Space Bounds in Ontological Reasoning via Extended Bell Numbers

(DISCUSSION PAPER)

Cinzia Marte[0000−0003−3920−8186]

University of Calabria, Rende, Italy
marte@mat.unical.it

Abstract. In ontology-based query answering a user query is typically evaluated over instances containing both known and anonymous individuals. In this context, the algebraic notion of isomorphism is relevant in many application scenarios, such as the so-called parsimonious chase. Two atoms are isomorphic if there is a bijection among them that, in addition, is the identity on the known individuals. A naive upper bound on the maximum cardinality of any instance containing non-isomorphic atoms is well-known from the literature. However, there are cases in which this bound is far from being optimal. This paper generalizes Bell numbers to provide a tight bound in the above setting. The main result is also relevant in classical databases by characterizing the family of non-equivalent atomic queries.

Keywords: ontology-based query answering · existential rules · parsimonious chase · Bell numbers · atomic queries.

CEUR-WS.org/Vol-2400/paper-44.pdf

1 Introduction

Ontology-Based Query Answering (OBQA) consists in querying databases by taking ontological knowledge into account. It is a fascinating research topic deeply studied not only in database theory [1,13], but also in artificial intelligence [6,5,11] and in logic [2,3,12]. Moreover, OBQA is strictly related to others important application areas such as data integration [21], data exchange [9], and consistent query answering [23,24]. In particular, OBQA is the problem of answering a query q against a logical theory consisting of an extensional database D paired with an ontology Σ. The goal is to find certain answers to q, i.e. the query must be true in every possible model of the theory [20,7]. Here, we focus on ontologies expressed via existential rules, also known as tuple generating dependencies (TGDs) or $\mathsf{datalog}^{\exists}$ rules. They are at the core of Datalog$^{\pm}$ [14], an emerging family of ontology languages, which collects the basic decidable

Copyright © 2019 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors. SEBD 2019, June 16-19, 2019, Castiglione della Pescaia, Italy.

classes of TGDs, and generalizes several ontology specification languages such as Description Logics (DLs) [10]. Indeed, $\mathsf{datalog}^{\exists}$ generalizes the well-known language Datalog [16] with existential quantification in the head.

OBQA can be reduced to the problem of answering q over a universal model U that can be homomorphically embedded into every other model of the logical theory. A way to compute a universal model is to employ the so called chase procedure. Starting from D, the chase "repairs" violations of rules by repeatedly adding new atoms –introducing fresh values, called nulls, whenever required by an existential variable– until a fixed point satisfying all rules is reached. Therefore, in the classical setting, the chase is sound and complete. But, unfortunately, the chase does not always terminates [17,18].

Recently, in [22] a new class of $\mathsf{datalog}^{\exists}$ ontologies, called Shy, has been singled out for existential rules. It enjoys a new semantic property called *parsimony* and results in a powerful and decidable class that combines positive aspects of different Datalog$^{\pm}$ classes [4]. The parsimony property is based on the *parsimonious chase* (pchase) procedure that repairs violations of rules only if the (inferred) head atom can not be homomorphically mapped to any atom previously produced. For some classes of Datalog$^{\pm}$, the parsimony property is sound and complete with respect to atomic query answering. Moreover, the termination of the pchase is always guaranteed, and computational complexity has been studied [22]. Understanding the nature of the pchase procedure can lead to obtain practical improvements of existing implementations [22]. Indeed, so far, the research has been focused on establishing the termination of the pchase procedure, thus providing just a very rough upper bound of its maximal size, without any understanding of the relations between the logical theory and the atoms generated by the pchase. We fill this gap deepening this relevant connection. In particular, we need to understand what kind of atoms belong to the pchase. This has as side effect and it is strictly related to count the number of atoms generated by the pchase. An immediate consequence of this better understanding of the chase leads to re-prove computational complexity results, as we improve the upper bounds previously identified in [22]. Indeed, these are very large, as they are aimed at demonstrating computational complexity, and not how many atoms can be produced by the chase.

In this paper, we present contents already set out and published in [8]. In particular, we provide an exact upper bound for the pchase. To this end, we exploit the notion of "equality type" defined in [19], which we show to be strictly related to the form of non-isomorphic atoms of a given predicate. Then, by exploiting the notion of Bell numbers, counting the number of distinct partitions of a finite set, we compute an upper bound for the number of atoms generating by the pchase procedure and we show that there exists a family of ontologies for which the pchase can produce exactly the upper bound previously computed, so that it corresponds to the maximal number of atoms effectively generated by the pchase procedure.

Finally, as a corollary, our estimation of the maximum cardinality of any instance containing non-isomorphic atoms also provides a tight bound on the maximum number of non-equivalent atomic queries over a given relation, where two queries are considered *equivalent* if they give the same answers on every database.

2 Preliminaries

Throughout this paper we use the following notation. Let $\Delta = \Delta_C \cup \Delta_N \cup \Delta_V$ the domain of the *terms*, consisting of the union of the three countably infinite domains of *constants*, *nulls* and *variables*, respectively. We write φ to denote a null; X a variable; $\underline{a}$ an atom, that is an expression of the form $p(\mathbf{t})$, where $p = pred(\underline{a})$ is a predicate, $\mathbf{t} = t_1, \ldots, t_k$ is a *tuple* of terms, $k = arity(\underline{a})$ is the arity of $\underline{a}$ or p, and $\underline{a}[i]$ is the i-th term of $\underline{a}$. Moreover, $const(\underline{a})$ (resp., $vars(\underline{a})$) is the set of constants (resp., variables) occurring in $\underline{a}$. The set of predicates is denoted by $\mathcal{R}$. Let $T \subseteq \Delta$ a nonempty subset, then the set of all atoms that can be formed with predicates of $\mathcal{R}$ and terms from T is denoted by $base(T)$. Moreover, any subset of $base(\Delta_C \cup \Delta_N)$ constitutes an *instance* I, and whenever $I \subseteq base(\Delta_C)$, then it is also called *database*. A *substitution* is a total mapping $s : \Delta \to \Delta$. Let χ_1 and χ_2 be two structures containing atoms. An *homomorphism* $h : \chi_1 \to \chi_2$ is a substitution such that: (1) if $c \in \Delta_C$, then $h(c) = c$; (*ii*) if $\varphi \in \Delta_N$, then $h(\varphi) \in \Delta_C \cup \Delta_N$; (*iii*) $h(\chi_1)$ is a substructure of χ_2. An *existential rule* r is a logical implication of the form $\forall \mathbf{X} \forall \mathbf{Y} (\exists \mathbf{Z}\ \underline{a}(\mathbf{X}, \mathbf{Z}) \leftarrow \phi(\mathbf{X}, \mathbf{Y}))$, where $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ denote sets of variables; $head(r) = \underline{a}(\mathbf{X}, \mathbf{Z})$, while $body(r) = \phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms and can also be empty. We define a $\mathsf{datalog}^{\exists}$ program P as a finite set of existential rules, called ontology and denoted by $dep(P)$ (*dependencies* of P), paired with a database instance, denoted by $data(P)$. Moreover, $pred(P)$ (resp., $const(P)$) represents the set of predicates (resp., constants) occurring in (P) and $arity(P)$ is the maximum arity over $pred(P)$.

Given an instance I, we say that a rule r *is satisfied by* I if whenever there is a homomorphism $h : body(r) \to I$, there is a homomorphism $h' \supset h|_{vars(body(r))}$ s.t. $h' : head(r) \to I$. An instance I is a *model* of a program P if each rule of $dep(P)$ is satisfied by I, and $data(P) \subseteq I$. A *firing homomorphism* for r and I is any homomorphism $h : body(r) \to I$ s.t. $h = h|_{vars(body(r))}$. The *fire* of r via h produces the atom $fire(r, h) = \sigma(h(head(r)))$, where $\sigma = \sigma|_{vars(h(head(r)))}$ (i.e., it replaces each existential variable of r with a different fresh null). Given a firing homomorphism h for a rule r and an instance I, we say that the pair $\langle r, h \rangle$ satisfies the *parsimonious fire condition* w.r.t. an instance $I' \supseteq I$ if there is no homomorphism from $\{h(head(r))\}$ to I'. Finally, given a $\mathsf{datalog}^{\exists}$ program P, the *parsimonious chase* (pchase) of P ($pchase(P)$) is constructed as follows. We start from $I' = data(P)$ and create a copy of it in I. Then, for each r in $dep(P)$, for each unspent firing homomorphism h for the pair $\langle r, I \rangle$ we add the $fire(r, h)$ to I' if $\langle r, h \rangle$ satisfies the parsimonious fire condition w.r.t. I'. If $I \neq I'$, we create a new copy of I' and repeat the previous steps. Otherwise, we return I.

3 Parsimonious Chase Estimation

In this section, we introduce some basic notions that will help us to find a tight upper bound for the pchase. We highlight a main property of the pchase, based on isomorphic atoms, a crucial notion in several Datalog$^{\pm}$ classes [15].

Theorem 1. *Given a program P, $pchase(P)$ does not contain isomorphic atoms.*

Proof. Assume, by contradiction, that there are two isomorphic atoms $\underline{a}$ and $\underline{a}'$ in $pchase(P)$. Thus, there is a homomorphism h from $\{\underline{a}\}$ to $\{\underline{a}'\}$ s.t. h^{-1} is a homomorphism from $\{\underline{a}'\}$ to $\{\underline{a}\}$. W.l.o.g. assume that $\underline{a} \in I$, for some I generated during the pchase procedure. As $\underline{a}' \in pchase(P)$, then there is a rule r, an instance $I' \supseteq I$, and an unspent firing homomorphism h' for $\langle r, I' \rangle$, s.t. $fire(r, h') = \underline{a}'$, against the fact that $h^{-1} \circ \sigma$ is a homomorphism from $\{h'(head(r))\}$ to I'. Indeed, $(h^{-1} \circ \sigma)(h'(head(r)) = h^{-1}(\sigma(h'(head(r)) = h^{-1}(fire(r, h')) = h^{-1}(\underline{a}') = \underline{a} \in I \subseteq I'$. □

To provide a precise upper bound for the number of steps execute by the pchase, we introduce the concept of *type* that is equivalent to the notion of equality type defined in.

Definition 1 (Type). *Let m be a positive integer, S an arbitrary partition of $\{1, \ldots, m\}$, C a set with $|C| \leq |S|$, and $f : C \to S$ an injective map. We define the* type *of S, C and f as the family of sets $\mathcal{T}(S, C, f) = \big\{s \cup f^{-1}(s) \mid s \in S\big\}$.*

Example 1. Let $m = 6$, $C = \{c_1, c_2\}$, and let $S = \big\{\{1, 2\}, \{3, 6\}, \{4\}, \{5\}\big\}$ be a partition of $\{1, \ldots, 6\}$. Consider the injective map $f : C \to S$ such that $f(c_1) = \{3, 6\}$ and $f(c_2) = \{5\}$. Then, $\mathcal{T}(S, C, f) = \big\{\{1, 2\}, \{3, 6, c_1\}, \{4\}, \{5, c_2\}\big\}$.

Fixed an integer m, our aim is to count the number of all possible types that can be generated from any partition of the set $\{1, \ldots, m\}$, by varying C on a superset D of a fixed size d. In order to do this, we resort to the Bell number B_n, that is the number of ways to partition a set of n labeled elements.

Theorem 2. *Let $m \in \mathbb{N}$, D a finite set of size $d > 0$, and B_n the n-th Bell number. Hence, the number of all possible types generated from all the partitions of the set $\{1, \ldots, m\}$ and all subsets of D is given by $\gamma_m^d = \sum_{h=0}^{m} \binom{m}{h} d^h B_{m-h}$.*

Proof Sketch. Recall that, given two sets A and B with $|A| = \alpha \leq |B| = \beta$, the number of injective maps from A to B is $\frac{\beta!}{(\beta-\alpha)!}$. Then, fixed a partition S of $\{1, ..., m\}$ with $|S| = s$, the number of injective maps from any subset $C \subseteq D$ to S, with $|C| = c \leq s$, is $\frac{s!}{(s-c)!}$, while the number of subsets of size c is $\binom{d}{c}$. Thus, the number of all possible types for the fixed partition S is $\sum_{c=0}^{min\{s,d\}} \binom{d}{c} \cdot \frac{s!}{(s-c)!}$. Hence, the number of types generated from all the partitions of the set $\{1, \ldots, m\}$ and all subsets of D is given by $\sum_{s=1}^{m} S(m, s) \cdot \sum_{c=0}^{min\{s,d\}} \binom{d}{c} \cdot \frac{s!}{(s-c)!}$, where $S(m, s)$

is the Stirling number counting the number of partitions of size s on m elements. It can be shown that it is equivalent to γ_m^d. □

Taking advantage of the notion of type, we can provide a new representation of an arbitrary atom.

Definition 2 (Atom Type). *Given an atom $\underline{a} = p(\boldsymbol{t})$ of arity m, we define the type of the atom $\underline{a}$ as $T_{\underline{a}} = \mathcal{T}(S, C, f)$, where $C = const(\underline{a})$; $S = \big\{\{n \mid \underline{a}[n] = t_i\} \mid \ i = 1, \ldots, m\big\}$; and $f : C \to S$ such that $f(c) = \{n \mid \underline{a}[n] = c\}$.*

Hence, the type of an atom $\underline{a}$ has the form $T_{\underline{a}} = \{\sigma(t_1), \ldots, \sigma(t_m)\}$, where σ is such that $\sigma(t_i) = \{n \mid n \in \{1, \ldots, m\} \wedge \underline{a}[n] = t_i\} \cup \{t_i\}$ if t_i is a constant, and $\sigma(t_i) = \{n \mid n \in \{1, \ldots, m\} \wedge \underline{a}[n] = t_i\}$ otherwise. Intuitively, the type of an atom is formed by the sets of positions where a term occurs, by highlighting positions where constants occur.

Example 2. Let $\underline{a} = p_1(\varphi_1, \varphi_3, \varphi_2, \varphi_1)$ and $\underline{b} = p_2(c, \varphi_1, d, c, \varphi_2, \varphi_2, \varphi_1)$. Then, $T_{\underline{a}} = \big\{\{1, 4\}, \{2\}, \{3\}\big\}$ and $T_{\underline{b}} = \big\{\{1, 4, c\},\ \{2, 7\},\ \{3, d\}, \{5, 6\}\big\}$.

Theorem 3. *Let $\underline{a} = p(t_1, \ldots, t_k)$ and $\underline{a}' = p(t'_1, \ldots, t'_k)$ be two atoms. Then, $\underline{a}$ and $\underline{a}'$ are isomorphic if, and only if, $pred(\underline{a}) = pred(\underline{a}')$ and $T_{\underline{a}} = T_{\underline{a}'}$.*

Proof. Let us consider two atoms $\underline{a}$ and $\underline{a}'$. If $pred(\underline{a}) \neq pred(\underline{a}')$ or $arity(\underline{a}) \neq arity(\underline{a}')$, then of course can not exists an isomorphism between them. Hence, we can take for granted that the two atoms have same predicate and arity.
[⇒] Assume that there is an isomorphism between $\underline{a}$ and $\underline{a}'$, i.e., there is a homomorphism $h : \{\underline{a}\} \to \{\underline{a}'\}$ s.t. $h(t_i) = t'_i,\ i = 1, \ldots, k$ and s.t. $h^{-1} : \{\underline{a}'\} \to \{\underline{a}\}$ is a homomorphism. Let $T_{\underline{a}} = \{\sigma(t_1), \ldots, \sigma(t_k)\}$ and $T_{\underline{a}'} = \{\sigma'(t_1), \ldots, \sigma'(t_k)\}$. We claim that $\sigma(t_i) = \sigma(t'_i)$, for $i = 1, \ldots, k$. Assume that $\sigma(t_i) \subseteq \sigma(t'_i)$, and let $n \in \sigma(t_i) \cap \mathbb{N}$, so that $\underline{a}[n] = t_i$. Therefore, we have that $t'_i = h(t_i) = h(\underline{a}[n]) = h(\underline{a})[n] = \underline{a}'[n]$. Hence, $n \in \sigma(t'_i)$. Moreover, if $n = c$ is a constant, by definition of homomorphism, we have $c \in \sigma(t_i) \Rightarrow t_i = c \Rightarrow t'_i = h(t_i) = t_i = c \Rightarrow c \in \sigma(t'_i)$. The reverse inclusion can be easily proved by replacing h by h^{-1}.
[⇐] Let us assume that $\mathcal{T}_{\underline{a}} = \mathcal{T}_{\underline{a}'}$. Let $h : \{\underline{a}\} \to \{\underline{a}'\}$ be s.t. $h(t_i) = t'_i$. First, we prove that h is a homomorphism. Let $t_i = c$ be a constant. Suppose that $c \in \sigma(t_i)$, then by assumption $c \in \sigma(t'_i)$, hence $t'_i = c$. It remains to be shown that h is also injective. Let $t'_i = t'_j$. Then, $\sigma(t'_i) = \sigma(t'_j) \Rightarrow \sigma(t_i) = \sigma(t_j) \Rightarrow t_i = t_j$. □

Now, we are able to provide an upper bound for the maximum number of atoms generating by the pchase procedure.

Theorem 4. *Let P be a program with $arity(P) = w$, $|const(P)| = d$, and l_m the number of predicates in $pred(P)$ of arity m. Then, $|pchase(P)| \leq \sum_{m=0}^{w} l_m \gamma_m^d$.*

Proof. By Theorem 2 and Theorem 3, the total number of non isomorphic atoms over $pred(P)$ and $const(P) \cup \Delta_N$ is given by $\sum_{m=0}^{w} l_m \gamma_m^d$. Moreover, by Theorem 1, we know that $pchase(P)$ does not contain isomorphic atoms. Hence, $|pchase(P)| \leq \sum_{m=0}^{w} l_m \gamma_m^d$. □

Let Γ_w^d be the upper bound in Theorem 4. To show that it is also tight, we introduce an ordering on types that will allow us to build a program with a sequence of firing homomorphisms generating a pchase of size exactly Γ_w^d.

Definition 3 (Type ordering). *Let $T = \mathcal{T}(S, C, f)$ and $T' = \mathcal{T}(S', C', f')$. Then, T* precedes *T', if (i) $|C| < |C'|$, or (ii) $|C| = |C'|$ and $|S| > |S'|$.*

Intuitively, such a program should have a rule for each possible atom tag, whenever constants are allowed in the rules. Otherwise, we need a predicate to collect all constants of the database. To better understand our idea, we give an example of such a program before we provide the formal result.

Example 3. Let C be a finite set of constant, and P be a program such that $data(P) = \{t(c_1), t(c_2)\}$, and $dep(P)$ is given by

$$\begin{array}{rlll}
\exists X,Y,Z p(X,Y,Z) & \exists X,Y p(Z,X,Y) \leftarrow t(Z) & \exists X p(X,Y,Z) \leftarrow t(Y), t(Z) \\
\exists X,Y p(X,X,Y) & \exists X,Y p(X,Z,Y) \leftarrow t(Z) & \exists X p(Y,X,Z) \leftarrow t(Y), t(Z) \\
\exists X,Y p(X,Y,X) & \mathbf{\exists X,Y p(X,Y,Z) \leftarrow t(Z)} & \exists X p(Y,Z,X) \leftarrow t(Y), t(Z) \\
\exists X,Y p(X,Y,Y) & \exists X p(X,X,Y) \leftarrow t(Y) & p(X,Y,Z) \leftarrow t(X), t(Y), t(Z) \\
\exists X p(X,X,X) & \exists X p(X,Y,X) \leftarrow t(Y) & \\
 & \exists X p(Y,X,X) \leftarrow t(Y) & \exists X t(X)
\end{array}$$

We build $pchase(P)$ by starting from rules in the first column from top to bottom. For each rule r in this ordering, we consider all firing homomorphism h for r. E.g., the rule in bold produces the atoms $\{p(\varphi_1, \varphi_2, c_1), p(\varphi_3, \varphi_4, c_2)\}$. Thus, the number of atoms with predicate p generated by the pchase will be $37 = \gamma_3^2$, and $|pchase(P)| = 40 = \gamma_3^2 + \gamma_1^2$.

Theorem 5. *Let w be a positive integer, D a set of constants of size d, and Γ_w^d as above. Then, there is a family P_w of programs s.t. $|pchase(P_w)| = \Gamma_w^d$.*

Proof Sketch. We build a program P_w having two predicates p (of arity w) and t (of arity 1). We set $data(P_w) = \{t(c) \mid c \in D\}$, and define $dep(P_w)$ as follows. Given a partition $S_i = \{\Lambda_1, \ldots, \Lambda_n\}$ of w, where $n = |S_i|$, we construct a rule r_i with an empty body, by adding $X_1, \ldots, X_n$ existential variables so that $\Lambda_j = \{k \mid p[k] = X_j,\ k \in [w]\}$. Now, fixed a rule r_i with $n > 1$ existential variables, we produce $n - 1$ blocks of rules as follows. We translate j existential variables into universal ones, by adding j atoms over predicate t in the body. Hence, we construct $\binom{n}{j}$ rules. Then, we add the rules $p(X_1, \ldots, X_w) = t(X_1), \ldots, t(X_w)$, and $\exists X t(X)$. Finally, we remove all rules having in the head more than one repeated universal variable. To prove that $|pchase(P_w)| = \Gamma_w^d$, we provide a sequence of $\Gamma_w^d - d$ firing homomorphisms. To each rule r in $dep(P_w)$ we associate uniquely an atom $g(head(r))$, where g maps existential variables to fresh nulls, and universal variables to a fixed constant. The type ordering on the atoms gives an ordering on the rules, and so to the sequence of firing homomorphisms. □

4 Discussion and Future Work

In this work, we identified the maximal number of distinct atoms generable by the pchase procedure. In particular, γ_m^d improves the bound given in [22], that is $(d+m)^m$. In particular, $d^m \leq \gamma_m^d \leq (d+m)^m$. Since in the OBQA context, normally, d is much bigger than m, it could seem that the effort to find such a precise upper bound can be useless for practical purposes. However, this is not the case, as shown in the paper. Indeed, the search for a precise upper bound led to identify the fundamental notions of type and type ordering that highlighted some qualitative characteristics of the pchase. Moreover, there could be other contexts where m is much bigger than d (think for example to scenarios where tuples encode strings over a certain alphabet, as in complexity proofs based on Turing Machine simulation). In this cases, our bound represents a concrete improvement. As future work, we plan to extend the pchase condition to rules with a complex head, and to compute the maximal number of distinct atoms generable in this case. Then, we will try to analyze the orderings of fire homomorphisms in the generation of the pchase to understand if we can identify a sort of best ordering that minimizes the number of atoms produced. Finally, we will try to apply this metodology to give exact estimations of others chase versions.

References

1. Alviano, M., Pieris, A.: Default negation for non-guarded existential rules. In: Proc of PODS (2015)
2. Amendola, G., Leone, N., Manna, M.: Finite model reasoning over existential rules. TPLP **17**(5-6), 726–743 (2017)
3. Amendola, G., Leone, N., Manna, M.: Querying finite or arbitrary models? no matter! existential rules may rely on both once again (discussion paper). In: SEBD. CEUR Workshop Proceedings, vol. 2037, p. 218. CEUR-WS.org (2017)
4. Amendola, G., Leone, N., Manna, M.: Finite controllability of conjunctive query answering with existential rules: Two steps forward. In: IJCAI. pp. 5189–5193. ijcai.org (2018)
5. Amendola, G., Leone, N., Manna, M., Veltri, P.: Reasoning on anonymity in datalog+/-. In: ICLP (Technical Communications). OASICS, vol. 58, pp. 3:1–3:5. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
6. Amendola, G., Leone, N., Manna, M., Veltri, P.: Enhancing existential rules by closed-world variables. In: IJCAI. pp. 1676–1682. ijcai.org (2018)
7. Amendola, G., Libkin, L.: Explainable certain answers. In: IJCAI. pp. 1683–1690. ijcai.org (2018)
8. Amendola, G., Marte, C.: Extending bell numbers for parsimonious chase estimation. In: JELIA. Lecture Notes in Computer Science, vol. 11468, pp. 490–497. Springer (2019)
9. Arenas, M., Barceló, P., Libkin, L., Murlak, F.: Foundations of Data Exchange. Cambridge University Press (2014)
10. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2003)

11. Baget, J., Leclère, M., Mugnier, M., Salvat, E.: On rules with existential variables: Walking the decidability line. Artif. Intell. **175**(9-10), 1620–1654 (2011)
12. Bárány, V., Gottlob, G., Martin Otto: Querying the guarded fragment. LMCS **10**(2) (2014)
13. Bourhis, P., Manna, M., Morak, M., Pieris, A.: Guarded-based disjunctive tuple-generating dependencies. ACM TODS **41**(4) (2016)
14. Calì, A., Gottlob, G., Lukasiewicz, T.: Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In: Proc. of ICDT (2009)
15. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: PODS. pp. 77–86. ACM (2009)
16. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. **1**(1), 146–166 (1989)
17. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Proc. of PODS (2008)
18. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. TCS **336**(1), 89–124 (2005)
19. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: ICDE. pp. 2–13. IEEE Computer Society (2011)
20. Imielinski, T., Lipski, W.: Incomplete information in relational databases. J. ACM **31**(4), 761–791 (1984)
21. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS. pp. 233–246. ACM (2002)
22. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable Datalog$^{\exists}$ programs. In: Proc. of KR (2012)
23. Manna, M., Ricca, F., Terracina, G.: Consistent query answering via ASP from different perspectives: Theory and practice. TPLP **13**(2), 227–252 (2013)
24. Manna, M., Ricca, F., Terracina, G.: Taming primary key violations to query large inconsistent data via ASP. TPLP **15**(4-5), 696–710 (2015)