Revisiting Semiring Provenance for Datalog

Camille Bourgaux[1] , **Pierre Bourhis**[2] , **Liat Peterfreund**[3] , **Michaël Thomazo**[1]
[1]DI ENS, ENS, CNRS, PSL University & Inria, Paris, France
[2]CRIStAL, CNRS, University of Lille, Inria, Lille, France
[3]LIGM, 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; Deutch et al. 2014; Deutch, 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 (Achs 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 alternative 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 $\mathbf{P}$, $\mathbf{C}$, and $\mathbf{V}$ be mutually disjoint, possibly infinite sets of *predicates*, *constants*, and *variables* respectively. Elements of $\mathbf{C} \cup \mathbf{V}$ are called *terms*. An *atom* has the form $p(t_1, \dots, t_n)$ where $p \in \mathbf{P}$ 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: $\forall \vec{x} \forall \vec{y} (\phi(\vec{x}, \vec{y}) \rightarrow \psi(\vec{x}))$ where $\vec{x}$ and $\vec{y}$ are tuples of variables and $\phi(\vec{x}, \vec{y})$ and $\psi(\vec{x})$ are conjunctions of atoms whose variables are $\vec{x} \cup \vec{y}$ and $\vec{x}$ respectively. We call $\phi(\vec{x}, \vec{y})$ and $\psi(\vec{x})$ 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 $H(\vec{x})$, and quantifiers are implicit. The *domain* $\mathcal{D}(\mathcal{A})$ of a set $\mathcal{A}$ 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 $\mathcal{S}(D)$ (resp. $\mathcal{S}(\Sigma)$) 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 $\mathcal{A}$ of atoms to a set $\mathcal{B}$ of atoms is a function $h : \mathcal{D}(\mathcal{A}) \rightarrow \mathcal{D}(\mathcal{B})$ such that $h(t) = t$ for all $t \in \mathbf{C}$, and $p(t_1, \dots, t_n) \in \mathcal{A}$ implies $h(p(t_1, \cdots, t_n)) := p(h(t_1), \dots, h(t_n)) \in \mathcal{B}$. We denote by $h(\mathcal{A})$ the set $\{h(p(t_1, \dots, t_n)) \mid p(t_1, \dots, t_n) \in \mathcal{A}\}$. 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 $r := \phi(\vec{x}, \vec{y}) \rightarrow \psi(\vec{x})$, denoted by $I \models r$, if every homomorphism h from $\phi(\vec{x}, \vec{y})$ to I is also a homomorphism from $\psi(\vec{x})$ to I; it is a *model* of a Datalog program Σ if $I \models r$ for every $r \in \Sigma$; it is a *model* of a database D if $D \subseteq I$. A fact α is *entailed* by D and Σ, denoted $\Sigma, D \models \alpha$, if $\alpha \in I$ for every model I of Σ and D.

Example 1. *Let Σ contain the rules $B(x) \rightarrow A(x)$, $R(x, y) \wedge A(y) \rightarrow B(x)$, and $R(x, y) \rightarrow R(y, x)$, 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 Σ, D.*

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 $\alpha \in D$, or α such that there exists a rule $r := \phi(\vec{x}, \vec{y}) \rightarrow \psi(\vec{x})$ and a homomorphism h from $\phi(\vec{x}, \vec{y})$ to D such that $h(\psi(\vec{x})) = \alpha$.

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 $(p(t_1, \dots, t_m), r, h)$ where*

- *$p(t_1, \dots, t_m)$ is a fact over the schema $\mathcal{S}(\Sigma)$;*
- *r is a rule from Σ of the form $\phi(\vec{x}, \vec{y}) \rightarrow p(\vec{x})$;*
- *h is a homomorphism from $\phi(\vec{x}, \vec{y})$ to the facts of the labels of the node children, such that $h(p(\vec{x})) = p(t_1, \dots, t_m)$;*
- *there is a bijection f between the node children and the atoms of $\phi(\vec{x}, \vec{y})$, such that for every $q(\vec{z}) \in \phi(\vec{x}, \vec{y})$, $f(q(\vec{z}))$ is of the form $(h(q(\vec{z})), r', h')$ or is a leaf labeled by $h(q(\vec{z}))$.*

Moreover, if $(p(t_1, \cdots, t_m), r, h)$ or $p(t_1, \cdots, t_m)$ is the root of t, then $p(t_1, \cdots, t_m) = \alpha$.

Example 2. *Let Σ contain $r_1 := R(x, y) \rightarrow H(x, x)$, $r_2 := R(x, y) \rightarrow H(x, y)$ and $r_3 := S(x, y, z) \wedge S(x, z, y) \rightarrow H(x, x)$. If $D = \{R(a, a), S(a, b, c), S(a, c, b)\}$, then the fact $\alpha := H(a, a)$ has the following derivation trees*

$$\begin{array}{c} (\alpha, r_1, h) \\ | \\ R(a,a) \end{array} \quad \begin{array}{c} (\alpha, r_2, h) \\ | \\ R(a,a) \end{array} \quad \begin{array}{c} (\alpha, r_3, h_3) \\ \diagup \quad \diagdown \\ S(a,b,c)\,S(a,c,b) \end{array} \quad \begin{array}{c} (\alpha, r_3, h'_3) \\ \diagup \quad \diagdown \\ S(a,c,b)\,S(a,b,c) \end{array}$$

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 when there is no ambiguity.

[1]Note that we do not require the set of predicates of atoms appearing in heads of rules to be disjoint from $\mathcal{S}(D)$; naturally, all of our results are valid under this assumption as well.

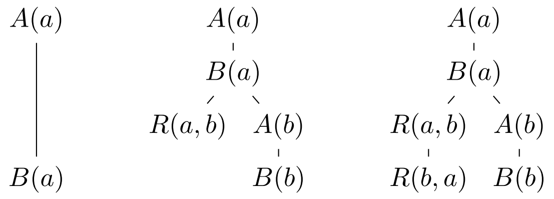


Figure 1: Some derivation trees of $A(a)$ in Example 1.

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 Σ and database D, written $\Sigma, D \models q$, if and only if $I \models q$ for every model I of Σ and D. Note that $\Sigma, D \models q$ if and only if $\Sigma \cup \{\phi(\vec{y}) \rightarrow \mathsf{goal}\}, D \models \mathsf{goal}$, where goal is a nullary predicate such that $\mathsf{goal} \notin \mathcal{S}(\Sigma) \cup \mathcal{S}(D)$. A tuple of constants $\vec{a}$ is an *answer* to a CQ $q(\vec{x}) := \exists \vec{y}\, \phi(\vec{x}, \vec{y})$ over Σ and D if $\vec{a}$ and $\vec{x}$ have the same arity and $\Sigma, D \models q(\vec{a})$ where $q(\vec{a})$ 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 $q(\vec{a})$ 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* $\mathbb{K} = (K, +_{\mathbb{K}}, \times_{\mathbb{K}}, 0_{\mathbb{K}}, 1_{\mathbb{K}})$ is a set K with distinguished elements $0_{\mathbb{K}}$ and $1_{\mathbb{K}}$, equipped with two binary operators: $+_{\mathbb{K}}$, called the *addition*, which is an associative and commutative operator with identity $0_{\mathbb{K}}$, and $\times_{\mathbb{K}}$, called the *multiplication*, which is an associative operator with identity $1_{\mathbb{K}}$. It also holds that $\times_{\mathbb{K}}$ distributes over $+_{\mathbb{K}}$, and $0_{\mathbb{K}}$ is annihilating for $\times_{\mathbb{K}}$. 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, $\mathbb{K} = (K, +_{\mathbb{K}}, \times_{\mathbb{K}}, 0_{\mathbb{K}}, 1_{\mathbb{K}})$ is a semiring, and $\lambda : D \mapsto K \setminus \{0_{\mathbb{K}}\}$ maps facts into semiring elements different from $0_{\mathbb{K}}$.*

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 $\mathbb{T} = (\mathbb{R}_{+}^{\infty}, \mathsf{min}, +, \infty, 0)$ is used to compute minimal-cost paths. We define $\lambda_{\mathbb{N}} : D \mapsto \mathbb{N} \setminus \{0\}$ by $\lambda_{\mathbb{N}}(B(a)) = 3$, $\lambda_{\mathbb{N}}(B(b)) = 1$, $\lambda_{\mathbb{N}}(R(a, b)) = 2$, $\lambda_{\mathbb{N}}(R(b, a)) = 1$; And $\lambda_{\mathbb{T}} : D \mapsto \mathbb{R}_{+}$ by $\lambda_{\mathbb{T}}(B(a)) = 10$, $\lambda_{\mathbb{T}}(B(b)) = 1$, $\lambda_{\mathbb{T}}(R(a, b)) = 5$, $\lambda_{\mathbb{T}}(R(b, a)) = 2$.*

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 $a, b \in K$, $a \times b = 0$ 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 ω-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 *ω-continuous* if $\sqsubseteq$ is a partial order, every (infinite) ω-chain $a_0 \sqsubseteq a_1 \sqsubseteq a_2 \dots$ 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 $\exists xy\, (R(x, y) \wedge B(y))$ 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 $(D, \mathbb{T}, \lambda_{\mathbb{T}})$ with minimal cost $\mathsf{min}(\lambda_{\mathbb{T}}(R(a, b)) + \lambda_{\mathbb{T}}(B(b)), \lambda_{\mathbb{T}}(R(b, a)) + \lambda_{\mathbb{T}}(B(a))) = 6$.*

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 α entailed by Σ and D the following sum, where $T_D^{\Sigma}(\alpha)$ is the set of all derivation trees for α w.r.t. Σ and D and $\Lambda(t) := \prod_{v \text{ is a leaf of } t} \lambda(v)$ is the $\mathbb{K}$-annotation of the derivation tree t (since $\mathbb{K}$ is commutative, the result of the product is well-defined).

$$\mathcal{P}^{\mathrm{AT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \sum_{t \in T_D^{\Sigma}(\alpha)} \Lambda(t).$$

Since $T_D^{\Sigma}(\alpha)$ may be infinite, $\mathcal{P}^{\mathrm{AT}}$ is well-defined for all Σ, $(D, \mathbb{K}, \lambda)$ and α only in the case where $\mathbb{K}$ is ω-continuous.

Example 5 (Ex. 3 cont'd)**.** *The fact $\alpha := A(a)$ is entailed with minimal cost: $\mathcal{P}^{AT}(\Sigma, D, \mathbb{T}, \lambda_{\mathbb{T}}, \alpha) = \mathsf{min}_{t \in T_D^{\Sigma}(\alpha)} \Sigma_{v \text{ is a leaf of } t} \lambda_{\mathbb{T}}(v) = 3$. Since $\mathbb{N}$ is not ω-continuous, $\mathcal{P}^{AT}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \alpha)$ 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* $\mathit{Prov}(X)$ 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 $PosBool(X) := (PosBool(X), \vee, \wedge, \mathsf{false}, \mathsf{true})$ 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 $S = \Sigma_{m \in \mathsf{mon}(X)} S(m)m$ where $\mathsf{mon}(X)$ 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 $\mathbb{N}^\infty[\![X]\!]$ of formal power series with coefficients from $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$.

A *semiring homomorphism* from $\mathbb{K} = (K, +_\mathbb{K}, \times_\mathbb{K}, 0_\mathbb{K}, 1_\mathbb{K})$ to $\mathbb{K}' = (K', +_{\mathbb{K}'}, \times_{\mathbb{K}'}, 0_{\mathbb{K}'}, 1_{\mathbb{K}'})$ is a mapping $h : K \to K'$ such that $h(0_\mathbb{K}) = 0_{\mathbb{K}'}$, $h(1_\mathbb{K}) = 1_{\mathbb{K}'}$, and for all $a, b \in K$, $h(a +_\mathbb{K} b) = h(a) +_{\mathbb{K}'} h(b)$ and $h(a \times_\mathbb{K} b) = h(a) \times_{\mathbb{K}'} h(b)$. A semiring homomorphism between ω-continuous semirings is *ω-continuous* if it preserves least upper bounds: $h(\sup((a_i)_{i \in \mathbb{N}})) = \sup((h(a_i))_{i \in \mathbb{N}})$.

Following Deutch et al. (2014), we say that a provenance semiring $Prov(X)$ *specializes* correctly to a semiring $\mathbb{K}$, if any valuation $\nu : X \to K$ extends uniquely to a (ω-continuous if $Prov(X)$ and $\mathbb{K}$ are ω-continuous) semiring homomorphism $h : Prov(X) \to K$, allowing the computations for $\mathbb{K}$ 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 $\mathbb{N}[X]$ is universal for commutative semirings, and $\mathbb{N}^\infty[\![X]\!]$ is universal for commutative ω-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 Σ, annotated database $(D, \mathbb{K}, \lambda)$, and fact α, a semiring element $\mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha)$.

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, μ^I) where I is a set of facts and μ^I is a function that annotates facts of I, and formulate requirements for them to be models of Σ and $(D, \mathbb{K}, \lambda)$, extending standard models of Σ 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 incognizant semantics uses the maximum-based union, while the 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 Σ and a source I, a target J is an *incognizant solution* for I w.r.t. Σ if (I, J) satisfies every s-t tgd in Σ. It is a *cognizant solution* if for every $r \in \Sigma$, there is a target instance J_r such that (I, J_r) satisfies r and $\uplus J_r \subseteq J$, where $\uplus$ 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) *certain answers* to a query q w.r.t. Σ on I are defined using bag intersection of the answers over the incognizant (resp. cognizant) solutions for I w.r.t. Σ, 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 $D = \{(B(a), 1), (C(a), 1)\}$. Under the incognizant semantics, the multiplicity of the certain answer of the Boolean query $A(a)$ w.r.t. Σ 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. Σ 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 $\mathcal{P}^{\mathrm{AT}}$ on the counting semiring $\mathbb{N} = (\mathbb{N}, +, \times, 0, 1)$, 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 greatest lower bounds, it is defined on a restricted class of semirings.

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

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

The *annotated model-based provenance semantics* $\mathcal{P}^{\mathrm{AM}}$ is defined by

$$\mathcal{P}^{\text{AM}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \inf(\{\mu^I(\alpha) | (I, \mu^I) \models (\Sigma, D, \mathbb{K}, \lambda)\}).$$

Proposition 1. *If the Datalog rules in Σ are (1) s-t tgds, or (2) formulated in DL-Lite$_R$, then for every BCQ q, $\mathcal{P}^{AM}(\Sigma \cup \{q \rightarrow \mathsf{goal}\}, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal})$ is equal to the multiplicity of the empty tuple in (1) the incognizant certain answers or (2) the bag certain answers to q w.r.t. Σ and $(D, \mathbb{N}, \lambda_{\mathbb{N}})$.*

Set-Annotated Model-based We adapt the work on provenance for the description logics DL-Lite$_R$ and $\mathcal{ELH}^r$ (Calvanese et al. 2019; Bourgaux et al. 2020), where the semiring is assumed to be a $\times$-idempotent provenance semiring $Prov(X)$ and rules are also annotated. Annotated models of annotated knowledge bases are defined as set of facts annotated with sets of monomials from $Prov(X)$. Given a fact α and a monomial m over X, $(\Sigma, D, Prov(X), \lambda_X) \models (\alpha, m)$ holds when m belongs to the annotation set of α in every models of Σ and $(D, Prov(X), \lambda_X)$.

To obtain an analog provenance semantics for Datalog, we define interpretations which associate facts with (possibly infinite) *sets of annotations*, and formulate the requirements for them to be models of Σ and $(D, \mathbb{K}, \lambda)$.

Let $\mathbb{K} = (K, +, \times, 0, 1)$ be a commutative ω-continuous semiring, Σ be a Datalog program, and $(D, \mathbb{K}, \lambda)$ be an annotated database. We define $\mathbb{K}$-*set-annotated interpretations* as pairs (I, μ^I) where I is a set of facts, and μ^I is a function from I to the *power-set of* K. We say that a $\mathbb{K}$-set-annotated interpretation (I, μ^I) is a model of Σ and $(D, \mathbb{K}, \lambda)$, denoted by $(I, \mu^I) \models (\Sigma, D, \mathbb{K}, \lambda)$, if

1. $D \subseteq I$, and for every $\alpha \in D$, $\lambda(\alpha) \in \mu^I(\alpha)$;
2. for every $\phi(\vec{x}, \vec{y}) \rightarrow H(\vec{x})$ in Σ, whenever there is a homomorphism $h : \phi(\vec{x}, \vec{y}) \mapsto I$, then $h(H(\vec{x})) \in I$ and if $h(\phi(\vec{x}, \vec{y})) = \beta_1 \wedge \cdots \wedge \beta_n$, $\{\Pi_{i=1}^n k_i \mid (k_1, \ldots, k_n) \in \mu^I(\beta_1) \times \cdots \times \mu^I(\beta_n)\} \subseteq \mu^I(h(H(\vec{x})))$.

The *set-annotated model-based provenance semantics* $\mathcal{P}^{\text{SAM}}$ is defined by

$$\mathcal{P}^{\text{SAM}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \sum_{k \in \bigcap_{(I, \mu^I) \models (\Sigma, D, \mathbb{K}, \lambda)} \mu^I(\alpha)} k.$$

Connections between semantics Let $\sqsubseteq$ be the binary relation between provenance semantics such that $\mathcal{P} \sqsubseteq \mathcal{P}'$ if and only if $\mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha) \sqsubseteq \mathcal{P}'(\Sigma, D, \mathbb{K}, \lambda, \alpha)$ for every Σ, $(D, \mathbb{K}, \lambda)$ and α on which $\mathcal{P}$ and $\mathcal{P}'$ are well-defined.

Proposition 2. *The following holds:*

$$\mathcal{P}^{AM} \sqsubseteq \mathcal{P}^{AT} \text{ and } \mathcal{P}^{SAM} \sqsubseteq \mathcal{P}^{AT}.$$

Next examples show that $\mathcal{P}^{\text{AM}}$ and $\mathcal{P}^{\text{SAM}}$ are incomparable.

Example 6. *Let $\Sigma = \{A(x) \rightarrow \mathsf{goal}, B(x) \rightarrow \mathsf{goal}\}$, $D = \{A(a), B(a)\}$, $\lambda_{\mathbb{N}}(A(a)) = 2$ and $\lambda_{\mathbb{N}}(B(a)) = 3$.*

Annotated models of Σ and $(D, \mathbb{N}, \lambda_{\mathbb{N}})$ are such that $\mu^I(\mathsf{goal}) \geq 3$, so $\mathcal{P}^{AM}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal}) = 3$.

Set-annotated models of Σ and $(D, \mathbb{N}, \lambda_{\mathbb{N}})$ are such that $\{2, 3\} \subseteq \mu^I(\mathsf{goal})$, so $\mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal}) = 5$.

Hence $\mathcal{P}^{AM}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal}) < \mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal})$.

Example 7. *Let $\Sigma = \{R(x, y) \rightarrow \mathsf{goal}\}$, $D = \{R(a, b), R(a, c)\}$, $\lambda_{\mathbb{N}}(R(a, b)) = 2$ and $\lambda_{\mathbb{N}}(R(a, c)) = 2$.*

Annotated models of Σ and $(D, \mathbb{N}, \lambda_{\mathbb{N}})$ are such that $\mu^I(\mathsf{goal}) \geq 4$, so $\mathcal{P}^{AM}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal}) = 4$.

Set-annotated models of Σ and $(D, \mathbb{N}, \lambda_{\mathbb{N}})$ are such that $\{2\} \subseteq \mu^I(\mathsf{goal})$, so $\mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal}) = 2$.

Hence $\mathcal{P}^{AM}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal}) > \mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}, \lambda_{\mathbb{N}}, \mathsf{goal})$.

Despite of their inherently different approaches, $\mathcal{P}^{\text{AM}}$, $\mathcal{P}^{\text{SAM}}$ and $\mathcal{P}^{\text{AT}}$ coincide on a large class of semirings.

Proposition 3. *If $\mathbb{K}$ is a commutative $+$-idempotent ω-continuous semiring, then for every Σ, $(D, \mathbb{K}, \lambda)$, and α, $\mathcal{P}^{AM}(\Sigma, D, \mathbb{K}, \lambda, \alpha) = \mathcal{P}^{SAM}(\Sigma, D, \mathbb{K}, \lambda, \alpha) = \mathcal{P}^{AT}(\Sigma, D, \mathbb{K}, \lambda, \alpha)$.*

Additional insights on the connection between definitions can be gained by considering the provenance semiring $\mathbb{N}^\infty[\![X]\!]$: the monomials with non-zero coefficients are the same with all semantics but their coefficients may differ ($\mathcal{P}^{\text{AT}}$ leading to the highest coefficients by Proposition 2).

Proposition 4. *Let λ_X be an injective function from D to X.*

- *A monomial occurs in $\mathcal{P}^{AT}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \alpha)$ if and only if it occurs in $\mathcal{P}^{AM}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \alpha)$.*
- *$\mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \alpha)$ is obtained by setting all non-zero coefficients to 1 in $\mathcal{P}^{AT}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \alpha)$.*

An example where $\mathcal{P}^{\text{AT}}$ and $\mathcal{P}^{\text{AM}}$ or $\mathcal{P}^{\text{SAM}}$ differ on $\mathbb{N}^\infty[\![X]\!]$ is the following: Let Σ contain $A(x) \rightarrow B(x)$, $B(x) \rightarrow A(x)$, $D = \{A(a)\}$ and $\lambda_X(A(a)) = x$. Since there are infinitely many derivation trees for $A(a)$, $\mathcal{P}^{\text{AT}}(\Sigma, D, \mathbb{K}, \lambda_X, A(a)) = \infty x$ while for $\mathcal{P} \in \{\mathcal{P}^{\text{AM}}, \mathcal{P}^{\text{SAM}}\}$, $\mathcal{P}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, A(a)) = x$, as $\{A(a), B(a)\}$ with both facts annotated with x (resp. $\{x\}$) is a (resp. set-)annotated model for Σ and $(D, \mathbb{N}^\infty[\![X]\!], \lambda_X)$.

Note that $\mathcal{P}^{\text{AM}}$ and $\mathcal{P}^{\text{SAM}}$ can still lead to infinite provenance expressions: Let $\Sigma = \{A(x) \wedge B(x) \rightarrow A(x)\}$, $D = \{A(a), B(a)\}$, $\lambda_X(A(a)) = x$ and $\lambda_X(B(a)) = y$. For $\mathcal{P} \in \{\mathcal{P}^{\text{AM}}, \mathcal{P}^{\text{SAM}}\}$, $\mathcal{P}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, A(a)) = x + xy + xy^2 + xy^3 + \ldots$.

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_Σ. Applying T_Σ on a set of annotated facts $(I, \mathbb{K}, \lambda)$ results in $(I_{T_\Sigma}, \mathbb{K}, \lambda_{T_\Sigma})$ where I_{T_Σ} is the result of applying the immediate consequence operator to Σ and I, and λ_{T_Σ} annotates facts in I_{T_Σ} with the relational provenance (over $(I, \mathbb{K}, \lambda)$) of the UCQ formed by the bodies of the rules that create them. Formally,

$$I_{T_\Sigma} := \{H(\vec{a}) \mid I \models \exists \vec{y}\, \phi(\vec{a}, \vec{y}),\ \phi(\vec{x}, \vec{y}) \rightarrow H(\vec{x}) \in \Sigma\}$$

$$\lambda_{T_\Sigma}(H(\vec{a})) := \sum_{\substack{h(\vec{x}) = \vec{a},\ I \models h(\phi(\vec{x}, \vec{y})) \\ \phi(\vec{x}, \vec{y}) \rightarrow H(\vec{x}) \in \Sigma}} \prod_{\beta \in h(\phi(\vec{x}, \vec{y}))} \lambda(\beta)$$

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 $I^0_{\mathsf{n}}(\Sigma, D, \mathbb{K}, \lambda) := (D, \mathbb{K}, \lambda)$, and define inductively $I^{i+1}_{\mathsf{n}}(\Sigma, D, \mathbb{K}, \lambda) := T_\Sigma(I^i_{\mathsf{n}}(\Sigma, D, \mathbb{K}, \lambda)) \cup (D, \mathbb{K}, \lambda)$. Note that the subscript n of I_{n} is an abbreviation for 'naive', and the superscript i indicates how many times T_Σ was applied.

Let $(I^i_{\mathsf{n}}, \mathbb{K}, \lambda^i_{\mathsf{n}})$ denote $I^i_{\mathsf{n}}(\Sigma, D, \mathbb{K}, \lambda)$. We say that $I^i_{\mathsf{n}}(\Sigma, D, \mathbb{K}, \lambda)$ *converges* if there is some k such that $I^\ell_{\mathsf{n}} = I^k_{\mathsf{n}}$ for every $\ell \geq k$, and $\sup(\lambda^i_{\mathsf{n}}(\alpha))$ exists for every $\alpha \in I^k_{\mathsf{n}}$.

Proposition 5. *For every $\Sigma, D, \mathbb{K}, \lambda$, if $\mathbb{K}$ is ω-continuous then $I^i_{\mathsf{n}}(\Sigma, D, \mathbb{K}, \lambda)$ converges.*

In this case, we define $I^\infty_{\mathsf{n}} := I^k_{\mathsf{n}}$ and $\lambda^\infty_{\mathsf{n}} := \sup_{i\to\infty} \lambda^i_{\mathsf{n}}$. The *naive execution provenance semantics* $\mathcal{P}^{\text{NE}}$ is defined by

$$\mathcal{P}^{\text{NE}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \begin{cases} \lambda^\infty_{\mathsf{n}}(\alpha) & \alpha \in I^\infty_{\mathsf{n}} \\ 0 & \text{otherwise} \end{cases}$$

and is equivalent to the all-tree semantics.

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

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 $I^0_{\mathsf{o},\alpha}(\Sigma, D, \mathbb{K}, \lambda) := (D, \mathbb{K}, \lambda)$, and

$$I^{i+1}_{\mathsf{o},\alpha}(\Sigma, D, \mathbb{K}, \lambda) := \begin{cases} T_\Sigma(I^i_{\mathsf{o},\alpha}(\Sigma, D, \mathbb{K}, \lambda)) \cup (D, \mathbb{K}, \lambda) & \alpha \notin I^i_{\mathsf{o},\alpha} \\ I^i_{\mathsf{o},\alpha}(\Sigma, D, \mathbb{K}, \lambda) & \text{otherwise} \end{cases}$$

where $I^i_{\mathsf{o},\alpha}$ is such that $I^i_{\mathsf{o},\alpha}(\Sigma, D, \mathbb{K}, \lambda) := (I^i_{\mathsf{o},\alpha}, \mathbb{K}, \lambda^i_{\mathsf{o},\alpha})$.

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

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

$$\mathcal{P}^{\text{OE}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \begin{cases} \lambda^k_{\mathsf{o},\alpha}(\alpha) & \alpha \in I^k_{\mathsf{o},\alpha} \\ 0 & \text{otherwise} \end{cases}$$

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, Subotic, and Scholz 2020). Formally, let $\mathtt{depth}(t)$ denote the depth of tree t. We say that $t \in T^\Sigma_D(\alpha)$ is *of minimal depth* if for every $t' \in T^\Sigma_D(\alpha)$ it holds that $\mathtt{depth}(t) \leq \mathtt{depth}(t')$. The *minimal depth tree provenance semantics* $\mathcal{P}^{\text{MDT}}$ is defined by

$$\mathcal{P}^{\text{MDT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \sum_{\substack{t \in T^\Sigma_D(\alpha) \\ \text{is of minimal depth}}} \Lambda(t)$$

and is equivalent to the optimized naive execution.

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

Seminaive Evaluation / Hereditary Minimal Depth Trees In the seminaive evaluation algorithm, facts are derived only once. We introduce a new consequence operator Δ_Σ that derives only new facts and is defined as follows: $\Delta_\Sigma(I, \mathbb{K}, \lambda) := (I_{\Delta_\Sigma}, \mathbb{K}, \lambda_{\Delta_\Sigma})$ where $T_\Sigma(I, \mathbb{K}, \lambda) := (I_{T_\Sigma}, \mathbb{K}, \lambda_{T_\Sigma})$, $I_{\Delta_\Sigma} := I_{T_\Sigma} \setminus I$, and λ_{Δ_Σ} is the restriction of λ_{T_Σ} to I_{Δ_Σ}. We can now define the annotation aware version of the seminaive evaluation: $I^0_{\mathsf{sn}}(\Sigma, D, \mathbb{K}, \lambda) := (D, \mathbb{K}, \lambda)$ and $I^{i+1}_{\mathsf{sn}}(\Sigma, D, \mathbb{K}, \lambda) := I^i_{\mathsf{sn}}(\Sigma, D, \mathbb{K}, \lambda) \cup \Delta_\Sigma(I^i_{\mathsf{sn}}(\Sigma, D, \mathbb{K}, \lambda))$.

Proposition 9. *For every $\Sigma, D, \mathbb{K}, \lambda$, there exists $k \geq 0$ such that $I^k_{\mathsf{sn}}(\Sigma, D, \mathbb{K}, \lambda) = I^\ell_{\mathsf{sn}}(\Sigma, D, \mathbb{K}, \lambda)$ for every $\ell \geq k$.*

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

$$\mathcal{P}^{\text{SNE}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \begin{cases} \lambda^k_{\mathsf{sn}} & \alpha \in I^k_{\mathsf{sn}} \\ 0 & \text{otherwise} \end{cases}$$

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

$$\mathcal{P}^{\text{HMDT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \sum_{\substack{t \in T^\Sigma_D(\alpha) \\ \text{is hereditary minimal-depth}}} \Lambda(t)$$

and is equivalent to the seminaive execution.

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

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 a 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

$$\mathcal{P}^{\text{NRT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) := \sum_{\substack{t \in T^\Sigma_D(\alpha) \\ \text{is non-recursive}}} \Lambda(t).$$

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

Proposition 11. *The following hold:*

$$\mathcal{P}^{HMDT} \sqsubseteq \mathcal{P}^{NRT} \sqsubseteq \mathcal{P}^{AT} \text{ and } \mathcal{P}^{HMDT} \sqsubseteq \mathcal{P}^{MDT} \sqsubseteq \mathcal{P}^{AT}$$

Moreover $\mathcal{P}^{\text{NRT}}$ and $\mathcal{P}^{\text{AT}}$ coincide on specific semirings.

Proposition 12. *For every $\Sigma, D, \mathbb{K}, \lambda$ and α, if $\mathbb{K}$ is a commutative absorptive ω-continuous semiring, then $\mathcal{P}^{NRT}(\Sigma, D, \mathbb{K}, \lambda, \alpha) = \mathcal{P}^{AT}(\Sigma, D, \mathbb{K}, \lambda, \alpha)$.*

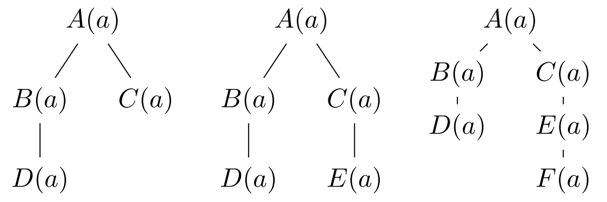
If $\mathbb{K}$ is not absorptive, there exists Σ, $(D, \mathbb{K}, \lambda)$ and α such that $\mathcal{P}^{\text{NRT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha) \neq \mathcal{P}^{\text{AT}}(\Sigma, D, \mathbb{K}, \lambda, \alpha)$, even in the case where $\mathbb{K}$ is $+$-idempotent and $\times$-idempotent: Let Σ consist of the rule $A(x) \wedge B(x) \rightarrow A(x)$ and $D = \{A(a), B(a)\}$. Then $\mathcal{P}^{\text{NRT}}(\Sigma, D, \mathbb{K}, \lambda, A(a)) = \lambda(A(a))$ while $\mathcal{P}^{\text{AT}}(\Sigma, D, \mathbb{K}, \lambda, A(a)) = \lambda(A(a)) + \lambda(A(a)) \times \lambda(B(a))$.

The other semantics differ even under strong restrictions.

Example 8. *This example shows that $\mathcal{P}^{NRT}$, $\mathcal{P}^{MDT}$ and $\mathcal{P}^{HMDT}$ differ even if $\mathbb{K}$ is $+$ and $\times$-idempotent and absorptive.*

$$\begin{aligned} \textit{Let } \Sigma =& \{B(x) \wedge C(x) \rightarrow A(x),\ D(x) \rightarrow B(x),\\ & E(x) \rightarrow C(x),\ F(x) \rightarrow E(x)\}\\ D =& \{C(a),\ D(a),\ E(a),\ F(a)\} \end{aligned}$$

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



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

$$\begin{aligned} \mathcal{P}^{NRT}(\Sigma, D, \mathbb{K}, \lambda, A(a)) =& c \times d + d \times e + d \times f\\ \mathcal{P}^{MDT}(\Sigma, D, \mathbb{K}, \lambda, A(a)) =& c \times d + d \times e\\ \mathcal{P}^{HMDT}(\Sigma, D, \mathbb{K}, \lambda, A(a)) =& c \times d \end{aligned}$$

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 Σ, annotated database $(D, \mathbb{K}, \lambda)$ and fact α, an element $\mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha)$ in K such that:*

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

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

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, $\mathcal{P}$, Σ, D, $\mathbb{K}$, λ and α 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 λ_X an injective function $\lambda_X : D \mapsto X$.

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 Σ 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 the body of any rule. In this case, the equivalent UCQ Q^{Σ} of Σ 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 Σ is UCQ-defined with rule head $H(\vec{x})$ and $H \notin \mathcal{S}(D)$, then for every tuple $\vec{a}$ of same arity as $\vec{x}$, the relational provenance of $Q^{\Sigma}(\vec{a})$ is equal to $\mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, H(\vec{a}))$.*

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)**.**

$$\mathcal{P}(\Sigma, D, PosBool(X), \lambda_X, \alpha) = \bigvee_{\substack{D' \subseteq D\\ \Sigma, D' \models \alpha}} \bigwedge_{\beta \in D'} \lambda_X(\beta)$$

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 $\mathbb{K}_1$ to $\mathbb{K}_2$, then $h(\mathcal{P}(\Sigma, D, \mathbb{K}_1, \lambda, \alpha)) = \mathcal{P}(\Sigma, D, \mathbb{K}_2, h \circ \lambda, \alpha)$.*

We call Property 3 restricted to ω-continuous homomorphisms *Commutation with ω-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 ω-continuous semirings.

Property 4 (Any (ω-Continuous) Semiring)**.** *$\mathcal{P}$ satisfies the* Any Semiring Property *(resp.* Any ω-Continuous Semiring Property*) if the semiring domain of $\mathcal{P}$ contains the set of all commutative (resp. commutative ω-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* $(\alpha_1^1, \cdots, \alpha_{n_1}^1)$, …, $(\alpha_1^m, \cdots, \alpha_{n_m}^m)$*, it holds that*

$$\mathcal{P}(\Sigma', D, \mathbb{K}, \lambda, \mathsf{goal}) = \Sigma_{i=1}^m \Pi_{j=1}^{n_i} \mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha_j^i)$$

where $\Sigma' = \Sigma \cup \{\bigwedge_{j=1}^{n_i} \alpha_j^i \to \mathsf{goal} \mid 1 \leq i \leq m\}$*.*

We weaken the above by referring to each mode separately:

Property 6 (Joint Use)**.** *For all facts* $\alpha_1, \cdots, \alpha_n$*,*

$$\mathcal{P}(\Sigma', D, \mathbb{K}, \lambda, \mathsf{goal}) = \Pi_{j=1}^n \mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha_j)$$

where $\Sigma' = \Sigma \cup \{\bigwedge_{j=1}^{n} \alpha_j \to \mathsf{goal}\}$*.*

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

$$\mathcal{P}(\Sigma', D, \mathbb{K}, \lambda, \mathsf{goal}) = \Sigma_{i=1}^m \mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha_i)$$

where $\Sigma' = \Sigma \cup \{\alpha_i \to \mathsf{goal} \mid 1 \leq i \leq m\}$*.*

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* $\lambda(\alpha) \sqsubseteq \mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha)$*.*

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* Σ_D of Σ w.r.t. D, defined by $\Sigma_D = \{h(\phi(\vec{x}, \vec{y})) \to h(H(\vec{x})) \mid \phi(\vec{x}, \vec{y}) \to H(\vec{x}) \in \Sigma, h : \vec{x} \cup \vec{y} \mapsto \mathcal{D}(D)\}$. 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* Σ_D *then* $\mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha) = \lambda(\alpha)$*.*

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* $\Sigma, D, \mathbb{K}, \lambda, \alpha$, $\mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha) = \mathcal{P}(\Sigma_D, D, \mathbb{K}, \lambda, \alpha)$*, 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 *Nec* the set of such facts.

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

A fact is *usable* to $\Sigma, D \models \alpha$ if it occurs in some derivation tree in $T_D^\Sigma(\alpha)$. 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* λ' *that differs from* λ *only on facts that are not usable to* $\Sigma, D \models \alpha$*, it holds that* $\mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha) = \mathcal{P}(\Sigma, D, \mathbb{K}, \lambda', \alpha)$*.*

4.5 Data Modification

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

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

$$\mathcal{P}(\Sigma, D, \mathbb{K}, \lambda, \alpha) + \mathcal{P}(\Sigma, D', \mathbb{K}, \lambda', \alpha) \sqsubseteq \mathcal{P}(\Sigma, D \cup D', \mathbb{K}, \lambda \cup \lambda', \alpha).$$

Maintaining provenance upon fact deletion is very useful in practice. We formalize this using a provenance semiring, which allows us to keep track of the facts. A partial evaluation of a provenance expression $p(X)$ over variables X is an expression obtained from $p(X)$ by replacing some of the variables by a given value.

Property 13 (Deletion)**.** *For every provenance semiring* $Prov(X)$ *and* $D' \subseteq D$*, if* λ' *is the restriction of* λ_X *to* D' *and* $\Delta = D \setminus D'$*, then* $\mathcal{P}(\Sigma, D', Prov(X), \lambda', \alpha)$ *is equal to the partial evaluation of* $\mathcal{P}(\Sigma, D, Prov(X), \lambda_X, \alpha)$ *obtained by setting the annotations of facts in* Δ *to* 0*:* $\mathcal{P}(\Sigma, D, Prov(X), \lambda_X, \alpha)[\{\lambda_X(x) = 0\}_{x \in \Delta}]$*.*

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.

	$\mathcal{P}^{\mathrm{AT}}$	$\mathcal{P}^{\mathrm{NRT}}$	$\mathcal{P}^{\mathrm{MDT}}$	$\mathcal{P}^{\mathrm{HMDT}}$	$\mathcal{P}^{\mathrm{AM}}$	$\mathcal{P}^{\mathrm{SAM}}$
Algebra Consistency	✓	✓	✓	✓		
Boolean Compat.	✓	✓			✓	✓
Com. with Hom.		✓	✓	✓		
Com. with ω-Cont.	✓	✓	✓	✓		
Any Semiring		✓	✓	✓		
Any ω-Cont. Sem.	✓	✓	✓	✓		✓
Joint and Alt. Use	✓	✓				
Joint Use	✓	✓		✓	✓	
Alternative Use	✓	✓				
Self	✓	✓	✓	✓	✓	✓
Parsimony	✓	✓	✓	✓	✓	✓
Necessary Facts	✓	✓	✓	✓		✓
Non-Usable Facts	✓	✓	✓	✓	✓	✓
Insertion	✓	✓				
Deletion	✓	✓			✓	✓

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

5.1 Tree- and Execution-Based Semantics Cases

We first discuss $\mathcal{P}^{\mathrm{MDT}}$ and $\mathcal{P}^{\mathrm{HMDT}}$, which have not been much investigated and stand out compared to $\mathcal{P}^{\mathrm{AT}}$ and $\mathcal{P}^{\mathrm{NRT}}$. 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 Σ and $(D, Prov(X), \lambda_X)$ as follows.*

$$\Sigma = \{A(x) \rightarrow \mathsf{goal},\ B(x) \rightarrow \mathsf{goal},\ C(x) \rightarrow B(x)\}$$
$$D = \{A(a),\ C(a)\} \text{ with } \lambda_X(A(a)) = a,\ \lambda_X(C(a)) = c$$

It holds that both $\mathcal{P}^{MDT}(\Sigma, D, Prov(X), \lambda_X, \mathsf{goal})$ and $\mathcal{P}^{HMDT}(\Sigma, D, Prov(X), \lambda_X, \mathsf{goal})$ are equal to a. For $\mathcal{P} \in \{\mathcal{P}^{MDT}, \mathcal{P}^{HMDT}\}$ we then have the following:
(i) The Boolean provenance of goal is $a \vee c$, hence $\mathcal{P}$ does not satisfy the Boolean Compatibility Property.
(ii) Since $\mathcal{P}(\emptyset, D, Prov(X), \lambda_X, A(a)) = a$ and $\mathcal{P}(\emptyset, D, Prov(X), \lambda_X, C(a)) = c$, $\mathcal{P}$ does not satisfy the Alternative Use, nor the Joint and Alternative Use Property.
(iii) Let $D' = \{\mathsf{goal}\}$ and $\lambda'(\mathsf{goal}) = g$. It holds that $\mathcal{P}(\Sigma, D \cup D', Prov(X), \lambda_X \cup \lambda'_X, \mathsf{goal}) = g$, which is different from $\mathcal{P}(\Sigma, D, Prov(X), \lambda_X, \mathsf{goal}) + \mathcal{P}(\Sigma, D', Prov(X), \lambda'_X, \mathsf{goal}) + e$ for every $e \in Prov(X)$. Hence $\mathcal{P}$ does not satisfy the Insertion Property.
(iv) Let $D' = D \setminus \{A(a)\} = \{C(a)\}$. The partial evaluation of $\mathcal{P}(\Sigma, D, Prov(X), \lambda_X, \mathsf{goal})$ where a is set to 0 is equal to 0 while $\mathcal{P}(\Sigma, D', Prov(X), \lambda_X, \mathsf{goal}) = c$. Hence $\mathcal{P}$ does not satisfy the Deletion Property.

We now illustrate the difference between $\mathcal{P}^{\mathrm{HMDT}}$ and $\mathcal{P}^{\mathrm{MDT}}$: $\mathcal{P}^{\mathrm{HMDT}}$ satisfies the Joint Use Property while $\mathcal{P}^{\mathrm{MDT}}$ does not.

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 $\Sigma' = \Sigma \cup \{A(a) \wedge B(a) \rightarrow \mathsf{goal}\}$. $\mathcal{P}^{MDT}(\Sigma', D, \mathbb{K}, \lambda, \mathsf{goal}) = b \times d + c \times d$ while $\mathcal{P}^{MDT}(\Sigma, D, \mathbb{K}, \lambda, A(a)) = d$ and $\mathcal{P}^{MDT}(\Sigma, D, \mathbb{K}, \lambda, B(a)) = b$. Hence $\mathcal{P}^{MDT}$ does not satisfy the Joint Use Property.*

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

Example 11. *Consider the semiring $\mathbb{N}^\infty$ with the classical operations, and define $\mathbb{N}^{\infty,\infty'}$ as its extension by an element ∞' 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 ω-continuous and $h : \mathbb{N}^\infty \mapsto \mathbb{N}^{\infty,\infty'}$ defined by $h(n) = n$ for every $n \in \mathbb{N}$, $h(\infty) = \infty'$ is a semiring homomorphism (which is not ω-continuous). Assume that $\mathcal{P}^{AT}(\Sigma, D, \mathbb{N}^\infty, \lambda, \mathsf{goal}) = \Sigma_{i\in\mathbb{N}} 1 = \infty$. Then $h(\mathcal{P}^{AT}(\Sigma, D, \mathbb{N}^\infty, \lambda, \mathsf{goal})) = \infty'$ is different from $\mathcal{P}^{AT}(\Sigma, D, \mathbb{N}^{\infty,\infty'}, h \circ \lambda, \mathsf{goal}) = \Sigma_{i\in\mathbb{N}} h(1) = \infty$.*

5.2 Model-Based Semantics Cases

On $+$-idempotent semirings, $\mathcal{P}^{\mathrm{AM}}$ and $\mathcal{P}^{\mathrm{SAM}}$ coincide with $\mathcal{P}^{\mathrm{AT}}$ so verify the same properties, and the semiring $\mathbb{B}[\![X]\!]$ 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 (ω-Continuous) Homomorphisms Properties.

Example 12. *Let $\Sigma = \{A(x) \rightarrow \mathsf{goal}, B(x) \rightarrow \mathsf{goal}\}$ 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 $\mathcal{P}^{AM}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \mathsf{goal})$ and $\mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \mathsf{goal})$ are equal to $x + y$.

Consider now the semiring $\mathbb{N}^\infty$, $\lambda_\mathbb{N}(A(a)) = 2$ and $\lambda_\mathbb{N}(B(a)) = 2$. Both $\mathcal{P}^{AM}(\Sigma, D, \mathbb{N}^\infty, \lambda_\mathbb{N}, \mathsf{goal})$ and $\mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}^\infty, \lambda_\mathbb{N}, \mathsf{goal})$ are equal to 2.

For $\mathcal{P} \in \{\mathcal{P}^{AM}, \mathcal{P}^{SAM}\}$ we then have the following:
(i) Let h be a ω-continuous homomorphism from $\mathbb{N}^\infty[\![X]\!]$ to $\mathbb{N}^\infty$ such that $h(x) = 2$ and $h(y) = 2$. Since $h(x + y) = h(x) + h(y) = 4$, $\mathcal{P}$ does not satisfy the Commutation with ω-Continuous Homomorphisms Property.
(ii) The relational provenance of $Q^\Sigma()$ w.r.t. $\mathbb{N}^\infty$ and $\lambda_\mathbb{N}$ is 4 so $\mathcal{P}$ does not satisfy the Algebra Consistency Property.
(iii) $\mathcal{P}(\emptyset, D, \mathbb{N}^\infty, \lambda_\mathbb{N}, A(a)) + \mathcal{P}(\emptyset, D, \mathbb{N}^\infty, \lambda_\mathbb{N}, B(a)) = 4$ so $\mathcal{P}$ does not satisfy the Alternative Use nor the Joint and Alternative Use Property.
(iv) Since $\mathcal{P}(\Sigma, \{A(a)\}, \mathbb{N}^\infty, \lambda_\mathbb{N}, \mathsf{goal}) + \mathcal{P}(\Sigma, \{B(a)\}, \mathbb{N}^\infty, \lambda_\mathbb{N}, \mathsf{goal}) = 4$ is strictly greater than $\mathcal{P}(\Sigma, D, \mathbb{N}^\infty, \lambda_\mathbb{N}, \mathsf{goal})$, $\mathcal{P}$ does not satisfy the Insertion Property.

Moreover, $\mathcal{P}^{\mathrm{SAM}}$ does not satisfy the Joint Use Property.

Example 13. *Let*

$$\Sigma = \{A(x) \rightarrow \mathsf{g}_1, A(x) \rightarrow \mathsf{g}_2, B(x) \rightarrow \mathsf{g}_1, B(x) \rightarrow \mathsf{g}_2\}$$
$$D = \{A(a), B(a)\} \text{ with } \lambda_X(A(a)) = x, \lambda_X(B(a)) = y.$$

Both $\mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \mathsf{g}_1)$ and $\mathcal{P}^{SAM}(\Sigma, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \mathsf{g}_2)$ are equal to $x + y$ but $\mathcal{P}^{SAM}(\Sigma \cup \{\mathsf{g}_1 \wedge \mathsf{g}_2 \rightarrow \mathsf{goal}\}, D, \mathbb{N}^\infty[\![X]\!], \lambda_X, \mathsf{goal}) = x^2 + y^2 + xy \neq (x + y)^2$.

We show that $\mathcal{P}^{\mathrm{AM}}$ 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. $\mathcal{P}^{\mathrm{NRT}}$ 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 $\mathcal{P}^{\mathrm{AT}}$, $\mathcal{P}^{\mathrm{MDT}}$, and $\mathcal{P}^{\mathrm{HMDT}}$. The equivalence between the tree-based $\mathcal{P}^{\mathrm{AT}}$, $\mathcal{P}^{\mathrm{NRT}}$ and model-based $\mathcal{P}^{\mathrm{AM}}$ and $\mathcal{P}^{\mathrm{SAM}}$ 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. Deutch et al. (2014) studied semirings for which the provenance expressions given by $\mathcal{P}^{\mathrm{AT}}$ 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 $\mathcal{P}^{\mathrm{MDT}}$ and $\mathcal{P}^{\mathrm{HMDT}}$ can be represented by polynomial size circuits regardless of the semiring. On the contrary, we show that (assuming $\mathsf{P} \neq \mathsf{NP}$) there is no polynomially computable circuit that computes $\mathcal{P}^{\mathrm{NRT}}$ 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 $\mathcal{P}^{\mathrm{NRT}}$ 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

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Achs, Á., and Kiss, A. 1995. Fuzzy extension of datalog. *Acta Cybern.* 12(2):153–166.

Arenas, M.; Conca, S.; and Pérez, J. 2012. Counting beyond a yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In Mille, A.; Gandon, F.; Misselis, J.; Rabinovich, M.; and Staab, S., eds., *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, 629–638. ACM.

Bourgaux, C.; Ozaki, A.; Peñaloza, R.; and Predoiu, L. 2020. Provenance for the description logic elhr. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 1862–1869. ijcai.org.

Bourgaux, C.; Bourhis, P.; Peterfreund, L.; and Thomazo, M. 2022. Revisiting semiring provenance for Datalog. arxiv.org/abs/2202.10766.

Calvanese, D.; Lanti, D.; Ozaki, A.; Peñaloza, R.; and Xiao, G. 2019. Enriching ontology-based data access with provenance. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 1616–1623. ijcai.org.

Ceylan, İ. İ.; Lukasiewicz, T.; Malizia, E.; and Vaicenavicius, A. 2019. Explanations for query answers under existential rules. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 1639–1646. ijcai.org.

Ceylan, İ. İ.; Lukasiewicz, T.; Malizia, E.; and Vaicenavicius, A. 2020. Explanations for ontology-mediated query answering in description logics. In Giacomo, G. D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; and Lang, J., eds., *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 672–679. IOS Press.

Cheney, J.; Chiticariu, L.; and Tan, W. C. 2009. Provenance in databases: Why, how, and where. *Found. Trends Databases* 1(4):379–474.

Cui, Y.; Widom, J.; and Wiener, J. L. 2000. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.* 25(2):179–227.

Dannert, K. M.; Grädel, E.; Naaf, M.; and Tannen, V. 2021. Semiring provenance for fixed-point logic. In Baier, C., and Goubault-Larrecq, J., eds., *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, 17:1–17:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Deutch, D.; Milo, T.; Roy, S.; and Tannen, V. 2014. Circuits for datalog provenance. In Schweikardt, N.; Christophides, V.; and Leroy, V., eds., *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, 201–212. OpenProceedings.org.

Deutch, D.; Gilad, A.; and Moskovitch, Y. 2018. Efficient provenance tracking for datalog using top-k queries. *VLDB J.* 27(2):245–269.

Esparza, J., and Luttenberger, M. 2011. Solving fixed-point equations by derivation tree analysis. In Corradini, A.; Klin, B.; and Cîrstea, C., eds., *Algebra and Coalgebra in Computer Science*, 19–35. Berlin, Heidelberg: Springer Berlin Heidelberg.

Green, T. J., and Tannen, V. 2017. The semiring framework for database provenance. In Sallinger, E.; den Bussche, J. V.; and Geerts, F., eds., *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, 93–99. ACM.

Green, T. J.; Karvounarakis, G.; and Tannen, V. 2007. Provenance semirings. In Libkin, L., ed., *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, 31–40. ACM.

Green, T. J. 2009. Containment of conjunctive queries on annotated relations. In Fagin, R., ed., *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*, 296–309. ACM.

Hernich, A., and Kolaitis, P. G. 2017. Foundations of information integration under bag semantics. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, 1–12. IEEE Computer Society.

Mumick, I. S.; Pirahesh, H.; and Ramakrishnan, R. 1990. The magic of duplicates and aggregates. In McLeod, D.; Sacks-Davis, R.; and Schek, H., eds., *16th International Conference on Very Large Data Bases, August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings*, 264–277. Morgan Kaufmann.

Nikolaou, C.; Kostylev, E. V.; Konstantinidis, G.; Kaminski, M.; Grau, B. C.; and Horrocks, I. 2017. The bag semantics of ontology-based data access. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 1224–1230. ijcai.org.

Nikolaou, C.; Kostylev, E. V.; Konstantinidis, G.; Kaminski, M.; Grau, B. C.; and Horrocks, I. 2019. Foundations of ontology-based data access under bag semantics. *Artif. Intell.* 274:91–132.

Senellart, P. 2017. Provenance and probabilities in relational databases. *SIGMOD Rec.* 46(4):5–15.

Soufflé. 2020. https://souffle-lang.github.io/index.html.

Stüber, T., and Vogler, H. 2008. Weighted monadic datalog. *Theoretical Computer Science* 403(2):221–238.

Zhao, D.; Subotic, P.; and Scholz, B. 2020. Debugging large-scale datalog: A scalable provenance evaluation strategy. *ACM Trans. Program. Lang. Syst.* 42(2):7:1–7:35.