# The Complexity of Why-Provenance for Datalog Queries (Extended Abstract)

**Marco Calautti**[1] , **Ester Livshits**[2] , **Andreas Pieris**[3,4] , **Markus Schneider**[5]

[1]University of Milan
[2]Technion
[3]University of Cyprus
[4]University of Edinburgh
[5]King's College London

## Datalog

Datalog has emerged in the 1970s as a logic-based query language from Logic Programming and has been extensively studied since then (Abiteboul, Hull, and Vianu 1995). The name Datalog reflects the intention of devising a counterpart of Prolog for data processing. It essentially extends the language of unions of conjunctive queries, which corresponds to the select-from-where fragment of SQL or the select-project-join-union fragment of relational algebra, with the key feature of recursion needed to express some natural queries. Among numerous applications, Datalog has been used in symbolic AI as a powerful knowledge representation language. It has been also used as a tool for efficiently executing queries over knowledge bases. In particular, for several important ontology languages based on description logics and existential rules, ontological query answering can be reduced to the problem of evaluating a Datalog query (see, e.g., (Eiter et al. 2012; Benedikt et al. 2022)), which enables the exploitation of efficient Datalog engines such as DLV (Leone et al. 2006) and Clingo (Gebser et al. 2016).

## Explainability of Datalog

As for any other query language, explaining why a result to a Datalog query is obtained is crucial towards explainable and transparent data-intensive applications. A standard way for providing such explanations to query answers is, among others, the so-called *why-provenance* (Buneman, Khanna, and Tan 2001). Its essence is to collect all the subsets of the input database that as a whole can be used to derive a certain answer. More precisely, in the case of Datalog queries, the why-provenance of an answer tuple $(c_1, \ldots, c_n)$ is obtained by considering all the possible proof trees $T$ of the fact $\mathrm{Answer}(c_1, \ldots, c_n)$, with $\mathrm{Answer}$ being the answer predicate of the Datalog query in question, and then collecting all the database facts that label the leaves of $T$. Recall that a proof tree of a fact $\alpha$ w.r.t. a database $D$ and a set $\Sigma$ of Datalog rules forms a tree-like representation of a way for deriving $\alpha$ by starting from $D$ and executing the rules occurring in $\Sigma$ (Abiteboul, Hull, and Vianu 1995).

Interestingly, why-provenance (as well as other notions of provenance that can be found in the literature) for Datalog queries can be captured via the unifying framework of provenance semirings proposed by Green, Karvounarakis, and Tannen (Green, Karvounarakis, and Tannen 2007). The central idea is to annotate, using some function $\mu$, the facts of the input database $D$ with values coming from a certain semiring $\mathcal{K}$, and then define, for each fact $\alpha$ that can be derived using $D$ and the input Datalog program $\Sigma$, its provenance w.r.t. $\mathcal{K}$ as

$$\bigoplus_{T \in \mathsf{trees}(\alpha, D, \Sigma)} \quad \bigotimes_{v \text{ is a leaf of } T} \mu(\lambda_T(v)),$$

where $\mathsf{trees}(\alpha, D, \Sigma)$ is the set of all proof trees of $\alpha$ w.r.t. $D$ and $\Sigma$, $\oplus$ and $\otimes$ are the addition and multiplication operators provided by $\mathcal{K}$, and $\lambda_T$ is the labeling function that assigns facts to the nodes of $T$. Interestingly, depending on the choice of the semiring $\mathcal{K}$, the above expression encodes different kinds of provenance. In particular, why-provenance is captured by choosing $\mathcal{K}$ to be the so-called *why-provenance semiring* (Green 2011; Green and Tannen 2017).

There are several works that studied why-provenance for Datalog queries. In particular, there are theoretical studies on computing the why-provenance (Damásio, Analyti, and Antoniou 2013; Deutch et al. 2014; Khamis et al. 2022), attempts to under-approximate the why-provenance towards an efficient computation (Zhao, Subotic, and Scholz 2020), studies on the restricted setting of non-recursive Datalog queries (Lee, Ludäscher, and Glavic 2019), attempts to compute the why-provenance by transforming the grounded Datalog rules to a system of equations (Esparza, Luttenberger, and Schlund 2014), and attempts to compute the why-provenance on demand via transformations to existential rules (Elhalawati, Krötzsch, and Mennicke 2022).

## Main Research Question

Despite all the above research activity, the fundamental task of pinpointing the exact data complexity of why-provenance for Datalog queries, i.e., of the problem of deciding whether a subset of the database explains a certain result to a Datalog query according to why-provenance, has been only recently addressed in (Calautti et al. 2024a).[1] More precisely, the goal of (Calautti et al. 2024a) was, for a Datalog query $Q$, to study the complexity of the following problem, dubbed $\mathsf{Why\text{-}Provenance}[Q]$: given a database $D$, a tuple $\bar{t}$, and a subset $D'$ of $D$, is it the case that $D'$ belongs to the why-provenance of $\bar{t}$ w.r.t. $D$ and $Q$? Pinpointing the complexity

[1]This extended abstract is based on (Calautti et al. 2024a).

of the above decision problem let us understand the inherent complexity of why-provenance for Datalog queries w.r.t. the size of the database, which is precisely what matters when using why-provenance in practice.

## Main Results

The takeaway of (Calautti et al. 2024a) is that explaining Datalog queries via why-provenance is, in general, an intractable problem. In particular, for a Datalog query $Q$, Why-Provenance$[Q]$ is in NP, and there are queries for which it is NP-hard. It has been further analyzed the complexity of the problem when $Q$ is linear (i.e., the recursion is restricted to be linear) or non-recursive, with the aim of clarifying whether the feature of recursion affects the inherent complexity of why-provenance. We know from (Calautti et al. 2024a) that restricting the recursion to be linear does not affect the complexity, namely the problem is in NP and for some queries it is even NP-hard. However, completely removing the recursion significantly reduces the complexity; in particular, we know that the problem is in $\mathrm{AC}_0$, which has been shown via first-order rewritability. Note that the NP-completeness results mentioned above opened up the possibility of devising algorithms for computing the why-provenance for Datalog queries based on SAT solvers. First steps towards this direction have been recently made in (Calautti et al. 2024b).

It is clear that the notion of why-provenance for Datalog queries, and hence the problem Why-Provenance$[Q]$, heavily rely on the notion of proof tree. However, as already discussed in the literature (see, e.g., the recent work (Bourgaux et al. 2022)), there are proof trees that are counter-intuitive since they represent unnatural derivations (e.g., a fact is used to derive itself), and this also affects the why-provenance. With the aim of overcoming this conceptual limitation of proof trees, (Calautti et al. 2024a) considered refined classes of proof trees, which have been recently proposed in (Bourgaux et al. 2022). In particular, non-recursive and minimal-depth proof trees, which are inspired by practical needs, have been considered. For instance, non-recursive proof trees resembles the approach taken by some graph query languages to handle queries with possibly infinite outputs by allowing to explicitly restrict the output to include only simple paths. Moreover, minimal-depth proof trees capture the behavior of Datalog engines, such as Souffle, that store only minimal-depth proof trees instead of storing them all (which might be impossible in case there are infinitely many). The problem Why-Provenance$[Q]$ focusing on non-recursive and minimal-depth proof trees has been also studied in (Calautti et al. 2024a), and shown that its complexity remains the same. This is a rather positive outcome as we can overcome the limitation of arbitrary proof trees without increasing the data complexity.

## References


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

Benedikt, M.; Buron, M.; Germano, S.; Kappelmann, K.; and Motik, B. 2022. Rewriting the infinite chase. *PVLDB* 15(11):3045–3057.

Bourgaux, C.; Bourhis, P.; Peterfreund, L.; and Thomazo, M. 2022. Revisiting semiring provenance for datalog. In *KR*.

Buneman, P.; Khanna, S.; and Tan, W. C. 2001. Why and where: A characterization of data provenance. In *ICDT*, 316–330.

Calautti, M.; Livshits, E.; Pieris, A.; and Schneider, M. 2024a. The complexity of why-provenance for datalog queries. *Proc. ACM Manag. Data* 2(2):83.

Calautti, M.; Livshits, E.; Pieris, A.; and Schneider, M. 2024b. Computing the why-provenance for datalog queries via sat solvers. In *AAAI*.

Damásio, C. V.; Analyti, A.; and Antoniou, G. 2013. Justifications for logic programming. In *LPNMR*, 530–542.

Deutch, D.; Milo, T.; Roy, S.; and Tannen, V. 2014. Circuits for datalog provenance. In *ICDT*, 201–212.

Eiter, T.; Ortiz, M.; Simkus, M.; Tran, T.; and Xiao, G. 2012. Query rewriting for horn-shiq plus rules. In *AAAI*.

Elhalawati, A.; Krötzsch, M.; and Mennicke, S. 2022. An existential rule framework for computing why-provenance on-demand for datalog. In *RuleML+RR*.

Esparza, J.; Luttenberger, M.; and Schlund, M. 2014. Fpsolve: A generic solver for fixpoint equations over semirings. In *CIAA*, 1–15.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; and Wanko, P. 2016. Theory solving made easy with clingo 5. In *ICLP*, 2:1–2:15.

Green, T. J., and Tannen, V. 2017. The semiring framework for database provenance. In *PODS*, 93–99. ACM.

Green, T. J.; Karvounarakis, G.; and Tannen, V. 2007. Provenance semirings. In *PODS*, 31–40.

Green, T. J. 2011. Containment of conjunctive queries on annotated relations. *Theory Comput. Syst.* 49(2):429–459.

Khamis, M. A.; Ngo, H. Q.; Pichler, R.; Suciu, D.; and Wang, Y. R. 2022. Convergence of datalog over (pre-) semirings. In *PODS*, 105–117. ACM.

Lee, S.; Ludäscher, B.; and Glavic, B. 2019. PUG: a framework and practical implementation for why and why-not provenance. *VLDB J.* 28(1):47–71.

Leone, N.; Pfeifer, G.; Faber, W.; Eiter, T.; Gottlob, G.; Perri, S.; and Scarcello, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log.* 7(3):499–562.

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.