Foundations of Declarative Data Analysis Using Limit Datalog Programs (Extended Abstract)

Mark Kaminski, Bernardo Cuenca Grau, Egor V. Kostylev, Boris Motik, and Ian Horrocks

Department of Computer Science, University of Oxford, UK

Analysing complex datasets is currently a hot topic in information systems. The term 'data analysis' covers a broad range of techniques that often involve tasks such as data aggregation, property verification, or query answering. Such tasks are currently often solved imperatively (e.g., using Java or Scala) by specifying *how* to manipulate the data, and this is undesirable because the objective of the analysis is often obscured by evaluation concerns. It has recently been argued that data analysis should be *declarative* [1, 12, 16, 17]: users should describe *what* the desired output is, rather than how to compute it. For example, instead of computing shortest paths in a graph by a concrete algorithm, one should (i) describe what a path length is, and (ii) select only paths of minimum length. Such a specification is independent of evaluation details, allowing analysts to focus on the task at hand. An evaluation strategy can be chosen later, and general parallel and/or incremental evaluation algorithms can be reused 'for free'.

An essential ingredient of declarative data analysis is an efficient language that can capture the relevant tasks, and Datalog is a prime candidate since it supports recursion. Apart from recursion, however, data analysis usually also requires integer arithmetic to capture quantitative aspects of data (e.g., the length of a shortest path). Research on combining the two dates back to the '90s [14, 10, 2, 18, 4, 8, 15], and is currently experiencing a revival [7, 13]. This extensive body of work, however, focuses primarily on integrating recursion and arithmetic with *aggregate functions* in a coherent semantic framework, where technical difficulties arise due to nonmonotonicity of aggregates. Surprisingly little is known about the computational properties of integrating recursion with arithmetic, apart from that a straightforward combination is undecidable [5]. Undecidability also carries over to the above formalisms and practical Datalog-based systems such as BOOM [1], DeALS [17], Myria [19], SociaLite [16], Overlog [11], Dyna [6], and Yedalog [3].

To develop a sound foundation for Datalog-based declarative data analysis, we study $Datalog_{\mathbb{Z}}$—negation-free Datalog with integer arithmetic and comparisons. Our main contribution is a new *limit* $Datalog_{\mathbb{Z}}$ fragment that, like the existing data analysis languages, is powerful and flexible enough to naturally capture many important analysis tasks. However, unlike $Datalog_{\mathbb{Z}}$ and the existing languages, reasoning with limit programs is decidable, and it becomes tractable in data complexity under an additional *stability* restriction.

In limit $Datalog_{\mathbb{Z}}$, all intensional predicates with a numeric argument are *limit predicates*. Instead of keeping all numeric values for a given tuple of objects, such predicates

Research supported by the Royal Society and the EPSRC projects DBOnto, MaSI^3, and ED^3.

keep only the minimal (min) or only the maximal (max) bounds of numeric values entailed for the tuple. For example, if we encode a weighted directed graph using a ternary predicate $edge$, then rules (1) and (2), where sp is a min limit predicate, compute the cost of a shortest path from a given source node v_0 to every other node.

$$\rightarrow sp(v_0, 0) \tag{1}$$

$$sp(x, m) \wedge edge(x, y, n) \rightarrow sp(y, m + n) \tag{2}$$

If these rules and a dataset entail a fact $sp(v, k)$, then the cost of a shortest path from v_0 to v is at most k; hence, $sp(v, k')$ holds for each $k' \geq k$ since the cost of a shortest path is also at most k'. Rule (2) intuitively says that, if x is reachable from v_0 with cost at most m and $\langle x, y \rangle$ is an edge of cost n, then v' is reachable from v_0 with cost at most $m + n$. This is different from $Datalog_{\mathbb{Z}}$, where there is no implicit semantic connection between $sp(v, k)$ and $sp(v, k')$, and such semantic connections allow us to prove decidability of limit $Datalog_{\mathbb{Z}}$. We provide a direct semantics for limit predicates based on Herbrand interpretations, but we also show that this semantics can be axiomatised in standard $Datalog_{\mathbb{Z}}$. Our formalism can thus be seen as a fragment of $Datalog_{\mathbb{Z}}$, from which it inherits well-understood properties such as monotonicity and existence of a least fixpoint model [5].

Our contributions are as follows. First, we introduce limit $Datalog_{\mathbb{Z}}$ programs and argue that they can naturally capture many relevant data analysis tasks. We prove that fact entailment in limit $Datalog_{\mathbb{Z}}$ is undecidable, but, after restricting the use of multiplication, it becomes coNEXPTIME- and coNP-complete in combined and data complexity, respectively. To achieve tractability in data complexity (which is very important for robust behaviour on large datasets), we additionally introduce a *stability* restriction and show that this does not prevent expressing the relevant analysis tasks.

References

1. Alvaro, P., Condie, T., Conway, N., Elmeleegy, K., Hellerstein, J.M., Sears, R.: BOOM analytics: exploring data-centric, declarative programming for the cloud. In: EuroSys. ACM (2010)
2. Beeri, C., Naqvi, S.A., Shmueli, O., Tsur, S.: Set constructors in a logic database language. J. Log. Program. 10(3&4) (1991)
3. Chin, B., von Dincklage, D., Ercegovac, V., Hawkins, P., Miller, M.S., Och, F.J., Olston, C., Pereira, F.: Yedalog: Exploring knowledge at scale. In: SNAPL (2015)
4. Consens, M.P., Mendelzon, A.O.: Low complexity aggregation in GraphLog and Datalog. Theor. Comput. Sci. 116(1) (1993)
5. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Comput. Surv. 33(3) (2001)
6. Eisner, J., Filardo, N.W.: Dyna: Extending datalog for modern AI. In: Datalog (2011)
7. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artif. Intell. 175(1) (2011)
8. Ganguly, S., Greco, S., Zaniolo, C.: Extrema predicates in deductive databases. J. Comput. Syst. Sci. 51(2) (1995)
9. Kaminski, M., Grau, B.C., Kostylev, E.V., Motik, B., Horrocks, I.: Foundations of declarative data analysis using limit datalog programs. CoRR abs/1705.06927 (2017)

10. Kemp, D.B., Stuckey, P.J.: Semantics of logic programs with aggregates. In: ISLP (1991)
11. Loo, B.T., Condie, T., Garofalakis, M.N., Gay, D.E., Hellerstein, J.M., Maniatis, P., Ramakrishnan, R., Roscoe, T., Stoica, I.: Declarative networking. Commun. ACM 52(11) (2009)
12. Markl, V.: Breaking the chains: On declarative data analysis and data independence in the big data era. PVLDB 7(13) (2014)
13. Mazuran, M., Serra, E., Zaniolo, C.: Extending the power of datalog recursion. VLDB J. 22(4) (2013)
14. Mumick, I.S., Pirahesh, H., Ramakrishnan, R.: The magic of duplicates and aggregates. In: VLDB. pp. 264–277 (1990)
15. Ross, K.A., Sagiv, Y.: Monotonic aggregation in deductive databases. J. Comput. System Sci. 54(1) (1997)
16. Seo, J., Guo, S., Lam, M.S.: SociaLite: An efficient graph query language based on datalog. IEEE Trans. Knowl. Data Eng. 27(7) (2015)
17. Shkapsky, A., Yang, M., Interlandi, M., Chiu, H., Condie, T., Zaniolo, C.: Big data analytics with datalog queries on Spark. In: SIGMOD. ACM (2016)
18. Van Gelder, A.: The well-founded semantics of aggregation. In: PODS (1992)
19. Wang, J., Balazinska, M., Halperin, D.: Asynchronous and fault-tolerant recursive datalog evaluation in shared-nothing engines. PVLDB 8(12) (2015)