Example 8. This example shows that \( \mathcal{P}_{\text{RRT}}, \mathcal{P}_{\text{NRT}} \) and \( \mathcal{P}_{\text{HMDT}} \) differ even if \( K \) is + and \( \times \)-idempotent and absorptive.

Let \( \Sigma = \{ B(x) \land C(x) \rightarrow A(x), D(x) \rightarrow B(x), E(x) \rightarrow C(x), F(x) \rightarrow E(x) \} \)
\( D = \{ C(a), D(a), E(a), F(a) \} \)

The three derivation trees of \( A(a) \) w.r.t. \( \Sigma \) and \( D \) are non-recursi, but only the first two are of minimal depth and only the first one is a hereditary minimal-depth tree.

\[
\begin{array}{ccc}
A(a) & \quad & A(a) \\
/ & \quad & / \\
B(a) & \quad & B(a) \\
C(a) & \quad & C(a) \\
\mid & \quad & \mid \\
D(a) & \quad & E(a) \\
\end{array}
\]

Thus, if \( (D, K, \lambda) \) is such that \( \lambda(C(a)) = c, \lambda(D(a)) = d, \lambda(E(a)) = e, \) and \( \lambda(F(a)) = f \) then

\[
\begin{align*}
\mathcal{P}_{\text{RRT}}(\Sigma, D, K, \lambda, A(a)) &= c \times d + d \times e + d \times f \\
\mathcal{P}_{\text{NRT}}(\Sigma, D, K, \lambda, A(a)) &= c \times d + d \times e \\
\mathcal{P}_{\text{HMDT}}(\Sigma, D, K, \lambda, A(a)) &= c \times d
\end{align*}
\]

4 Basics Properties

In this section, we provide a framework allowing to compare the provenance semantics presented in the previous section. It is clear that they all fulfill the following definition.

Definition 3 (Provenance semantics). A provenance semantics is a partial function that assigns to a Datalog program \( \Sigma \), annotated database \((D, K, \lambda)\) and fact \( \alpha \), an element \( \mathcal{P}(\Sigma, D, K, \lambda, \alpha) \) in \( K \) such that:

1. \( \Sigma, D \models \alpha \) implies \( \mathcal{P}(\Sigma, D, K, \lambda, \alpha) = 0_K \).
2. If \( K \) is positive, \( \mathcal{P}(\Sigma, D, K, \lambda, \alpha) = 0_K \) implies \( \Sigma, D \not\models \alpha \).

We call the semiring domain of \( \mathcal{P} \) the maximal set \( S \) of semirings such that \( \mathcal{P}(\Sigma, D, K, \lambda, \alpha) \) is defined for every \( K \in S \), and every \( \Sigma, (D, K, \lambda) \) and \( \alpha \).

Intuitively, Definition 3 means that the semantics reflects fact (non)-entailment. It is extremely permissive: We could define such a semantics that associates to each entailed fact a random semiring element different from zero, and does not bring any information beyond facts entailment. In the sequel, we state and discuss a number of properties that may be expected to be satisfied by a provenance semantics.

Throughout this section, when not stated otherwise, \( \mathcal{P}, \Sigma, D, K, \lambda \) and \( \alpha \) denote respectively an arbitrary provenance semantics, Datalog program, database, commutative semiring \((K, +, \times, 0, 1)\), function from \( D \) to \( K \setminus \{0\} \), and fact. We phrase properties as conditions, and say that \( \mathcal{P} \) satisfies a property if it satisfies the condition. We also denote by \( \lambda_X \) an injective function \( \lambda_X : D \rightarrow X \).

4.1 Compatibility with Classical Notions

Property 1 is a sanity check: if a Datalog program amounts to a UCQ, the provenance should be the same as the one defined for relational databases (Green, Karvounarakis, and Tannen 2007). A Datalog program \( \Sigma \) is UCQ-defined if its rules are of the form \( \phi(\vec{x}, \vec{y}) \rightarrow H(\vec{x}) \) where \( H \) is a predicate that does not occur in any rule. In this case, the equivalent UCQ \( Q^\Sigma \) of \( \Sigma \) is \( \bigcup_{\phi(\vec{x}, \vec{y}) \rightarrow H(\vec{x}) \in \Sigma} \exists \vec{y} \phi(\vec{x}, \vec{y}) \).

Property 1 (Algebra Consistency). If \( \Sigma \) is UCQ-defined with rule head \( H(\vec{x}) \) and \( H \notin S(D) \), then for every tuple \( \vec{a} \) of same arity as \( \vec{x} \), the relational provenance of \( Q^\Sigma(\vec{a}) \) is equal to \( \mathcal{P}(\Sigma, D, K, \lambda, H(\vec{a})) \).

While Property 1 considers the behavior of a provenance semantics on a restricted class of queries, we can alternatively consider its behavior on a specific semiring. Boolean provenance has a very natural definition, based on the database subsets that entail the query, and is widely used, notably for probabilistic databases (Senellart 2017), but also for ontology-mediated query explanation (e.g., in Datalog**/- or description logics (Ceylan et al. 2019; Ceylan et al. 2020)). It is formalized with the semiring \( PosBool(X) \).

Property 2 (Boolean Compatibility).

\[
\mathcal{P}(\Sigma, D, PosBool(X), \lambda_X, \alpha) = \bigcup_{\vec{d} \in D, \beta \in D'} \lambda_X(\beta)
\]

Property 2 expresses ‘insensibility’ to syntax, that is, every provenance semantics that satisfies Property 2 agrees on equivalent programs (i.e., those that have the same models) for the semiring \( PosBool(X) \). This is related to ideas from (Green 2009) on the provenance of equivalent UCQs.

4.2 Compatibility with Specialization

Semiring provenance has been introduced to abstract from the particular semiring at hand, and factor the computations in some provenance semiring which specializes correctly to any semiring of interest. The next property allows one to do so, and is thus highly desirable (Bourgaux et al. 2022).

Property 3 (Commutation with Homomorphisms). If there is a semiring homomorphism \( h \) from \( K_1 \) to \( K_2 \), then \( h(\mathcal{P}(\Sigma, D, K_1, \lambda, \alpha)) = \mathcal{P}(\Sigma, D, K_2, h \circ \lambda, \alpha) \).

We call Property 3 restricted to \( \omega \)-continuous homomorphisms Commutation with \( \omega \)-Continuous Homomorphisms.

Specializing correctly is all the more useful when \( \mathcal{P} \) is well-defined for a lot of semirings, in particular on all commutative or at least all commutative \( \omega \)-continuous semirings.

Property 4 (Any \( \omega \)-Continuous Semiring). \( \mathcal{P} \) satisfies the Any Semiring Property (resp. Any \( \omega \)-Continuous Semiring Property) if the semiring domain of \( \mathcal{P} \) contains the set of all commutative (resp. commutative \( \omega \)-continuous) semirings.
4.3 Joint and Alternative Use of the Data

How is the actual usage of the data reflected in the provenance semantics? The next property formalizes that multiplication reflects joint use of the data, and addition alternative use. For the rest of this section, we set goal to be a nullary predicate not in \( \mathcal{S}(\Sigma) \cup \mathcal{S}(D) \).

**Property 5 (Joint and Alternative Use).** For all tuples of facts \((\alpha_1, \ldots, \alpha_n), \ldots, (\alpha^n_1, \ldots, \alpha^n_m)\), it holds that

\[
\mathcal{P}(\Sigma' \cup \{\lambda\}) = \prod_{i=1}^{m} \mathcal{P}(\alpha_i)
\]

**Property 6 (Joint Use).** For all facts \(\alpha_1, \ldots, \alpha_m\),

\[
\mathcal{P}(\Sigma', D, \mathcal{K}, \lambda, \gamma) = \prod_{j=1}^{n} \mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \gamma)
\]

**Property 7 (Alternative Use).** For all facts \(\alpha_1, \ldots, \alpha_m\),

\[
\mathcal{P}(\Sigma', D, \mathcal{K}, \lambda, \gamma) = \sum_{i=1}^{m} \mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \gamma)
\]

4.4 Fact Roles in Entailment

After considering how facts can be combined or used alternatively to entail a result, we ponder their possible roles w.r.t. the entailment. Property 8 asserts that the original annotation of a fact takes part in the provenance of its entailment.

**Property 8 (Self).** If \(\alpha \in D\), then \(\lambda(\alpha) \subseteq \mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \alpha)\).

Moreover, if a database fact cannot be alternatively derived using the rules, then its provenance should be exactly its original annotation. To phrase this property we use the grounding \(\Sigma_D\) of \(\Sigma\) w.r.t. \(D\), defined by \(\Sigma_D = \{h(\phi(\vec{x}, \vec{y})) \mapsto h(\vec{H})(\vec{x}) | \phi(\vec{x}, \vec{y}) \mapsto H(\vec{x}) \in \Sigma, h : \vec{x} \cup \vec{y} \mapsto D(D)\}\). It holds that \(\Sigma, D \models \alpha\) if and only if \(\Sigma_D, D \models \alpha\).

**Property 9 (Parsimony).** If \(\alpha \in D\) does not occur in any rule head in \(\Sigma_D\) then \(\mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \alpha) = \lambda(\alpha)\).

Parsimony Property together with other constraints guarantee Algebra Consistency Property.

**Proposition 13.** If \(\mathcal{P}\) satisfies Properties 5 (Joint and Alternative Use) and 9 (Parsimony), and is such that for every \(\Sigma, D, \mathcal{K}, \lambda, \alpha, \mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \alpha) = \mathcal{P}(\Sigma_D, D, \mathcal{K}, \lambda, \alpha)\), then it satisfies Property 1 (Algebra Consistency).

Property 10 states that \(\mathcal{P}\) reflects the necessity of a fact for the entailment. We say that \(\beta \in D\) is necessary to \(\Sigma, D \models \alpha\) if \(\Sigma, D \setminus \{\beta\} \not\models \alpha\), and denote by \(\mathcal{Nec}\) the set of such facts.

**Property 10 (Necessary Facts).** There exists \(e \in \mathcal{K}\) such that \(\mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \alpha) = \Pi_{\beta \in \mathcal{Nec}} \lambda(\beta) \times e\).

A fact is usable to \(\Sigma, D \models \alpha\) if it occurs in some derivation tree in \(\mathcal{T}_D^\Sigma(\alpha)\). Usable facts are related to the notion of lineage (Cui, Widom, and Wiener 2000) and can be defined without resorting to derivation trees (Bourgaux et al. 2022). Intuitively, if a fact is not usable to derive another fact, it should not have any influence on its provenance.

**Property 11 (Non-Usable Facts).** For every \(X\) that differs from \(\lambda\) only on facts that are not usable to \(\Sigma, D \models \alpha\), it holds that \(\mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \alpha) = \mathcal{P}(\Sigma, D, \mathcal{K}, X, \alpha)\).

4.5 Data Modification

The last two properties indicate how provenance is impacted when facts are inserted or deleted.

**Property 12 (Insertion).** For every \((D', \mathcal{K}, \lambda')\) such that \(D \cap D' = \emptyset\),

\[
\mathcal{P}(\Sigma, D, \mathcal{K}, \lambda, \alpha) = \mathcal{P}(\Sigma, D', \mathcal{K}, \lambda', \alpha)
\]

**Property 13 (Deletion).** For every provenance semiring \(\mathcal{P}(\Sigma, D, \mathcal{K}, \alpha)\) and \(\mathcal{P}(\Delta, D', \mathcal{K}, \lambda', \alpha)\) obtained by setting the annotations of facts in \(\Delta\) to \(0\),

\[
\mathcal{P}(\Sigma, D, \mathcal{K}, \alpha) \{\lambda_X(x) = 0\}_{x \in \Delta}
\]

5 Semantics Analysis w.r.t. Properties

In this section, we analyze the semantics proposed in Section 3 w.r.t. the properties introduced in Section 4. The properties each semantics satisfies are summarized in Table 1. Proofs of the positive cases are given in the appendix of (Bourgaux et al. 2022) and we discuss the negative cases, which may be more characteristic, in the sequel.

<table>
<thead>
<tr>
<th>Property</th>
<th>PAT</th>
<th>PHRT</th>
<th>PMDT</th>
<th>PDHT</th>
<th>PAM</th>
<th>PSEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebra Consistency</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Boolean Comp.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Com. with Hom.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Com. with ω-Cont.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Any Semiring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Any ω-Cont. Sem.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Joint and Alt. Use</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Joint Use</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Alternative Use</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Self</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Parsimony</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Necessary Facts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Non-Usable Facts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Insertion</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Deletion</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1: Does a property hold for a provenance semantics?

5.1 Tree- and Execution-Based Semantics Cases

We first discuss \(\mathcal{P}_{MDT}\) and \(\mathcal{P}_{DHT}\), which have not been much investigated and stand out compared to \(\mathcal{P}_{AT}\) and \(\mathcal{P}_{PHRT}\). The next example shows that they do not satisfy the Boolean Compatibility, Joint and Alternative Use, Alternative Use, Insertion and Deletion Properties.
Example 9. Consider $\Sigma$ and $(D, Prov(X), \lambda_X)$ as follows.

$\Sigma = \{ A(x) \rightarrow \text{goal}, B(x) \rightarrow \text{goal}, C(x) \rightarrow B(x) \}$

$D = \{ A(a), C(a) \}$ with $\lambda_X(A(a)) = a$, $\lambda_X(C(a)) = c$

It holds that both $\mathcal{P}^{\text{HNDT}}(\Sigma, D, Prov(X), \lambda_X, \text{goal})$ and $\mathcal{P}^{\text{HDT}}(\Sigma, D, Prov(X), \lambda_X, \text{goal})$ are equal to $a$. For $P \in \{ \mathcal{P}^{\text{HDT}}, \mathcal{P}^{\text{HNDT}} \}$ we then have the following:

(i) The Boolean provenance of goal is $a \lor c$, hence $P$ does not satisfy the Boolean Compatibility Property.

(ii) Since $\mathcal{P}(0, D, Prov(X), \lambda_X, A(a)) = a$ and $\mathcal{P}(0, D, Prov(X), \lambda_X, C(a)) = c$, $P$ does not satisfy the Alternative Use, nor the Joint and Alternative Use Property.

(iii) Let $D' = \{ \{ A(a) \} \}$ and $\lambda'(goal) = y$. It holds that $\mathcal{P}(\Sigma, D \cup D', Prov(X), \lambda_X \cup \lambda'_X, \text{goal}) = g$, which is different from $\mathcal{P}(\Sigma, D, Prov(X), \lambda_X, \text{goal}) + \mathcal{P}(\Sigma, D', Prov(X), \lambda'_X, \text{goal}) + e$ for every $e \in Prov(X)$. Hence $P$ does not satisfy the Insertion Property.

Example 10. Let $\Sigma = \{ C(x) \rightarrow B(x), D(x) \rightarrow A(x) \}$, $D = \{ B(a), C(a), D(a) \}$ and $\lambda(B(a)) = b$, $\lambda(C(a)) = c$, $\lambda(D(a)) = d$, and consider $\Sigma' = \Sigma \cup \{ A(a) \land B(a) \rightarrow \text{goal} \}$. $\mathcal{P}^{\text{HDT}}(\Sigma, D, \lambda, \text{goal}) = b \times d + c \times d$ while $\mathcal{P}^{\text{HDT}}(\Sigma, D, \lambda, \text{goal}) = d$ and $\mathcal{P}^{\text{HDT}}(\Sigma, D, \lambda, B(a)) = b$. Hence $\mathcal{P}^{\text{HDT}}$ does not satisfy the Joint Use Property.

We conclude this discussion with the remark that $\mathcal{P}^{\text{AT}}$ satisfies the Commutation with $\omega$-Continuous Homomorphisms but not the Commutation with Homomorphisms Property.

Example 11. Consider the semiring $\mathbb{N}^\infty$ with the classical operations, and define $\mathbb{N}^{\infty \times}$ as its extension by an element $\infty'$ such that for every $n \in \mathbb{N} \cup \{ \infty \}$, $n + \infty' = \infty'$, and $n \times \infty' = \infty'$ if $n \neq 0$, and $0$ otherwise.

Both semirings are $\omega$-continuous and $h : \mathbb{N}^\infty \rightarrow \mathbb{N}^{\infty \times}$ defined by $h(n) = n$ for every $n \in \mathbb{N}$, $h(\infty) = \infty'$ is a semiring homomorphism (which is not $\omega$-continuous).

Assume that $\mathcal{P}^{\text{AT}}(\Sigma, D, \infty, \lambda, \text{goal}) = \Sigma_{i \in \mathbb{N}} 1 = \infty$. Then $h(\mathcal{P}^{\text{AT}}(\Sigma, D, \infty, \lambda, \text{goal})) = \infty'$ is different from $\mathcal{P}^{\text{AT}}(\Sigma, D, \mathbb{N}^{\infty \times}, h \circ \lambda, \text{goal}) = \Sigma_{i \in \mathbb{N}} h(1) = \infty$.

5.2 Model-Based Semantics Cases

On $+$-idempotent semirings, $\mathcal{P}^{\text{AM}}$ and $\mathcal{P}^{\text{SAM}}$ coincide with $\mathcal{P}^{\text{AT}}$ so verify the same properties, and the semiring $\mathbb{R}[X]$ of formal power series with Boolean coefficients can be used to compute them in any $+$-idempotent semiring (Bourgaux et al. 2022). However, on non-idempotent semirings, they do not satisfy several properties, and in particular the Commutation with ($\omega$-Continuous) Homomorphisms Properties.

Example 12. Let $\Sigma = \{ A(x) \rightarrow \text{goal}, B(x) \rightarrow \text{goal} \}$ and $D = \{ A(a), B(a) \}$ and consider the provenance semiring $\mathbb{N}^\infty[X]$ with $\lambda_X(A(a)) = x$ and $\lambda_X(B(a)) = y$.

It holds that both $\mathcal{P}^{\text{AT}}(\Sigma, D, \mathbb{N}^\infty[X], \lambda_X, \text{goal})$ and $\mathcal{P}^{\text{AM}}(\Sigma, D, \mathbb{N}^\infty[X], \lambda_X, \text{goal})$ are equal to $x + y$.

Example 13. Let $\Sigma = \{ A(x) \rightarrow g_1, A(x) \rightarrow g_2, B(x) \rightarrow g_1, B(x) \rightarrow g_2 \}$

$D = \{ A(a), B(a) \}$ with $\lambda_X(A(a)) = x$, $\lambda_X(B(a)) = y$.

Both $\mathcal{P}^{\text{AM}}(\Sigma, D, \mathbb{N}^\infty[X], \lambda_X, g_1)$ and $\mathcal{P}^{\text{SAM}}(\Sigma, D, \mathbb{N}^\infty[X], \lambda_X, g_2)$ are equal to $x + y$ but non-$\mathcal{P}^{\text{SAM}}(\Sigma \cup \{ g_1 \land g_2 \rightarrow \text{goal} \}, D, \mathbb{N}^\infty[X], \lambda_X, \text{goal}) = x^2 + y^2 + xy \neq (x+y)^2$.

We show that $\mathcal{P}^{\text{AT}}$ does not satisfy the Necessary Facts Property in the appendix of (Bourgaux et al. 2022) because we needed to craft a specific semiring to get a counter-example.

6 Complexity Considerations and Conclusion

In this paper, we present alternative provenance semantics for Datalog based on models, execution algorithms and derivation trees, and compare them through the lens of different properties. $\mathcal{P}^{\text{HDT}}$ is the only one that satisfies all the studied properties but does not coincide with an execution based semantics contrary to the other tree-based semantics $\mathcal{P}^{\text{AT}}, \mathcal{P}^{\text{HDT}}, \text{and } \mathcal{P}^{\text{HNDT}}$. The equivalence between the tree-based $\mathcal{P}^{\text{AT}}, \mathcal{P}^{\text{RT}}$, and model-based $\mathcal{P}^{\text{AM}}$ and $\mathcal{P}^{\text{SAM}}$ definitions on absorptive semirings may also indicates a robust provenance on this restricted setting.

One of the main complexity sources of Datalog provenance stems from its infinite representation. Deucht et al. (2014) studied semirings for which the provenance expressions given by $\mathcal{P}^{\text{AT}}$ are finite, and showed that they can be represented by polynomial size circuits. We show that the annotations produced at each iteration of our execution algorithms can be represented by arithmetic circuits of polynomial size in the data (Bourgaux et al. 2022). Consequently, both $\mathcal{P}^{\text{HDT}}$ and $\mathcal{P}^{\text{HNDT}}$ can be represented by polynomial size circuits regardless of the semiring. On the contrary, we show that (assuming $P \neq \text{NP}$) there is no polynomially computable circuit that computes $\mathcal{P}^{\text{RT}}$ on $\mathbb{N}^\infty[X]$, by a reduction from a result by Arenas, Conca, and Pérez (2012). Whether it is possible to polynomially compute circuits for $\mathcal{P}^{\text{HDT}}$ on provenance semirings less expressive than $\mathbb{N}^\infty[X]$ but non-absorptive remains open.
Acknowledgements
This work is supported by the ANR project CQFD (ANR-18-CE23-0003).

References


