Datalog: Bag Semantics via Set Semantics

Leopoldo Bertossi1, Georg Gottlob2,3 and Reinhard Pichler4

1 Carleton University, Ottawa and RelationalAI Inc.. bertossi@scs.carleton.ca
2 University of Oxford. georg.gottlob@cs.ox.ac.uk
3 TU Wien. firstname.surname@tuwien.ac.at

ABSTRACT

Duplicates in data management are common and problematic. In this work, we present a translation of Datalog under bag semantics into a well-behaved extension of Datalog (the so-called warded Datalog) under set semantics. From a practical point of view, this allows us to handle the bag semantics of Datalog by powerful, existing query engines for the required extension of Datalog. Moreover, this translation has the potential for large body of query optimization techniques.

1. INTRODUCTION

Duplicates are a common feature in data management. They appear, for instance, in relational databases queried by means of SQL and RDF data queried by means of SPARQL. However, the semantics of data operations and queries in the presence of duplicates is not always clear, mostly related to the fact that duplicates are handled by bags or multivalues, whereas the common logic-based semantics used in data management are set-theoretical, making it difficult to tell apart duplicates through the use of sets alone.

To address this problem, a bag semantics for Datalog programs was proposed in [13], what we call the derivation-tree bag semantics (DTB semantics). Intuitively, two duplicates of the same tuple in an intentional predicate are accepted as such, if they have syntactically different derivation trees. Also, an equivalent formulation was given in terms of semi-naive evaluation. The DTB semantics was used in [1] to provide a bag semantics for SPARQL.

The DTB semantics has two major drawbacks: first, it is operational – thus losing the declarative, logic-based semantics of Datalog; and second, it is defined via new constructs (the DTBs) – thus leaving the world of query languages and losing the applicability of the large body of query optimization techniques.

The goal of this paper, instead, is to identify an extension of Datalog which allows us, to express the bag semantics in terms of the classical set semantics, and stay within the realm of query languages. To this end, we show that the DTB semantics of a Datalog program can be represented by means of its transformation into a Datalog program, in such a way that the intended model of the former, including duplicates, can be characterized as the result of the duplicate-free chase instance for the latter. This is achieved by creating the right tuple identifiers (tids) by means of existential rules. Duplicates with different tids of the same tuple will be admissible and usual duplicates when falling back to a bag semantics for the original Datalog program. We establish the correspondence between the DTB semantics and ours.

The Datalog programs required for this task belong to the well-behaved class of warded Datalog programs [9], which properly extends Datalog, has a tractable (conjunctive) query answering (CQA) problem, and has been successfully applied to represent SPARQL under the OWL 2 QL core entailment regime [8], with set semantics though [9] (see also [2]). Warded Datalog looks promising as a general language for specifying different data management tasks.

We also show that Datalog with stratified negation can be captured by a similar transformation into a well-behaved class of Datalog. In this way, we achieve a fully declarative way of expressing the bag semantics of an important query language; and we immediately recover full relational algebra (including set difference) with bag semantics in terms of a well-behaved query language under set semantics. Moreover, the translation into Datalog ensures polynomial-time CQA.

2. PRELIMINARIES

We assume the reader to be familiar with the relational data model. An n-ary predicate P has positions: P[1],...,P[n]. With Pos(P) we denote the set of positions of predicate P. Similarly, Pos(Π) denotes the set of positions of (predicates in) a Datalog program Π.

Basic Multiset Operations. We follow [1]3. Consider multisets M and elements e from some domain with non-negative integer multiplicities, mult(e,M). By definition, e ∈ M if mult(e,M) ≥ 1. To a multiset M, we associate a set: set(M) := {e | e ∈ M}. Now consider multiset relations R,S. Unless stated otherwise, we assume that R,S contain tuples of the same arity, say n. Multiset R is (multi-) contained in S, denoted, R ⊆ m S iff for every e ∈ R, mult(e,R) ≤ mult(e,S).

1L. Bertossi is member of the “Millenium Research Institute on Foundations of Data” (IMFD), Chile.
Moreover, we define the following multiset operations:

The multiset union $\cup_m$ is defined by $R \cup_m S := T$, with $\text{mult}(e, T) := \text{mult}(e, R) + \text{mult}(e, S)$. Multiset selection $\sigma^m_C(R)$, with $C$ a condition, is defined as the multiset $T$ containing all tuples in $R$ that satisfy $C$ with multiplicities the same as in $R$.

The multiset projection $\pi^m_k(R)$ is defined as follows: let $k$ be a $k$-tuple $\langle t_1, \ldots, t_k \rangle$ of elements from $\{1, \ldots, n\}$; accordingly, for an $n$-tuple $\bar{t} = \langle t_1, \ldots, t_n \rangle \in R$, we consider the $k$-tuple $\langle t_{i_1}, \ldots, t_{i_k} \rangle$, with $\text{mult}(\pi^m_k(R))$ defined as the sum of the multiplicities in $R$ of tuples $\bar{t}$ producing $\bar{e}$.

For the multiset (natural) join assume that tuples have arity $n$ in $R$ and arity $n'$ in $S$. To simplify the presentation, assume that the natural join is via the last attribute of $R$ and first of $S$. Then we define the following multiset of $(n + n' - 1)$-tuples: $\bar{r} = \langle t_1, \ldots, t_{n+n'-1} \rangle \in R \bowtie^m S$ iff there are $\bar{r} \in R$ and $\bar{s} \in S$, such that $r_n = s_1$, $\bar{t}_{[1, \ldots, n]} = \bar{r}$ and $\bar{t}_{[n+1, \ldots, n+n'-1]} = \bar{s}$, with $\text{mult}(\bar{r}, R \bowtie^m S) = \sum_{\bar{r}, \bar{s} \text{ s.t. } r_n = s_1} \text{mult}(\bar{r}, R) \times \text{mult}(\bar{s}, S)$.

For the multiset difference, two definitions are considered: $R \setminus_m S := T$, with $\text{mult}(e, T) := \max\{\text{mult}(e, R) - \text{mult}(e, S), 0\}$. Alternatively, $R \setminus_{an} S := T$, with $\text{mult}(e, T) := \text{mult}(e, R)$ if $e \not\in S$ and $\text{mult}(e, T) := 0$, otherwise ("an" stands for all-or-nothing).

The multiset intersection $\cap_m$ is a contentious operation. Extending the derivation-tree based semantics from [13] to multiset intersection would treat $\cap_m$ as a special case of the join, which may be counter-intuitive. However, intersection is not treated or used in [11] [13].

### Derivation-tree bag semantics of Datalog

We follow [13], where tuples are “colored” to tell apart duplicates of a same element in the extensional database (EDB), via an infinite, ordered list $C$ of colors $c_1, c_2, \ldots$. For a multiset $M$ and $e \in M$, with $\text{mult}(e, M) = n > 0$, the $n$ copies, $e_1, e_2, \ldots, e_n$, of $e$ are colored with $c_1, \ldots, c_n$, respectively. So, $\text{col}(e) := \{e_1, \ldots, e_n\}$ becomes a set. For a multiset $M$, $\text{col}(M) := \cup_{e \in M} \text{col}(e)$, which is a set. For a “colored” set $S$, $\text{col}^{-1}(S)$ produces a multiset by stripping tuples from their colors.

**Example 2.1.** For $M = \{a, a, a, b, b, c\}$, $\text{col}(a) = \{a: 1, a: 2, a: 3\}$, and $\text{col}(M) = S = \{a: 1, a: 2, a: 3, b: 1, b: 2, c: 1\}$. The inverse operation, the decoloration, gives: $\text{col}^{-1}(a: 2) := a$; and $\text{col}^{-1}(S) := \{a, a, a, b, b, c\}$, a multiset. □

We now consider Datalog programs $P$ with multiset predicates, i.e. their extensions can be multisets, and multiset EDBs $E$. A derivation tree (DT) for $P$ is a tree with labeled nodes and edges, as follows:

1. For an EDB predicate $Q$ and $h \in \text{col}(Q(E))$, a DT for $\text{col}^{-1}(h)$ contains a single node with label $h$.
2. For each rule of the form
   \[ r: H \leftarrow A_1, A_2, \ldots, A_k, \quad k > 0, \]
   and for each tuple $\langle \tau_1, \ldots, \tau_k \rangle$ of DTs for the atoms $\langle d_1, \ldots, d_k \rangle$ that unify with $\langle A_1, A_2, \ldots, A_k \rangle$ with mgu $\theta$, generate a DT for $H\theta$ with $H\theta$ as the root and $(\tau_1, \ldots, \tau_k)$ as the edges, and $r$ as the label for the edges from the root to the children.

For a DT $\tau$, we define $\text{Atoms}(\tau) := \text{root-label or col}^{-1}$ of the root, with the latter case when $\tau$ is a single-node tree. For a set of DTs $\Xi$: $\text{Atoms}(\Xi) := \bigcup_{\tau \in \Xi} \text{Atoms}(\tau)$, which is a multiset that multi-contains $E$. If $DT(P, \text{col}(E))$ is the set of (syntactically different) DTs, the derivation-tree bag (DTB) semantics for $P$ is the multiset:

\[ \text{DTBS}(P, E) := \text{Atoms}(DT(P, \text{col}(E))). \]
with the $c_i$ pairwise different nulls from $\mathcal{T}$ as tids, and not used to identify any other element of $E$. We obtain a set EDB $E^+$ from the multiset EDB $E$.

Given a rule in $\Pi$, we introduce tid-variables (i.e. appearing in the 0-th positions) and existential quantifiers in the rule head, to formally generate fresh tids when the rule applies. More precisely, a rule in $\Pi$ of the form $r: H(x) \leftarrow A_1(x_1), A_2(x_2), \ldots, A_k(x_k)$, with $k > 0$, $x \subseteq \bigcup_i x_i$, becomes the Datalog$^\pm$ rule:

$r^\pm: \exists z H(z; x) \leftarrow A_1(z_1; x_1), A_2(z_2; x_2), \ldots, A_k(z_k; x_k)$

with fresh, different variables $z, z_1, \ldots, z_k$.

The resulting Datalog$^\pm$ program $\Pi^+$ can be evaluated according to the usual set semantics on the set EDB $E^+$. For this, we use the classical chase, i.e., when the instantiated body of rule $r^\pm$ becomes true, say with $A_1(u_1; a_1), A_2(u_2; a_2), \ldots, A_k(u_k; a_k)$, then the new tuple $H(c; u)$ is created, with $c$ the first (new) null from $T$ that has not been used yet. The new null stands for the tid of the newly created atom. The chase variant assumed here is the so-called obvious chase [11], i.e., a tid is activated for every instantiation of the rule body that makes the rule body true, but a rule is never activated more than once with the same instantiated body.

The chase instance is obtained by collecting all atoms obtained by applying the chase with $E^+$ and $\Pi^+$, i.e. that are final atoms in a derivation sequence. More precisely, a derivation sequence is of the form: $D_1 \leadsto r_1; A_1; \ldots; D_n \leadsto r_n; A_n$, with atoms $A_i$, such that the $r_i$ are tgd$s$, and each $D_k$ is a set of atoms contained in $E^+ \cup \{A_1, \ldots, A_{k-1}\}$ that makes the body of $r_k$ true, then $A_k$ is created.

**Example 3.1.** (ex. 2.2 cont.) Consider the modified program $\Pi = \{r_1, r_2, r_3\}$, with rules $r_2, r_3$ as before, and rule $r_1$ replaced by: $r_1': P(z, y) \leftarrow R(x, y), \neg S(x, y)$, i.e., body atom $S(x, y)$ is now negated.

Its rewriting into a Datalog$^\pm$ rule as in the positive case yields $r_1'^+: \exists z P(z, y) \leftarrow R(z_1, x), \neg S(z_2, x)$, variable $z_2$ occurring in a negated body atom but not in a positive one, making the rule no longer safe. We thus introduce an auxiliary predicate $Aux$ in order to eliminate this variable:

$r_1'^+: \exists z P(z, y) \leftarrow R(z_1, x), \neg Aux(x, y)$

This idea can be generalized to transform safe Datalog programs with stratified negation (denoted Datalog$^{\pm}$), by rewriting a rule of the form $r: H(x) \leftarrow A_1(x_1), \ldots, A_k(x_k)$, $\neg N_1(x_{k+1}), \ldots, \neg N_j(x_{k+j})$ into:

$r': \exists z H(z; x) \leftarrow A_1(z_1; x_1), \ldots, A_k(z_k; x_k), \neg Aux_1(x_{k+1}), \ldots, \neg Aux_j(x_{k+j})$.

We keep denoting with $PBBS(\Pi, E)$ the chase-based bag semantics for Datalog with stratified and safe negation (referred to as Datalog$^{\pm}$) obtained via this Datalog$^\pm$ rewriting with tids.

In [14], a bag semantics for Datalog$^{\pm}$ was introduced via derivation-trees (DT$s$), extending the DTB semantics in [13] for (positive) Datalog. This extension applies to Datalog programs with stratified negation that are range-restricted and safe (i.e. a variable in a rule head or a negative literal must appear in a positive literal in the body of the same rule), and leads to an all-or-nothing interpretation of negation, so as with the $\neg_an$ operator. We keep using $DTBS(\Pi, E)$ to denote the new DTB semantics, and we assume, from now on, that Datalog programs are stratified, range-restricted and safe. The following extensions of Theorem 3.1 and Proposition 3.1 can be obtained:
4. PROPERTIES OF PBB SEMANTICS

In [9], warded Datalog was introduced as a particularly well-behaved fragment of Datalog\(^\pm\), for which CQA is tractable (cf. [9] for details). Actually, in the journal version of [9] (currently under review), warded Datalog is extended with stratified negation, preserving the favorable properties of warded Datalog.

We can show that the Datalog\(^\pm\) program \(\Pi^+\) obtained from a Datalog or Datalog\(^\sim\) program \(\Pi\) by our construction in Section 3 is indeed warded:

**Theorem 4.1.** Let \(\Pi\) be a program in Datalog or Datalog\(^\sim\) and let \(\Pi^+\) be the Datalog\(^\pm\) program that represents its PBB semantics. Then \(\Pi^+\) is warded.

5. CONCLUSION AND FUTURE WORK

In this work, we have proposed and achieved a declarative and computable specification in Datalog\(^\pm\) of the multiset semantics of Datalog. Actually, we have extended this specification to Datalog\(^\sim\) thus capturing the full multiset relational algebra (MRA), including difference. The resulting Datalog\(^\pm\) programs have still good computational properties, and provably allow for tractable (conjunctive) query answering.

An immediate application of our results is to the evaluation of SPARQL with duplicates via the Datalog rewriting proposed in [4] as an intermediate step. This is illustrated by the following example:

**Example 5.1.** Consider the following SPARQL query from [TaskForce:PropertyPaths#Use_Cases]

```sparql
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE
{ ?x foaf:mbox <mailto:alice@example> ; ?x foaf:knows/foaf:name ?name . }
```

This query retrieves the names of people Alice knows. In our translation into Datalog, we omit the prefix and use the constant `alice` short for `mailto:alice@example` to keep the notation simple. Moreover, we assume that the RDF data over which the query has to be evaluated is given by a relational database with a single ternary predicate `t`, i.e., the database consists of triples \(t(X, Y, Z)\).

The following Datalog program with answer predicate `ans` is equivalent to the above SPARQL query:

\[
\begin{align*}
\text{ans}(N) & \leftarrow t(X, mbox, alice), \\
& \quad t(X, knows, Y), t(Y, name, N)
\end{align*}
\]

We now apply our transformation into Datalog\(^\pm\) from Section 3. Suppose that the query has to be evaluated over a multiset \(E\) of triples. Recall that in our transformation, we would first have to extend all triples in \(E\) to quadruples by adding a null (the tid) in the 0-position. Actually, this can be easily automatized by the first rule in the program below. We thus get the following Datalog\(^\pm\) program:

\[
\begin{align*}
\exists Z \ \text{ans}(Z; N) & \leftarrow t(Z_1; X, mbox, alice), \\
& \quad t(Z_2; X, knows, Y), \\
& \quad t(Z_3; Y, name, N) \\
\exists Z \ t(Z; U, V, V) & \leftarrow t(U, V, V)
\end{align*}
\]

We will work out in detail the application of our PBB semantics to the evaluation of SPARQL with bag semantics and give first experimental results with its implementation in the Vadalog system [5, 4] in the forthcoming full version of this paper. Recall that warded Datalog\(^\pm\) has been shown to be expressive enough to capture SPARQL under the OWL 2 QL core entailment regime [8], with set semantics though [2, 9]. We will tackle this extension of SPARQL with the OWL 2 QL core entailment regime under bag semantics with our transformation to warded Datalog\(^\pm\). Actually, the support of full SPARQL under OWL 2 entailment will also be profitable for other applications of the Vadalog system such as data wrangling [7] and data acquisition [12].

We are currently also working on inexpressibility results related to the transformation presented here: recall that our translation into Datalog\(^\pm\) does not cover multiset intersection \(\cap_m\); moreover, multiset difference is only handled in the all-or-nothing form \(\setminus_m\), while the sometimes more natural form \(\setminus_m\) has been left out. We conjecture that these two operations are not expressible in Datalog\(^\pm\) with set semantics. The verification of this conjecture is a matter of ongoing work.

6. REFERENCES

[1] R. Angles and C. Gutiérrez. The multiset semantics of SPARQL patterns. In Proc. ISWC 2016, volume 9981 of LNCS, pages 20–36, 2016.
[2] M. Arenas, G. Gottlob, and A. Pieris. Expressive languages for querying the semantic web. In Proc. PODS 14, pages 14–26. ACM, 2014.
[3] L. Bellomarini, G. Gottlob, A. Pieris, and E. Sallinger. Swift logic for big data and knowledge graphs. In Proc. JICAI 2017, pages 2–10. ijcai.org, 2017.
[4] L. Bellomarini, E. Sallinger, and G. Gottlob. The vadalog system: Datalog-based reasoning for knowledge graphs. PVLDB, 11(9):975–987, 2018.
[5] A. Cali, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. J. Artif. Intell. Res., 48:115–174, 2013.
[6] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. Theor. Comput. Sci., 336(1):89–124, 2005.
[7] T. Furche, G. Gottlob, B. Neumayr, and E. Sallinger. Data wrangling for big data: Towards a lingua franca for data wrangling. In Proc. AMW 2016, volume 1644 of CEUR Workshop Proceedings. CEUR-WS.org, 2016.
[8] B. Glimm, C. Ogbuji, S. Hawke, I. Herman, B. Parsia, A. Polleres, and A. Seaborne. SPARQL 1.1 Entailment Regimes. W3C Recommendation 21 march 2013, W3C, 2013. https://www.w3.org/TR/sparql11-entailment/.

[9] G. Gottlob and A. Pieris. Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue. In Proc. IJCAI 2015, pages 2999–3007. AAAI Press, 2015.

[10] D. S. Johnson and A. C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. J. Comput. Syst. Sci., 28(1):167–189, 1984.

[11] M. J. Maher and R. Ramakrishnan. Déjà vu in fixpoints of logic programs. In Proc. NAACL 1989, pages 963–980. MIT Press, 1989.

[12] C. Michels, R. R. Fayzrakhmanov, M. Ley, E. Sallinger, and R. Schenkel. Oxpath-based data acquisition for dblp. In Proc. JCDL 2017, pages 319–320. IEEE CS, 2017.

[13] I. S. Mumick, H. Pirahesh, and R. Ramakrishnan. The magic of duplicates and aggregates. In Proc. VLDB 1990, pages 264–277. Morgan Kaufmann, 1990.

[14] I. S. Mumick and O. Shmueli. Finiteness properties of database queries. In Proc. ADC ’93, pages 274–288. World Scientific, 1993.

[15] E. Sallinger. Reasoning about schema mappings. In Data Exchange, Information, and Streams, volume 5 of Dagstuhl Follow-Ups, pages 97–127. SD-LZI, 2013.