JOURNAL OF COMPUTER AND SYSTEM SCIENCES 55, 355–369 (1997)
ARTICLE NO. SS971528

Can Datalog Be Approximated?

Surajit Chaudhuri

Microsoft Research, One Microsoft Way, Redmond, Washington 98052

and

Phokion G. Kolaitis*

Computer Science Department, University of California, Santa Cruz, California 95064

Received April 15, 1995; revised August 24, 1995

In this paper, we investigate whether recursive Datalog predicates can be approximated by finite unions of conjunctive queries. We introduce a quantitative notion of error and examine two types of approximation, namely, *absolute approximation* and *relative approximation*. We also stipulate that the approximations obey certain qualitative criteria, namely we require them to be *upper envelopes* or *lower envelopes* of the Datalog predicate they approximate. We establish that *absolute approximation* by finite unions of conjunctive queries is not possible, which means that no unbounded Datalog predicate can be approximated by a finite union of conjunctive queries in such a way that the error is bounded uniformly by the same constant on all finite databases. After this, we examine *relative approximations*, i.e., approximations that guarantee bounds for the error relative to the size of the Datalog predicate under consideration. Although such approximations exist in some cases, we show that for several large and well-studied classes of unbounded Datalog predicates it is not possible to find finite unions of conjunctive queries that satisfy the aforementioned qualitative criteria and have the property that the relative error of the approximation is bounded by a constant. Finally, we consider first-order approximations and obtain sharp negative results for the approximability of the *transitive closure* query by first-order queries.

1. INTRODUCTION

The optimization problem for relational algebra queries has been studied extensively and some of the techniques developed for solving this problem have been incorporated in commercial relational products. In particular, efficient evaluation techniques exist for the class of Select-Project-Join queries, a class that contains all conjunctive queries. On the other hand, relational algebra has rather limited expressive power, since it does not have a built-in recursion mechanism. To remedy this situation, the query language Datalog and its extensions were introduced and studied in depth during the past decade (see [U89]). Datalog has augmented expressive power that makes it possible to express queries, such as *transitive closure*, that cannot be expressed in relational algebra or, equivalently, in first-order logic. Naturally, a great deal of attention has been devoted to optimization of Datalog programs. In general, Datalog queries are intrinsically harder to compute than first-order queries, since Datalog can express queries, such as *path systems*, that are complete for polynomial time [Co74], while all first-order queries are computable in logarithmic space. At first sight, it appears promising to identify Datalog programs that give rise to first-order queries, so that known optimization techniques from relational algebra can be applied. It is clear, however, that this approach cannot be applied to interesting Datalog programs, since "true" recursive Datalog queries are not first-order expressible.

Our goal in this paper is to investigate whether Datalog queries can be *approximated* in a reasonable way by "simpler" queries, such as finite unions of conjunctive queries or first-order queries. Although such a prospect is certainly attractive, several issues have to be addressed and resolved in order to establish the viability of this approach to optimization. First, one must formalize the concept of *approximation* and spell out what makes an approximation a "good" one. Next, one must present evidence that such approximations will indeed be useful. Finally, and perhaps more importantly, one must determine whether "good" approximations of Datalog exist in the first place and, if yes, how they can be found. We now examine each of these issues separately.

What is a "good" approximation of a Datalog query? It is clear that the approximating objects ought to be efficiently computable and "simpler" than the object they approximate. Since every Datalog query can be viewed as an infinite union of conjunctive queries, the most natural choice is to take finite unions of conjunctive queries as approximating objects. Additional conditions are needed,

* Part of the research on this paper was carried out while this author was visiting the Computer Science Department of Stanford University supported by a 1993 John Simon Guggenheim Fellowship. Research also partially supported by NSF Grants CCR-9108631 and CCR-9307758. E-mail: kolaitis@cse.ucsc.edu.

0022-0000/97 $25.00

however, in order to ensure that the approximating objects constitute "reasonable" approximations. These conditions should reflect qualitative or quantitative aspects of the relationship between the approximating objects and the objects of approximation. On the qualitative side, one could use the concept of *containment* between queries and require that over any database the approximating object returns a superset (or, a subset) of the collection of tuples that the recursive predicate evaluates to. We refer to such approximating objects as *upper envelopes* (respectively, *lower envelopes*). Since containment induces a partial-order among approximating objects, we can define the notion of *minimal* upper and lower envelopes, a notion that was investigated in [C93]. Unfortunately, qualitative properties alone, such as containment and minimality, do not guarantee that approximations are reasonable. Indeed, as shown in [C93], there are Datalog programs for which the size of every minimal upper envelope differs from that of the recursive predicate by an order of magnitude in the size of the database. Such approximations are clearly unreasonable.

In this paper, we introduce a quantitative concept of *error* in order to compare the approximating object with the object of approximation. To this effect, we define the *error* in approximating a Datalog query by some other query to be the cardinality of the symmetric difference of these two queries. Using this concept of error, we can formalize the concept of "good" approximation and define two different types of approximations. We first introduce *absolute approximations*, i.e., approximations guaranteeing that the error is bounded uniformly by the same constant on all finite databases. Since such guarantees are quite ambitious and often infeasible, we relax this stringent requirement and consider also *relative approximations*, i.e., approximations that ensure bounds for the error relative to the size of the Datalog predicate. These concepts are in direct analogy with concepts of error and approximation in combinatorial optimization, where researchers have investigated extensively the existence of efficient approximation algorithms for intractable maximization and minimization problems (see [GJ79; PS82]).

Let us now discuss some possible applications of approximations. In decision-support scenarios, where the queries are asked over large databases and are frequently modified (e.g., applications used in stock markets), approximations to the answer provide the user with feedback that can be used to modify the query. Approximations are also valuable in the context of efficiently evaluating queries, provided the use of approximation does not affect the answer to the query. For example, an upper envelope can be used to obtain approximations to recursive predicates that restrict sideways information passing in various rewriting schemes. Further applications of upper envelopes are presented in [C93].

We are now ready to turn to the task of determining whether or not "good" approximations of Datalog predicates exist. Of course, we know that perfect approximations do exist for predicates in *bounded* Datalog programs, since these are exactly the Datalog predicates that are equivalent to finite unions of conjunctive queries [Io86; Na89]. Can we then approximate *unbounded* Datalog predicates by finite unions of conjunctive queries? We show here that if a Datalog predicate is unbounded, then absolute approximations of this nature do not exist. As a matter of fact, we establish a stronger result, namely, the error introduced in approximating a truly recursive predicate by a finite union of conjunctive queries cannot be bounded by any "reasonable" quantity that depends only on the size of the database under consideration. Moreover, in this result the approximations need not be upper envelopes or lower envelopes of the predicate they approximate. In view of this negative result, we examine relative approximations. Although such approximations exist in some cases, we show that for several large and well-studied classes of unbounded Datalog predicates it is not possible to find relative approximations that are finite unions of conjunctive queries and satisfy the qualitative criteria of being an upper envelope or lower envelope. In particular, these nonapproximability properties are shared by all unbounded monadic predicates and all unbounded predicates in chain programs. After this, we take a closer look at unbounded Datalog predicates that possess relative approximations and investigate the question of whether the error of approximation can be made arbitrarily small. We establish that for a large class of such predicates there is a threshold on the error of relative approximations. Finally, we entertain the possibility of using arbitrary first-order queries as approximating objects. In particular, we focus on the *transitive closure* query TC, one of the most thoroughly analyzed Datalog queries, and show that TC does not have any "reasonable" relative first-order approximations. This result constitutes a strong generalization of the well-known theorem asserting that TC is not a first-order definable query [AU79; Fa75].

The results reported in this paper are established using homomorphism techniques and Ehrenfeucht–Fraïssé games. In the past, these methods have been used widely in database theory and finite model theory respectively in order to obtain results of qualitative character, including an analysis of containment and equivalence between conjunctive queries, and lower bounds for first-order expressibility. The technical innovation developed here is the systematic use of these tools in obtaining results of quantitative flavor that strengthen considerably their qualitative counterparts.

2. PRELIMINARIES

A *vocabulary* σ is a finite set of relation symbols. If n is a positive integer, then a *n-ary query* Q on σ is a function defined on databases over σ and such that for every database D over σ the value $Q(D)$ of the query Q on D is a

n-ary relation on D that is preserved under isomorphisms. A *Boolean query* Q on σ is a collection of databases over σ that is closed under isomorphisms. Equivalently, a Boolean query is a function Q that is defined on databases over σ, takes 0 or 1 as values, and has the property that if D_1 and D_2 are two isomorphic databases over σ, then $Q(D_1) = Q(D_2)$.

A *conjunctive query* is a query definable by a positive existential first-order formula having conjunction as its only Boolean connective, i.e., by a formula of the form

$$(\exists z_1) \cdots (\exists z_m)\, \psi(x_1, ..., x_n, z_1, ..., z_m),$$

where $\psi(x_1, ..., x_n, z_1, ..., z_m)$ is a conjunction of atomic formulas over the vocabulary σ. The free variables $x_1, ..., x_n$ in the formula defining the query Q are called the *distinguished variables* of Q. For example, the formula $(\exists z)(P(x, z) \wedge R(z, y))$ defines a conjunctive query Q with x and y as its distinguished variables. We will often represent conjunctive queries using logic programming notation, in particular the above query can be written as: $Q(x, y) : -P(x, z), R(z, y)$. A *union of conjunctive queries* is a query definable by a (possibly infinite) disjunction of positive existential formulas having the same free variables.

Let Q and Q' be two n-ary queries over a vocabulary σ. We say that Q is *equivalent* to Q', and write $Q \equiv Q'$, if $Q(D) = Q'(D)$ on every finite database D over σ. We say that Q is *contained* in Q', and write $Q \subseteq Q'$, if on every finite database D over σ we have that $Q(D)$ is a subset of $Q'(D)$.

The containment relation between conjunctive queries was analyzed by Chandra and Merlin [CM77], while the containment relation between unions of conjunctive queries was studied by Sagiv and Yannakakis [SY81]. In particular, we have the following result from [SY81].

THEOREM 2.1. *Let $Q = \bigcup_{i \in I} Q_i$ and $Q' = \bigcup_{j \in J} Q'_j$ be two queries that are the unions (finite or infinite) of the conjunctive queries Q_i, $i \in I$, and Q'_j, $j \in J$, respectively. Then $Q \subseteq Q'$ if and only if for each $i \in I$ there is some $j \in J$ such that $Q_i \subseteq Q'_j$.*

We assume familiarity with the Datalog query language (see [U89]). A *Datalog program* π is a finite set of definite Horn clauses without function symbols. The predicates that occur in the head of the rules are called IDB predicates (*intentional database predicates*), while the rest are called EDB predicates (*extensional database predicates*). Datalog programs can be given both declarative and operational semantics. On every database D, a Datalog program π gives rise to a monotone operator from a sequence of values for the IDB predicates to another sequence of values for these predicates obtained by applying once the rules of the program. The declarative semantics of π on the database D is defined as the *least fixpoint* of this operator on D and consists of a sequence of relations $P^\infty(D)$ on D, one for each IDB predicate P. It is well known that this declarative semantics is equivalent to an operational semantics obtained by applying the rules of the program repeatedly until the least fixpoint is reached. More specifically, for each IDB predicate P we have that $P^\infty(D) = \bigcup_{i=1}^{\infty} P^i(D)$, where $P^i(D)$ is the ith *stage of* P in the "bottom-up" evaluation of π on D, i.e., the value for the IDB predicate P obtained by applying at most i times the rules of the program on D. It is easy to verify that the stages of P form a monotone sequence of queries, that is to say, if $i \leqslant j$, then $P^i \subseteq P^j$. Moreover, for each $i \geqslant 1$ the query P^i is a finite union of conjunctive queries that are called the ith *level expansions* of P. As an example, consider the following Datalog program:

$$P(x, y) : -E(x, z), P(z, y)$$
$$P(x, y) : -R(x), T(x, y)$$
$$P(x, y) : -E'(x, y)$$
$$T(x, y) : -S(x, y).$$

For this program, $P^1(x, y) \equiv E'(x, y)$; thus, $E'(x, y)$ is the only 1-level expansion of P. On the other hand, P has three 2-level expansions, since

$$P^2(x, y) \equiv E'(x, y) \vee E(x, z), E'(z, y) \vee R(x), S(x, y).$$

Let π be a Datalog program and let P be one of its IDB predicates. If D is a finite database, then there is a smallest integer $m \geqslant 1$ such that $P^\infty(D) = P^m(D)$ and, consequently, $P^m(D) = P^{m'}(D)$ for all $m' \geqslant m$. In general, this integer m depends on D and may increase beyond any bounds as the database varies. If this is the case, then we say that the predicate P is *unbounded* in π, otherwise we say that P is *bounded* in π. More precisely, a Datalog predicate P is *bounded* in a program π if there exists a positive integer $m \geqslant 1$ such that on every finite database D

$$P^\infty(D) = P^m(D).$$

Using structural and preservation properties of positive existential formulas, one can show that if $P^\infty(D) = P^m(D)$ on all finite databases D, then $P^\infty(D) = P^m(D)$ on all infinite databases D. Thus, although boundedness is formally defined with respect to finite databases, it is robust enough to extend to infinite databases as well.

The concept of boundedness was introduced and studied first in the context of the universal relation database model by Maier, Ullman, Vardi [MUV84] and then in the context of Datalog by Ioannidis [Io86] and Naughton [Na89]. Since that time, boundedness has been investigated in depth from both an algorithmic and a structural standpoint. On the algorithmic side, Gaifman *et al.* [GMSV87] showed that it is an undecidable problem to determine whether a given IDB predicate P is bounded. On the structural side,

by applying the characterization of containment between unions of conjunctive queries in [SY81], it is not hard to show that a predicate P is bounded in a Datalog program π if and only if there is a query Q such that Q is a finite union of conjunctive queries and P^∞ is equivalent to Q, i.e., $P^\infty(D) = Q(D)$ on all finite databases. Ajtai and Gurevich [AG94] extended significantly this result by establishing a deeper characterization of boundedness, which asserts that a predicate P is bounded in a Datalog program π if and only if there is a first-order definable query Q such that P^∞ is equivalent to Q.

3. APPROXIMATION AND ERROR

Let P be a predicate that is unbounded in a Datalog program π and suppose that, instead of evaluating P^∞, we are only interested in "approximating" P^∞ by a "simpler" or "easier" query Q. There are two separate issues that have to be addressed in order to formalize this idea and extract a precise concept from it, namely, we must decide on what queries to use as approximations of unbounded Datalog predicates and on criteria for the "goodness" of the approximations.

What makes a query "simpler" or "easier" than an unbounded Datalog query? If we have efficient evaluation methods in mind, the most natural choice is to consider finite unions of conjunctive queries as approximating objects. Since P^∞ is an infinite union of conjunctive queries, this amounts to "approximating" P^∞ by finite unions of similar building blocks. Later on, we will relax this requirement and consider arbitrary first-order queries as approximating objects; i.e., we will attempt to approximate Datalog queries by queries expressible in relational algebra.

The second issue is to formulate quantitative and qualitative criteria that reflect the "goodness" of the approximation. If a query Q is used to approximate P^∞, then on a given database D the value $P^\infty(D)$ of the Datalog predicate may differ from the value $Q(D)$ of the query in that $P^\infty(D)$ contains tuples that are not in $Q(D)$ and vice versa. This difference between P^∞ and Q can be measured by utilizing the following quantitative concept of *error*.

DEFINITION 3.1. Let P be an IDB predicate in a Datalog program π and let Q be a query of the same arity as P. The *error* $\delta(P, Q)(D)$ *in approximating* P^∞ *by* Q *over a database* D is

$$\delta(P, Q)(D) = |P^\infty(D) - Q(D)| + |Q(D) - P^\infty(D)|.$$

Thus, the error is the sum of the cardinalities of the two one-sided errors. Equivalently, the error $\delta(P, Q)(D)$ is equal to the cardinality $|P^\infty(D) \triangle Q(D)|$ of the symmetric difference $P^\infty(D) \triangle Q(D)$. Notice that we can define the error $\delta(Q^*, Q)(D) = |Q^*(D) \triangle Q(D)|$ for any two queries Q^* and Q of the same arity, in other words Q^* need not be of the form P^∞ for some Datalog predicate P.

In addition to meeting quantitative criteria, it is natural to require that the approximations obey also certain semantic restrictions that capture qualitative aspects of the relation between approximating objects and the objects of approximation [C93].

DEFINITION 3.2. Let P be an IDB predicate in a Datalog program π and let Q be a query of the same arity as P. We say that Q is an *upper envelope* of P if $P^\infty \subseteq Q$. Similarly, we say that Q is a *lower envelope* of P if $Q \subseteq P^\infty$.

Upper and lower envelopes are in many respects analogous to feasible solutions of combinatorial optimization problems. In the case of minimization problems, the value of the objective function on a feasible solution is bigger than or equal to the optimum, while in the case of maximization problems this value is less than or equal to the optimum. Upper envelopes approximate unbounded Datalog predicates "from above," whereas lower envelopes approximate such predicates "from below." Thus, upper envelopes can be viewed as feasible solutions of minimization problems and lower envelopes as feasible solutions of maximization problems.

Notice that there is an abundance of lower envelopes that are finite unions of conjunctive queries, in particular each stage P^i, $i \geqslant 1$, is such a lower envelope of P. Observe also that if a query Q is an upper envelope or a lower envelope of a Datalog predicate P, then the (two-sided) error $\delta(P, Q)(D)$ introduced above reduces to one of the two one-sided errors, since $\delta(P, Q)(D) = |Q(D) - P^\infty(D)|$ for an upper envelope Q, while $\delta(P, Q)(D) = |P^\infty(D) - Q(D)|$ for a lower envelope Q.

In the next two sections, we will investigate the existence of upper and lower envelopes satisfying certain quantitative performance guarantees that are obtained by imposing bounds on the error.

4. ABSOLUTE APPROXIMATION

The most desirable quantitative performance guarantee of an approximation is for it to be an *absolute approximation*, which means that it induces an error that is bounded by a fixed constant uniformly on all finite databases. In this section, we show that bounded programs constitute the only class of Datalog programs for which we can find absolute approximations that are unions of conjunctive queries. As a matter of fact, a considerably stronger result holds, namely, if the bound on the error is some "reasonable" function of the size of the database, then the program must be bounded. Moreover, in this result the approximations are not required to satisfy any qualitative criteria, such as being an upper envelope or a lower envelope of the predicate they approximate.

DEFINITION 4.1. A function g from positive numbers to non-negative numbers is *sublinear* if for every positive integer l there exists a positive integer n such that $g(ln) < n$.

This definition is, perhaps, better grasped by considering the class of functions that fail to satisfy it. If a function g is not sublinear, then there is an integer l such that for all multiples n of l we have that $g(n) \geqslant n/l$. It is obvious that every constant function is sublinear. The functions $g(x) = \sqrt{x}$ and $g(x) = x/ln(x)$ are sublinear as well, while the functions $g(x) = x^2$ and $g(x) = 3x$ are not.

We now show that the existence of a sublinear bound on the error is a necessary and sufficient condition for equivalence of queries that are arbitrary unions of conjunctive queries.

THEOREM 4.2. *Let Q^* and Q be two queries of the same arity that are unions (finite or infinite) of conjunctive queries. Then the following statements are equivalent*:

(1) $Q^* \equiv Q$, *i.e.*, $Q^*(D) = Q(D)$ *on every finite database* D.

(2) *There is a sublinear function g such that $\delta(Q^*, Q)(D) \leqslant g(|D|)$ on every finite database D, where $|D|$ is the size of the database D.*

(3) *For every countable database D, there is a positive integer k such that $\delta(Q^*, Q)(D) \leqslant k$.*

Proof. It is obvious that condition (1) implies condition (2). For the converse, assume that Q^* and Q are conjunctive queries of the same arity such that condition (2) holds, but $Q^* \not\equiv Q$. It follows that there is a finite database D such that $Q^*(D) - Q(D) \neq \varnothing$ or $Q(D) - Q^*(D) \neq \varnothing$. In the first case, there is at least one tuple $\mathbf{a}$ of elements from D such that $\mathbf{a} \in Q^*(D) - Q(D)$. Since g is a sublinear function, there is an integer n such that $g(ln) < n$, where $l = |D|$ is the size of the database D. Let D_n be a database consisting of n disjoint copies of D, and let $\mathbf{a}_i$ be the replica of the tuple $\mathbf{a}$ in the ith copy of D, $1 \leqslant i \leqslant n$. Notice that there are homomorphisms h_i from D to D_n mapping $\mathbf{a}$ to $\mathbf{a}_i$, and homomorphisms h_i' from D_n to D mapping $\mathbf{a}_i$ to $\mathbf{a}$, $1 \leqslant i \leqslant n$. Since conjunctive queries are preserved under homomorphisms, it follows that $\mathbf{a}_i \in Q^*(D_n) - Q(D_n)$ for every $i \leqslant n$. Thus, $\delta(Q^*, Q)(D_n) \geqslant n > g(ln) = g(|D_n|)$, contradicting the hypothesis in condition (2) that the error is bounded by the sublinear function g. A symmetric argument works also for the case that $Q(D) - Q^*(D) \neq \varnothing$.

Note that if a conjunctive query is satisfied in an infinite database D by some tuple $\mathbf{a}$ from D, then there is a finite $D' \subseteq D$ that contains $\mathbf{a}$ and has the property that the conjunctive query is satisfied in D' by $\mathbf{a}$. From this fact and the preservation of conjunctive queries under homomorphisms, it follows that if Q^* and Q are two unions of conjunctive queries that take the same values on all finite databases, then they take the same values on all infinite databases. In turn, this shows that (1) $\Rightarrow$ (3). Finally, the direction (3) $\Rightarrow$ (1) is proved along the lines of (2) $\Rightarrow$ (1), the only difference being that we have to take a countably infinite union of disjoint copies of a finite database D such that $Q^*(D) - Q(D) \neq \varnothing$. ∎

As an immediate consequence of the preceding Theorem 4.2 we derive the following characterization of boundedness of Datalog predicates.

THEOREM 4.3. *Let P be an IDB predicate in a Datalog program π. Then the following statements are equivalent.*

(1) *The predicate P is bounded in π.*

(2) *There is a finite union Q of conjunctive queries and a sublinear function g such that on every finite database D we have that $\delta(P, Q)(D) \leqslant g(|D|)$, where $|D|$ is the size of the database D.*

(3) *There is a finite union Q of conjunctive queries such that for every countable database D there is a positive integer k for which $\delta(P, Q)(D) \leqslant k$.*

Proof. To show (1) $\Rightarrow$ (2), recall that each stage P^i, $i \geqslant 1$, is a finite union of conjunctive queries. Thus, if there is a positive integer m such that $P^\infty(D) = P^m(D)$ on all finite databases D, then we can take $Q = P^m$ and guarantee that $\delta(P, Q)(D) = 0$ on all finite databases D. For the converse direction, if Q is a finite union of conjunctive queries such that the error $\delta(P, Q)(D)$ is bounded by some sublinear function of the size of D, then from Theorem 4.2 it follows that P^∞ is equivalent to Q and, hence, as mentioned earlier in Section 2, P must be bounded. The direction (1) $\Rightarrow$ (3) is a consequence of the fact that if a Datalog predicate is bounded on finite structures, then it is also bounded on all infinite structures. Finally, (3) $\Rightarrow$ (1) follows from Theorem 4.2. ∎

The preceding Theorem 4.3 implies that every unbounded Datalog predicate exhibits a remarkable robustness, namely, not only it is not equivalent to any finite union of conjunctive queries on finite databases, but also its difference on finite databases from any finite union of conjunctive queries can not be kept within any "reasonable" bounds that depend on the size of the database only.

5. RELATIVE APPROXIMATION

In this section, we study approximations of Datalog predicates that satisfy more relaxed quantitative performance guarantees, but at the same time meet the qualitative criterion of being an upper envelope or a lower envelope. The motivation for the quantitative performance guarantees that we are about to introduce comes from the work on approximation algorithms for combinatorial optimization problems (see [GJ79, PS82]). Given an instance I of an optimization problem $\mathscr{P}$, an approximation algorithm A for $\mathscr{P}$ returns a feasible solution S that is used to approximate

the optimum of $\mathscr{P}$. More precisely, if $A(I)$ is the value of the objective function f of $\mathscr{P}$ on the feasible solution S produced by the algorithm A, then $A(I)$ is viewed as an approximation of the optimum value $OPT(I)$ of $\mathscr{P}$ on the instance I (the optimum value $OPT(I)$ is assumed to be a positive number). The performance of the algorithm A is measured by the *relative error* of A on I, i.e., by the quotient $|OPT(I) - A(I)|/OPT(I)$. In particular, if $g(x)$ is a function taking nonnegative values, then A is said to be a $g(x)$-*approximation algorithm* for $\mathscr{P}$ if for every instance I of $\mathscr{P}$ the relative error of A on I is bounded by $g(|I|)$, which means that $|OPT(I) - A(I)| \leqslant g(|I|)\ OPT(I)$. In direct analogy to the above, we now introduce the concepts of *relative approximation* for Datalog predicates.

DEFINITION 5.1. Let P be an IDB predicate in a Datalog program π, let Q be a query of the same arity as P, and let $g(x)$ be a function from positive numbers to nonnegative numbers:

- The query Q is $g(x)$-*approximation* of P if $\delta(P, Q)(D) \leqslant g(|D|)\ |P^\infty(D)|$ on every finite database D, where $|D|$ is the size of the database D.
- The query Q is a k-approximation of P if Q is a $g(x)$-approximation of P, where $g(x) = k$ for all x. We say that Q is a *constant approximation* of P if Q is a k-approximation of P for some $k \geqslant 0$. Similarly, we say that Q is a *sublinear approximation* of P if it is a $g(x)$-approximation of P for some sublinear function g.

Our goal in this section is to investigate whether Datalog predicates possess constant approximations that are both finite unions of conjunctive queries and upper or lower envelopes of the Datalog predicates they approximate. Stipulating that the approximation is a union of conjunctive queries is analogous to seeking polynomial-time approximation algorithms for NP-optimization problems, while, as explained earlier, the restriction to upper envelopes and lower envelopes is analogous to requiring that the approximation algorithms produce feasible solutions.

Let P be an IDB predicate in a Datalog program π. There are two cases in which it is obvious that we can find approximations of P that meet all the above requirements. First, if P is bounded in π, then any stage P^m such that $P^\infty \equiv P^m$ is such an approximation. Second, if Q is a lower envelope of P, then for every $k \geqslant 1$ we have that Q is a k-approximation of P, albeit an uninteresting one, since in this case

$$\begin{aligned}\delta(P, Q)(D) &= |P^\infty(D) - Q(D)| \\ &= |P^\infty(D)| - |Q(D)| \leqslant |P^\infty(D)|.\end{aligned}$$

In particular, even if P is an unbounded Datalog predicate, each stage P^i of P is a finite union of conjunctive queries that is both a lower envelope of P and a k-approximation of P for every $k \geqslant 1$. This situation reinforces the analogy between lower envelopes and feasible solutions of maximization problems. Indeed, approximation algorithms for maximization problems produce values that are less than or equal to the optimum; thus, the algorithms are interesting only when the relative error is less than 1. In contrast, approximation algorithms for minimization problems produce values that are bigger than or equal to the optimum; thus, any uniform bound on the relative error is acceptable.

The preceding remarks suggest that a finite union Q of conjunctive queries is a nontrivial constant approximation of an unbounded Datalog predicate P only when one of the following two conditions is true:

- Q is both an upper envelope of P and a k-approximation of P for some $k \geqslant 0$
- Q is both a lower envelope of P and a k-approximation of P for some $k < 1$.

In the sequel, we consider certain large and well-studied classes of Datalog programs and establish that nontrivial approximations do *not* exist for any unbounded predicates in programs belonging to these classes. On the other hand, we also demonstrate the existence of unbounded Datalog predicates that possess non-trivial approximations.

5.1. *Monadic Datalog Predicates*

A predicate P is *monadic* if it has arity one. A Datalog program π is *monadic* if every IDB predicate of π is monadic. In recent years, monadic Datalog programs have been the focus of extensive study that has shown them to possess several desirable properties (see [CGKV88; KA89]). Our next result establishes that for monadic predicates (and, a fortiori, for monadic Datalog programs) boundedness tantamounts to the existence of nontrivial constant approximations.

THEOREM 5.2. *Let P be a monadic IDB predicate in a Datalog program π. Then the following statements are equivalent*:

(1) *The predicate P is bounded.*

(2) *There is a query Q that is a finite union of conjunctive queries and either Q is both an upper envelope of P and a k-approximation of P for some $k \geqslant 0$ or Q is both a lower envelope of P and a k-approximation of P for some $k < 1$.*

Proof. The nontrivial part is to show that condition (2) implies condition (1). Assume first that Q is a finite union of conjunctive queries that is both an upper envelope of P and a k-approximation of P for some $k \geqslant 0$. Towards a contradiction, assume also that P is unbounded in π, which implies that $P^\infty \not\equiv Q$. Since $P^\infty \subseteq Q$, it follows that there is a finite database D and a single element (tuple of length 1)

a from D such that $a \in Q(D) - P^\infty(D)$. Let $m = \max\{1, |P^\infty(D)|\}$ and let B be the database obtained from D by adding mn "clones" of the element a for some $n > k$; that is to say, B is obtained from D by augmenting the universe of D with mn new elements $a_1, ..., a_{mn}$ that have exactly the same connections to other elements of D as a does, but have no connections to each other. We now claim that

$$Q(D) \cup \{a_1, ..., a_{mn}\} \subseteq Q(B) \quad \text{and} \quad P^\infty(B) = P^\infty(D).$$

The first claim is proved using the preservation of Q under homomorphisms and the fact that for every $i \leqslant mn$ there is a homomorphism h_i from D to B that maps a to a_i. From the existence of these homomorphisms and the preservation of P^∞ under homomorphisms, we can also conclude that $P^\infty(D) \subseteq P^\infty(B)$. The inclusion $P^\infty(B) \subseteq P^\infty(D)$ follows from the hypothesis that P is a unary predicate, the fact that P^∞ is preserved under homomorphisms, and the existence of a homomorphism h from B to D that is the identity on elements of D and maps each a_i to a, $1 \leqslant i \leqslant mn$. Indeed, since P is unary, the new elements $a_1, ..., a_{mn}$ are the only tuples from B that have a chance to be in $P^\infty(B) - P^\infty(D)$. These elements, however, do not enter $P^\infty(B)$, since, otherwise, the element a would have entered $P^\infty(D)$. Thus, we have verified the second claim, namely $P^\infty(B) = P^\infty(D)$. The two claims yield a contradiction, because they imply that $\delta(P, Q)(B) \geqslant mn > k\,|P^\infty(B)|$.

To complete the proof, assume next that Q is a finite union of conjunctive queries that is both a lower envelope of P and a k-approximation of P for some $k < 1$. Thus, on every finite database B we have $\delta(P, Q)(B) = |P^\infty(B)| - |Q(B)| \leqslant k\,|P^\infty(B)|$ and, consequently, $|P^\infty(B)| \leqslant |Q(B)|/(1-k)$ (the hypothesis that $k < 1$ is used at this point). If P is unbounded in π, then there is a finite database D such that $P^\infty(D) - Q(D) \neq \varnothing$. Using the same argument as before, we can show that for every positive integer n there is a database B such that $Q(B) = Q(D)$ and $|P^\infty(B)| \geqslant |Q(B)| + n$. By taking $n > k\,|Q(D)|/(1-k)$, we have that $|P^\infty(B)| > |Q(B)|/(1-k)$, which is again a contradiction. Thus, P must be bounded in π. ∎

Closer examination of the above proof of Theorem 5.2 reveals that the only property of P^∞ used was the fact that P^∞ is a union of unary conjunctive queries. Thus, the same proof yields the following stronger result.

THEOREM 5.3. *Let Q^* and Q be two queries that are unions (finite or infinite) of unary conjunctive queries. Then the following statements are equivalent*:

(1) $Q^* \equiv Q$, *i.e.*, $Q^*(D) = Q(D)$ *on every finite database D.*

(2) *Either Q is both an upper envelope of Q^* and a k-approximation of Q^* for some $k \geqslant 0$ or Q is both a lower envelope of Q^* and a k-approximation of Q^* for some $k < 1$.*

5.2. *Beyond Monadic Predicates*

In this section, we extend the nonapproximability results for monadic programs (Theorem 5.2) to certain large classes of Datalog programs that are not monadic.

With every conjunctive query Q one can associate a *connection graph* in which there is a node for every variable in the query and an edge between any two variables for which there is some literal in the query such that both variables occur in this literal. The *distance* between two variables is the length of the shortest path between the corresponding nodes in the connection graph of the query. We say that a conjunctive query Q is *connected* if its connection graph is connected.

DEFINITION 5.4. Let P be an IDB predicate of arity at least two in a Datalog program π. We say that P is an *unbounded distance predicate* if the following two conditions are satisfied:

- Each expansion of P is connected.
- For every positive integer n, there exists an expansion of P such that the distance between some pair of distinguished variables is at least n.

Many interesting Datalog programs contain IDB predicates that satisfy the above constraints. In particular, the *transitive closure* query and the *same-generation* [U89] query are definable by unbounded distance predicates. Notice that if P is an unbounded distance predicate in a Datalog program π, then P is also unbounded in π. Our next result shows that unbounded distance predicates do not possess upper envelopes that are nontrivial constant approximations.

THEOREM 5.5. *Let P be an unbounded distance predicate in a Datalog program π. Then, there is no finite union Q of conjunctive queries such that Q is both an upper envelope of P and a k-approximation of P for some $k \geqslant 0$.*

Proof. Towards a contradiction, assume that $Q = \bigcup_i Q_i$ is a finite union of conjunctive queries that is an k-approximating upper envelope of P where $k \geqslant 0$. Furthermore, assume that m is the largest distance between any two variables in any of the connection graphs of Q_i. Since P is an unbounded distance predicate, there exists an expansion φ of P, where the distance between some pair of distinguished variables (say, s and t) is u where $u > m$. Moreover, since Q is an upper envelope, there is some r such that $\varphi \subseteq Q_r$. However, there are no two *connected* variables in Q_r that have distance equal or more than u. Therefore, Q_r contains *at least* two variables that are in different components of the connection graph.

Let D be a database containing exactly one *reflexive* tuple for each predicate (for example, $R(a, a, ...a)$). If we consider a database D_j, $j \geqslant 1$, consisting of j disjoint (renamed) clones of the database D, then $|P^\infty(D_j)| = j$. Since Q_r has at

least two connected components, $|Q_r(D_j)| \geq j^2$ and, consequently, $|Q(D_j)| \geq j^2$. As a result, we have that $\delta(P, Q)(D_j) \geq (j-1)\,|P^\infty(D_j)|$ for every $j \geq 1$, which, in turn, implies that Q can not be a constant approximation of P. ∎

Theorem 5.5 does not rule out the possibility that for certain unbounded distance predicates there may exist finite unions of conjunctive queries that are both lower envelopes and k-approximations for some $k < 1$. We conjecture that such lower envelopes do *not* exist for any unbounded distance predicate. If this conjecture is true, then unbounded distance predicates will turn out to have the same non-approximability properties as monadic predicates. In what follows, we confirm this conjecture for a large subclass of unbounded distance predicates.

DEFINITION 5.6. Let P be an IDB predicate of arity at least two in a Datalog program π. We say that P is an *unbounded chain predicate* if the following two conditions are satisfied:

- Each expansion of P is connected.
- For every positive integer n, there exists an expansion of P such that the connection graph of the expansion is a path and the distance between some pair of distinguished variables is at least n.

A comparison of Definition 5.6 with Definition 5.4 shows that the set of unbounded chain predicates is a proper subset of unbounded distance predicates. Nonetheless, unbounded chain predicates encompass a large class of queries that contains properly the class of unbounded IDB predicates in *chain programs* [AP87]. In particular, the standard program defining *same-generation* [U89] query is not a chain program, but its IDB predicate is an unbounded chain predicate. Furthermore, unlike predicates in chain programs, neither an unbounded chain predicate nor the EDB predicates used to define it need be binary. We now show that unbounded chain predicates share the non-approximability properties of monadic predicates.

THEOREM 5.7. *Let P be an unbounded chain predicate in a Datalog program π.*

1. *There is no finite union Q of conjunctive queries such that Q is both an upper envelope of P and a k-approximation of P for some $k \geq 0$.*

2. *There is no finite union Q of conjunctive queries such that Q is both a lower envelope of P and a k-approximation of P for some $k < 1$.*

Proof. The first part of the theorem follows immediately from Theorem 5.5, since every unbounded chain predicate is also an unbounded distance predicate. Thus, we focus on the second part. Towards a contradiction, assume that Q is a finite union of conjunctive queries that is both a lower envelope of P and a k-approximation of P for some $k < 1$. Theorem 2.1 implies that there is a stage P^l of P such that $Q \subseteq P^l \subseteq P^\infty$. Therefore, the stage P^l is both a lower envelope of P and a k-approximation of P for some $k < 1$. It follows that $|P^\infty(D)| - |P^l(D)| \leq k\,|P^\infty(D)|$ on every finite database D and, as a result, $|P^\infty(D)| \leq k'\,|P^l(D)|$, where $k' = 1/(1-k) > 1$. Recall that the stage P^l is a union of finitely many conjunctive queries that are called the lth *level expansions* of P. Note that the connection graphs of these expansions need not be paths. Since each expansion of P is connected, there is a positive integer d such that the maximum distance between any two variables in the connection graph of every lth level expansions of P is at most $2d$.

Using the second condition in the definition of unbounded chain predicate, for *every* positive integer $j > d$ we can find an expansion φ of P such that the connection graph G_φ of φ is a path and for some pair (x, y) of distinguished variables the distance between them in G_φ is at least $2j$. Let us name the sequence of variables in the path from x to y as $v(1) \cdots v(2j)$ and for simplicity let us assume that the distance between x and y is $2j$, so that x is being identified with $v(1)$, while y is being identified with $v(2j)$.

Let D be the database whose objects are the variables $v(1) \cdots v(2j)$ and whose facts are the literals of the conjunctive query φ. The following two statements about this database are true:

1. $P^\infty(D)$ contains a tuple that has both $v(1)$ and $v(2j)$ among its attribute values.

2. $P^l(D)$ contains no tuple that has $v(j-d-p)$ and $v(j+d+q)$ among its attribute values, where $0 \leq p$, $q \leq j-d$.

The first statement is true, because D is the expansion φ of P in which $v(1)$ and $v(2j)$ (i.e., x and y) are among its distinguished variables. The second statement holds because of the following property of containment mappings: If γ is a containment mapping from α to β, and m and n are two variables in α, then the distance between $\gamma(m)$ and $\gamma(n)$ in β is less than or equal to the distance between m and n in α. Since the distance between any two variables in every expansion of P^l is at most $2d$ and the distance between $v(j-d-p)$ and $v(j+d+q)$ in the connection graph G_φ of φ is greater than $2d$, it follows that no tuple in $P^l(D)$ can have both $v(j-d-p)$ and $v(j+d+q)$ among its attribute values.

The above properties of the database D lead us to a new database D'. The desired key property of D' is that it is an extension of D such that tuples that have as attribute values $v(j-d-p)$ and $v(j+d+q)$ as above are in $P^\infty(D')$, but not in $P^l(D')$. Let D_p be a database having the same domain as D and obtained from D by renaming the literals of D in such a way that each variable $v(j-d-i)$ with $i > p$ is renamed to $v(j-d-p)$. Similarly, let D_q be the database

obtained from D by renaming the literals of D in such a way that each variable $v(j+d+s)$ with $s>q$ is renamed to $v(j+d+q)$. Using the fact that the connection graph G_φ of φ is a path, it can be shown that P^∞ over the database $D \cup D_p \cup D_q$ contains at least one tuple with attribute values $v(j-d-p)$ and $v(j+d+q)$ as above. With this as motivation, define the database D' as

$$D' = D \cup \left(\bigcup_p D_{j-d-p} \right) \cup \left(\bigcup_q D_{j+d+q} \right),$$

where $0 \leqslant p,\ q \leqslant j-d$. Note that $P^\infty(D')$ must contain at least $(j-d)^2$ tuples, one for each pair of choices of p and q. Otherwise, (1) cannot be true; i.e., $P^\infty(D)$ cannot contain a tuple that has both $v(1)$ and $v(2j)$ among its attribute values. Furthermore, the connection graph for D' (when viewed as a query) is precisely the same as that of φ (i.e., G_φ), because all new literals introduced in D' are "reflexive" and, thus, do not affect the distance between two distinct variables in D. As a consequence, for every object (variable) u of D' and for every positive integer t, there are at most 2 variables in D at a distance t from u. In particular, there are at most $4d$ variables whose distance from u is at most $2d$, which is the maximum distance between any two variables in an expansion of P^l). Keeping this observation in mind, let us compute an upper bound for the number of tuples in $P^l(D')$. For this, we consider the choices for the value of the first attribute of any tuple in $P^l(D')$. The value of such an attribute can be any one of the $2j$ variables $v(1) \cdots v(2j)$ in D'. However, for any given choice of such a variable, say u, every other attribute value in the tuple must be a variable in D' that is at a distance at most $2d$ from u. Therefore, by applying our previous observation, we conclude that there are at most $2jL$ tuples in $P^l(D')$, where L is a quantity that depends only on d and the arity of the predicate P. On the other hand, by our earlier assumption, $|P^\infty(D')| \leqslant k'\,|P^l(D')|$, which implies that $(j-d)^2 < 2k'jL$. Notice that, since P is an unbounded chain predicate, j can be made arbitrarily large. In particular, we can choose $j > 2(d+k'L)$, which, however, violates the previously derived inequality $(j-d)^2 < 2k'jL$. This completes the proof of the theorem. ∎

5.3. *Counterexamples*

The results obtained up to this point suggest that perhaps no unbounded Datalog predicate can be approximated in a nontrivial way by upper or lower envelopes that are finite unions of conjunctive queries. We now dispel this possibility by showing that the preceding nonapproximability results do not extend to arbitrary unbounded Datalog predicates.

EXAMPLE 5.8. Let R be an unbounded binary IDB predicate in a Datalog program ρ, let E be a binary EDB predicate not occurring in ρ, let P be a binary IDB predicate not occurring in ρ, and let π_1 be the Datalog program obtained from ρ by adding the following two rules:

$$P(x, y) :- E(x, y)$$
$$P(x, y) :- E(y, x), R(y, x).$$

The unboundedness of R in ρ implies that P is unbounded in π_1. Observe, however, that P is not an unbounded distance predicate in π_1. Let $Q(x, y)$ be the query $E(x, y) \vee E(y, x)$. It is obvious that Q is both a finite union of conjunctive queries and an upper envelope of P. Moreover, Q is a 1-approximation of P, since on every finite database D

$$\delta(P, Q)(D) = |Q(D)| - |P^\infty(D)| \leqslant |E| \leqslant |P^\infty(D)|.$$

Let $Q'(x, y)$ be the query $E(x, y)$, which is obviously both conjunctive query and a lower envelope of P. A moment's reflection shows that Q' is also an $\frac{1}{2}$-approximation of P, since on every finite database D

$$\delta(P, Q')(D) = |P^\infty(D)| - |Q'(D)| \leqslant |E| \leqslant |Q'(D)|.$$

Note that the above construction is not an isolated example of an unbounded and approximable Datalog predicate, but rather a whole family of examples having the same structure. Next, we construct a somewhat different family of unbounded and approximable Datalog predicates that will be of interest to us in the sequel.

EXAMPLE 5.9. Let T be an unbounded unary IDB predicate in a Datalog program τ, let E be a binary EDB predicate not occurring in τ, let P be a binary IDB predicate not occurring in τ, let S be a unary IDB predicate not occurring in τ, and let π_2 be the Datalog program obtained from τ by adding the following four rules:

$$P(x, y) :- E(x, y)$$
$$S(x) :- E(x, z)$$
$$S(x) :- E(z, x)$$
$$P(x, x) :- S(x), T(x).$$

As in the previous example, P is unbounded in π_2, but it is not an unbounded distance predicate in π_2. It is easy to verify that the following finite union Q of conjunctive queries is both an upper envelope of P and a 2-approximation of P:

$$Q(x, y) :- E(x, y)$$
$$Q(x, x) :- E(x, z)$$
$$Q(x, x) :- E(z, x).$$

Let $Q'(x, y)$ be the query $E(x, y)$, which is obviously both a conjunctive query and a lower envelope of P. Since $|P^\infty(D)| \leqslant 3\,|E|$ on every database D, it follows that Q' is also an 2/3-approximation of P.

There are two different issues that are raised by the above examples. The first is whether it is possible to delineate the boundary between approximability and nonapproximability of unbounded Datalog predicates. This could be achieved by finding general conditions that are sufficient for approximability and also by identifying additional syntactic or structural properties of unbounded IDB predicates that yield non-approximability results similar to the ones obtained in Theorems 5.2 and 5.5. The second issue is whether unbounded and approximable Datalog predicates can be approximated up to any desired degree of accuracy. In the next section we embark on an investigation of this second issue.

6. UPPER AND LOWER APPROXIMATION SCHEMES

It is well known that, unless P = NP, certain combinatorial optimization problems, such as MAX SAT, have an *approximation threshold* (see [Pa94]). This means that there is a constant $c > 0$ and a polynomial-time approximation algorithm with relative error bounded by c, but no polynomial-time approximation algorithm achieves relative error bounded by a constant $c' < c$, unless P = NP. In contrast to this, certain other combinatorial optimization problems, such as KNAPSACK, possess *polynomial-time approximation schemes*, that is to say for each $\varepsilon > 0$ there is a polynomial-time algorithm that approximates the optimization problem with relative error bounded by ε. In direct analogy with polynomial-time approximation schemes, we introduce here the concepts of *upper* and *lower approximation schemes* for Datalog predicates.

DEFINITION 6.1. Let P be an IDB predicate in a Datalog program π and let Q_n, $n \geqslant 1$, be a sequence of queries such that each Q_n is a finite union of conjunctive queries of the same arity as P.

- The sequence Q_n, $n \geqslant 1$, is an *upper approximation scheme* of P if each Q_n is an upper envelope of P and for every $\varepsilon > 0$ there is an integer $n(\varepsilon)$ such that the query $Q_{n(\varepsilon)}$ is an ε-approximation of P.
- The sequence Q_n, $n \geqslant 1$, is a *lower approximation scheme* P if each Q_n is an lower envelope of P and for every $\varepsilon > 0$ there is an integer $n(\varepsilon)$ such that the query $Q_{n(\varepsilon)}$ is an ε-approximation of P.

Observe that if Q_n, $n \geqslant 1$, is an upper (or lower) approximation scheme of P, then the sequence $Q'_n = \bigcap_{i=1}^{n} Q_i$, $n \geqslant 1$ (respectively, the sequence $Q'_n = \bigcup_{i=1}^{n} Q_i$, $n \geqslant 1$), is also an upper (or lower) approximation scheme of P. In other words, if P has an upper (or lower) approximation scheme, then there is a decreasing (respectively, increasing) sequence of queries Q_n, $n \geqslant 1$, that forms an upper (or lower) approximation scheme of P. Next, we claim that if Q_n, $n \geqslant 1$, is an upper (or a lower) approximation scheme of P, then $\bigcap_{n=1}^{\infty} Q_n \equiv P^\infty$ (respectively, $\bigcup_{n=1}^{\infty} Q_n \equiv P^\infty$). For this, it is enough to show that for every finite database D there is an integer n such that $|Q_n(D) - P^\infty(D)| = 0$. If D is a finite database such that $P^\infty(D) = \varnothing$, then we can take any integer n such that Q_n is a $\frac{1}{2}$-approximation of P. Otherwise, we take an integer n such that Q_n is a $1/(2\,|P^\infty(D)|)$-approximation of P. In this case, we have that $|Q_n(D) - P^\infty(D)| < \frac{1}{2}$ and, hence, $|Q_n(D) - P^\infty(D)| = 0$, since $|Q_n(D) - P^\infty(D)|$ must be a non-negative integer. Finally, note that if Q_n, $n \geqslant 1$, is a lower approximation scheme of P, then the sequence P^n, $n \geqslant 1$, of the stages of P is also a lower approximation scheme of P. Indeed, since each Q^n is a lower envelope of $P = \bigcup_{m=1}^{\infty} P^m$, it follows that for each n there is an m such that $Q^n \subseteq P^m$. The next proposition summarizes the preceding observations.

PROPOSITION 6.2. *Let P be an IDB predicate in a Datalog program π.*

- *If Q_n, $n \geqslant 1$, is an upper approximation scheme of P, then $P^\infty \equiv \bigcap_{n=1}^{\infty} Q_n$.*
- *If Q_n, $n \geqslant 1$, is a lower approximation scheme of P, then $P^\infty \equiv \bigcup_{n=1}^{\infty} Q_n$.*
- *P has a lower approximation scheme if and only if the sequence P^n, $n \geqslant 1$, of the stages of P is a lower approximation scheme of P.*

We do not know of any unbounded Datalog predicates that possess an upper or a lower approximation scheme. This leads us to formulate the following problem.

OPEN PROBLEM. Let P be an IDB predicate in a Datalog program π. Prove or disprove that the following statements are equivalent:

(1) The predicate P is bounded in π.

(2) There is an upper approximation scheme of P.

(3) There is a lower approximation scheme of P.

In what follows, we confirm the equivalence of the above statements for a large class of Datalog predicates. Before stating the result precisely, we need to give the following definitions.

DEFINITION 6.3. We say that a conjunctive query θ is *reflexive* if one of the variables occurs at least twice in the head of θ; otherwise, we say that θ is a *nonreflexive* query.

DEFINITION 6.4. Let π be a Datalog program and let P be an IDB predicate of π.

• We say that a rule r of π is *reflexive* if one of the variables occurs at least twice in the head of r; otherwise, we say that r is a *nonreflexive* rule.

• We say that P is a *reflexive* predicate of π if P occurs in the head of at least one reflexive rule of π; otherwise, we say that P is a *nonreflexive* predicate of π.

Remark 6.5. The above definition implies that if P is a nonreflexive Datalog predicate, then every expansion φ of P is a nonreflexive conjunctive query. Moreover, if θ is a conjunctive query such that $\varphi \subseteq \theta$, then θ must also be a nonreflexive conjunctive query. Indeed, otherwise there can be no containment mapping from θ to φ (and, hence, $\varphi \not\subseteq \theta$).

Note that in Example 5.8 the predicate P is nonreflexive, while in Example 5.9 the predicate P is reflexive. Our next result establishes that nonreflexive Datalog predicates are bounded if and only if they possess upper or lower approximation schemes.

THEOREM 6.6. *Let P be an unbounded and nonreflective predicate of arity s in a Datalog program π.*

• *For every upper envelope Q of P that is a finite union of conjunctive queries and for every constant $k < 1(s^s - 1)$, the query Q is not a k-approximation of P.*

• *For every lower envelope Q of P that is a finite union of conjunctive queries and for every constant $k < 1/s^s$, the query Q is not a k-approximation of P.*

As a result, if P is an unbounded and nonreflexive predicate in a Datalog program π, then P has neither an upper nor a lower approximation scheme.

Proof. Let P be an unbounded and nonreflexive binary predicate in a Datalog program π. Assume first that Q is a finite union of conjunctive queries that is an upper envelope of P. Let Q' be the finite union of conjunctive queries consisting of precisely the nonreflexive conjunctive queries in Q. We claim that Q' is an upper envelope of P. Indeed, since $P^\infty \subseteq Q$, for every expansion φ of P there is a conjunctive query θ in Q such that $\varphi \subseteq \theta$. Moreover, since P is a nonreflexive Datalog predicate, each such expansion φ of P is a nonreflexive query and, hence, from Remark 6.5 we conclude that θ is also a nonreflexive conjunctive query. Thus, we have that $P^\infty \subseteq Q' \subseteq Q$ and, consequently, if D is a finite database, then $\delta(P, Q')(D) \subseteq \delta(P, Q)(D) \leqslant k$. Therefore, without loss of generality, from now on we assume that all expansions in Q are nonreflexive conjunctive queries.

Since P is unbounded, $Q \not\equiv P^\infty$. Therefore, there exists a nonreflexive conjunctive query $\theta \in Q$ such that $\theta \not\subseteq P^\infty$. Let D be the "representative database" of θ, i.e., the objects in D are the variables in θ and the facts in D are the literals in the body θ. Let $\mathbf{t}$ be the tuple from D corresponding to the distinguished variables of θ. Note that $\mathbf{t} \in Q(D) - P^\infty(D)$. Moreover, since $\mathbf{t}$ is the tuple corresponding to the head of θ, all variables of $\mathbf{t}$ must be distinct, say $(t_1, ..., t_s)$. For every positive integer n, let D_n be the database obtained from D by adding n "clones" $t_{i,j}$, $1 \leqslant j \leqslant n$, of each variable t_i, $1 \leqslant i \leqslant s$. From the preservation of the queries Q and P^∞ under homomorphisms, it follows that if $j1, ..., js \in \{1 \cdots n\}$, then $(t_{1,j1}, ..., t_{s,js}) \in Q(D_n) - P^\infty(D_n)$. As a result,

$$|Q(D_n) - P^\infty(D_n)| \geqslant n^s. \tag{1}$$

Next, we will find an upper bound on the cardinality $|P^\infty(D_n)|$ of P^∞ on the database D_n. If a tuple $\mathbf{a} = (a_1, ..., a_s)$ belongs to $P^\infty(D_n)$, then there are three cases to consider:

(i) $\mathbf{a}$ is an element of $P^\infty(D)$;

(ii) at least one, but not all, of the a_j's, $1 \leqslant j \leqslant s$, is an element of D;

(iii) each a_j, $1 \leqslant j \leqslant s$, is one of the clones.

Note that the number of possible tuples in case (ii) is bounded by $n^{s-1} f(s, |D|)$, where f is a function of s and $|D|$ only (a trivial such bound is $n^{s-1} s\, |D|^s$). We now estimate an upper bound for (iii). Altogether, there are sn clones; hence, the total numbers of s-tuples of clones is $(sn)^s$. However, it is clear that for every $j1, ..., jn \in \{1, ..., n\}$, none of the n^s tuples $(t_{1,j1}, ..., t_{s,jn})$ belongs to $P^\infty(D_n)$, since $(t_1, ..., t_s) \notin P^\infty(D)$ and there are homomorphisms from D_n to D mapping $(t_{1,j1}, ..., t_{s,jn})$ to $(t_1, ..., t_s)$. Therefore, an upper bound on the number of tuples in (iii) is $(sn)^s - n^s$. As a result,

$$|P^\infty(D_n)| \leqslant |P^\infty(D)| + f(s, |D|) + (ns)^s - n^s. \tag{2}$$

From the above equations (1) and (2), it follows that

$$\frac{|Q(D_n) - P^\infty(D_n)|}{|P^\infty(D_n)|} \geqslant \frac{n^s}{|P^\infty(D)| + n^{s-1} f(s, |D|) + (ns)^s - n^s}.$$

Thus, if Q is a k-approximation of P for some k, then for all positive integers n

$$k \geqslant \frac{n^s}{|P^\infty(D)| + n^{s-1} f(s, |D|) + (ns)^s - n^s}.$$

Hence, by letting $n \to \infty$, we conclude that $k \geqslant 1/(s^s - 1)$.

The proof proceeds essentially along the same lines if Q is a query that is both a lower envelope of P and a finite union of conjunctive queries. In this case, there is some (nonreflexive) expansion φ of P^∞ that is not contained in Q. Thus, we clone the representative database of φ. The only difference is in computing an upper bound of $|P^\infty(D_n)|$,

since here we cannot exclude the tuples $(t_{1,j1} \cdots t_{s,js})$, where $j1, ..., js \in \{1..., n\}$. It follows that in this case

$$k \geqslant \frac{n^s}{|P^\infty(D)| + ns^{-1}f(s, |D|) + (ns)^s}.$$

Hence, by letting $n \to \infty$, we conclude that $k \geqslant 1/s^s$. ∎

Theorem 6.6 asserts the following for binary nonreflexive Datalog predicates P and binary queries Q that are finite unions of conjunctive queries:

1. If Q is an upper envelope of P and a k-approximation of P, then $k \geqslant 1/3$.
2. If Q is an lower envelope of P and a k-approximation of P, then $k \geqslant 1/4$.

Note that Theorem 6.6 applies to the predicate P in the program π_1 below, which was introduced earlier in Example 5.8:

$$P(x, y) : -E(x, y)$$
$$P(x, y) : -E(y, x), R(y, x).$$

Recall that P has an upper envelope that is a union of two conjunctive queries and a 1-approximation of P. With some extra effort, it can be shown that 1 is actually the approximation threshold of P by upper envelopes that are finite unions of conjunctive queries. Recall also that P has a lower envelope that is a conjunctive query and a 1/2-approximation of P. As it turns out, 1/2 is actually the approximation threshold of P by lower envelopes that are finite unions of conjunctive queries. Thus, the bounds derived in the preceding Theorem 6.6 do not constitute the approximation thresholds of all unbounded and nonreflexive Datalog predicates.

Let us now revisit the program π_2 below, which was introduced in Example 5.9,

$$P(x, y) : -E(x, y)$$
$$S(x) : -E(x, z)$$
$$S(x) : -E(z, x)$$
$$P(x, x) : -S(x), T(x),$$

where T is an unbounded predicate in a Datalog program τ. The predicate P is reflexive in π_2 and, thus, Theorem 6.6 does not apply to it. Recall that P has a lower envelope that is a conjunctive query and a 2/3-approximation of P. Using an ad hoc argument, it can be shown that P does not have a lower approximation scheme. Moreover, 2/3 turns out to be the approximation threshold of P by lower envelopes that are finite unions of conjunctive queries. We now describe briefly this argument. Let Q be a query that is both a lower envelope of P and a finite union of conjunctive queries. We will show that there is a finite database on which the error of approximating P by Q equals 2/3. It is clear that on every finite database D

$$Q(D) \subseteq \Big\{(d_1, d_2) : D \models E(d_1, d_2) \vee \Big[(d_1 = d_2) \wedge S(d_1) \wedge \Big(\bigvee_{m=1}^{\infty} T^m(d_1)\Big)\Big]\Big\},$$

where T^m, $m \geqslant 1$, are the stages of T. Moreover, since Q is a finite union of conjunctive queries, there is a positive integer m_0 such that for every database D

$$Q(D) \subseteq \{(d_1, d_2) : D \models E(d_1, d_2) \vee [(d_1 = d_2) \wedge S(d_1) \wedge T^{m_0}(d_1)]\}.$$

Since T is an unbounded predicate, there is a database D' over the EDB predicates of the program τ such that $|T^\infty(D') - T^{m_0}(D')| \geqslant 2$ (otherwise, T^{m_0} would be an absolute approximation of T, contradicting Theorem 4.3). Let a and b be two elements of $T^\infty(D') - T^{m_0}(D')$. Using these two elements, extend D' to a database D^* over the EDB predicates of π_2 in which E is interpreted by the singleton $\{(a, b)\}$. It follows that $Q(D^*) = \{(a, b)\}$, while $P^\infty(D^*) = \{(a, b), (a, a), (b, b)\}$. Thus, the error of approximating P by Q on D^* is 2/3, which implies that 2/3 is indeed the approximation threshold of P by lower envelopes that are finite unions of conjunctive queries.

As mentioned earlier, we have not found any examples of unbounded Datalog predicates that possess an upper or a lower approximation scheme. It should be pointed out, however, that such examples can be constructed easily for Datalog($\neq$), the extension of Datalog that allows for inequalities in the bodies of the rules. Indeed, assume that T is an unbounded IDB predicate in a Datalog program τ and let π_3 be the following Datalog($\neq$) program:

$$P(x, y) : -x \neq y$$
$$P(x, x) : -T(x).$$

We claim that the sequence P^m, $m \geqslant 1$, of the stages of P is a lower approximation scheme of P. It is easy to see that for every database D and every $m \geqslant 1$,

$$\frac{|P^\infty(D)| - |P^m(D)|}{|P^\infty(D)|} \leqslant \frac{|D|}{|D|^2 - D} = \frac{1}{|D| - 1}. \qquad (3)$$

Given a number $\varepsilon > 0$, let n be a positive integer such that $1/(n-1) < \varepsilon$. Then the stage P^{n^2} achieves an approximation error less than ε. Indeed, if a database D has at most n

elements in its universe, then $P^{\infty}(D) = P^{n^2}(D)$, so in this case the error is equal to 0. If D has more than n elements, then the above inequality (3) implies that the approximation error is at most $1/(|D|-1) < 1/(n-1) < \varepsilon$.

It should be emphasized that structural results about Datalog programs do not always extend to similar results about Datalog($\neq$) programs. A relevant case in point is the aforementioned result by Ajtai and Gurevich [AG94], which asserts that an IDB predicate P in a Datalog program π defines a first-order query if and only if P is bounded in π. This result, however, fails for IDB predicates in Datalog($\neq$). It is conceivable that a similar difference exists between Datalog and Datalog($\neq$) on the issue of boundedness versus approximation schemes.

7. FIRST-ORDER APPROXIMATION

So far, we have attempted to approximate unbounded Datalog queries using finite unions of conjunctive queries as approximating objects. In view of the nonapproximability results for monadic predicates and for unbounded chain predicates established in Section 5, it is natural to ask whether it is possible to obtain positive results by enlarging the class of approximating objects. The obvious candidate for this task is the class of first-order queries. Since first-order queries are essentially equivalent to relational algebra or relational calculus queries, this endeavor is an attempt to approximate recursive Datalog queries by queries expressible in a recursion-free, but relationally complete, language.

Clearly, if a Datalog query is shown not to be approximable by any first-order query, then, a fortiori, this Datalog query is not first-order definable. Thus, any such nonapproximability results yield nonexpressibility results as a by-product. We begin our investigation by observing that in many cases the converse is also possible, i.e., nonexpressibility results can be used to derive nonapproximability counterparts.

PROPOSITION 7.1. *Let Q^* be a n-ary query such that the Boolean query*

$$(\exists x_1 \cdots \exists x_n)\, Q^*(x_1, ..., x_n)$$

is not first-order definable. Then the following hold for the query Q^*:

- *There does not exist a first-order query Q that is both an upper envelope of Q^* and a k-approximation of Q^* for some $k \geqslant 0$.*
- *There does not exist a first-order query Q that is a k-approximation of Q^* for some $k < 1$.*

Proof. We claim that if Q were a first-order query satisfying one of the above conditions, then

$$(\exists x_1 \cdots \exists x_n)\, Q^*(x_1, ..., x_n) \Leftrightarrow (\exists x_1 \cdots \exists x_n)\, Q(x_1, ..., x_n)$$

and, consequently, the Boolean query $(\exists x_1 \cdots \exists x_n)\, Q^*(x_1, ..., x_n)$ would be first-order definable. Indeed, suppose that Q is a first-order query that is a k-approximation of Q^* for some $k < 1$. Thus, on every finite database D we have that $|Q^*(D) - Q(D)| + |Q(D) - Q^*(D)| \leqslant k\,|Q^*(D)|$. If $Q^*(D) = \varnothing$, then $|Q^*(D) - Q(D)| + |Q(D) - Q^*(D)| = 0$ and, hence, $Q^*(D) = Q(D) = \varnothing$. If, on the other hand, $Q^*(D) \neq \varnothing$, then $Q(D) \neq \varnothing$, since, otherwise, $|Q^*(D)| \leqslant k\,|Q^*(D)|$, which is ruled out by the assumption that $k < 1$. The argument for the case in which Q^* satisfies the first condition is similar (and simpler). ∎

We illustrate the uses and limitations of Proposition 7.1 by focusing on two of the most thoroughly studied Datalog queries namely, the *transitive closure* query *TC*, which asks "is there a path from x to y?" and the *cycle* query, which asks "is x on a cycle?" Notice that the transitive closure query is an unbounded chain predicate, while the cycle query is monadic, as they are defined by the IDB predicates in the Datalog program π below:

$$TC(x, y) :- E(x, y)$$
$$TC(x, y) :- E(x, z), TC(z, y)$$
$$Cycle(x) :- TC(x, x).$$

It is well known that the Boolean query *acyclicity*, which asks "is the graph acyclic?," is not first-order definable (see [Fa75; FSV93]). Since *acyclicity* is the complement of the Boolean query $(\exists x)\, Cycle\,(x)$, we can apply Proposition 7.1 and conclude that the *Cycle* query is not k-approximable by any first-order query Q such that either Q is an upper envelope of *Cycle* and k is an arbitrary constant or Q is an arbitrary first-order query and $k < 1$. In contrast to this, Proposition 7.1 gives no information as to whether the *transitive closure* query *TC* is first-order approximable, since

$$(\exists x)(\exists y)\ \mathrm{TC}(x, y) \Leftrightarrow (\exists x)(\exists y)\ E(x, y).$$

Our main technical contribution in this section a sharp nonapproximability result for the transitive closure query. The proofs of our earlier nonapproximability results involved methods, such as homomorphism techniques, that are tailored for unions of conjunctive queries. Since we allow here arbitrary first-order queries as approximations, a different technical tool is needed. In what follows, we use Ehrenfeucht–Fraïssé games to establish nonapproximability of the query *TC* by first-order queries. In the past, the method of combinatorial games has been used extensively in database theory and finite model theory to establish qualitative results that take the form of lower bounds for expressibility in database query languages and various logics (cf. [Fa75; KV90; FSV93]). What we offer here is an

apparently new use of combinatorial games that makes it possible to obtain results of quantitative character.

THEOREM 7.2. *The following hold for the Transitive Closure query TC.*

- *There does not exist a first-order query Q that is both an upper envelope of TC and a k-approximation of TC for some* $k \geqslant 0$.
- *There does not exist a first-order query Q that is a k-approximation of TC for some* $k < 1$.

Proof. This proof requires familiarity with Ehrenfeucht–Fraïssé games for first-order logic. We refer the reader to [Fa75; FSV93] for the relevant background on this topic.

For the first part of the theorem, let Q be a first-order definable query that is both an upper envelope of TC and a k-approximation for some $k \geqslant 0$. Let $\varphi(x, y)$ be a first-order formula defining Q and let m be the quantifier rank of $\varphi(x, y)$. We consider two undirected graphs G and H that are defined as

- G is a "very large" cycle; i.e., the number of nodes of G is "much larger" than m. For our purposes, it is enough to take a cycle with n nodes, where $n \geqslant 3^{3^m}$.
- H is the disjoint union of n copies $G_1, ..., G_n$ of the cycle G. Thus, H has n^2 nodes in total.

Let (a, b) be a pair of nodes in G such that the distance between a and b is at least 3^m. Using standard facts about Ehrenfeucht–Fraïssé games (see [FSV93]), it is not hard to see that the following holds: if (c, d) is a pair of nodes from H such that c and d are in different connected components (cycles) of H, then for every first-order formula $\psi(x, y)$ of quantifier rank m

$$G \models \psi(a, b) \Leftrightarrow H \models \psi(c, d).$$

In particular, the above equivalence holds for the formula $\varphi(x, y)$ that defines the query Q. Since $G \models TC(a, b)$ and $TC \subseteq Q$, it follows that $H \models \varphi(c, d)$ for every pair (c, d) of nodes of H such that c and d are in different cycles of H. As a result, $|Q(H)| \geqslant n^4 - n^3$, since there are $n^2 - n$ pairs of different cycles in H and each such pair contains n^2 pairs (c, d) as above. On the other hand, it is clear that $|TC(H)| = n^3$. Thus, $\delta(TC, Q)(H) \geqslant n - 2$ and, hence, the error cannot be bounded by any constant $k \geqslant 0$.

For the second part of the theorem, let Q be a first-order definable query that is a k-approximation of TC for some $k < 1$ (note that here we are not assuming that Q is a lower envelope of TC). We consider the following two cases that are determined by the behavior of the query Q on "very large" cycles. In the first case, we assume that there is some "very large" cycle G of cardinality n on which the query Q contains a pair (a, b) of nodes such that the distance from a to b is "sufficiently large." In this case, the proof proceeds as before; namely, we take a graph H that consists of n copies of the cycle G and show that $|Q(H)| \geqslant n^4 - n^3$, while $|TC(H)| = n^3$. In the second case, we must have that on all "very large" cycles the query Q contains only pairs of nodes that are not "very far apart", i.e., their distance is at most 3^m, where m is the quantifier rank of the first-order formula that defines the query Q. From this, it follows that $|Q(G)| = O(n)$ on all "very large" cycles G with n nodes. On the other hand, since Q is a k-approximation of TC, we must have that $|TC(G)| - |Q(G)| \leqslant |TC(G) - Q(G)| \leqslant k\ |TC(G)|$. Thus, $|TC(G)|\ (1 - k) \leqslant |Q(G)|$, which is a contradiction, since $|TC(G)| = n^2$, $|Q(G)| = O(n)$, and $k < 1$. ∎

Remark 7.3. Several remarks are in order now.

- Theorem 7.2 implies immediately that the transitive closure query TC is not first-order definable on finite databases. This well-known result had been established earlier also using games ([Fa75]) or quantifier-elimination ([AU79]).
- The proof of the first part of Theorem 7.2 actually shows that no first-order upper envelope of TC is a sublinear approximation of TC.
- On the other hand, TC turns out to have a 2-approximation Q that is a finite union of conjunctive queries, but is not an upper or a lower envelope of TC. Thus, the hypothesis that Q is an upper envelope is indispensable in proving the first part of Theorem 7.2. To see this, let $Q(x, y)$ be the query $E(x, y) \vee E(y, x)$. It is clear that Q is a finite union of conjunctive queries that is neither an upper nor a lower envelope of TC. Furthermore, it is easy to verify that on every finite graph G

$$\delta(TC, Q)(G) \leqslant |E| + |TC(G)| \leqslant 2\ |TC(G)|$$

and, consequently, Q is a 2-approximation of TC.

It remains an interesting open problem to determine whether or not the nonapproximability properties of TC are shared by all unbounded chain predicates.

ACKNOWLEDGMENTS

We thank Moshe Y. Vardi for listening to an informal presentation of some of the work reported here and for asking penetrating questions that made us realize that our proof of Theorem 4.3 can be extended to a proof of Theorem 4.2.

REFERENCES

[AP87] F. Afrati and C. H. Papadimitriou, The parallel complexity of simple chain queries, *in* "Proceedings, 6th ACM Symp. on Principles of Database Systems, 1987," pp. 210–213.

[AG94] M. Ajtai and Y. Gurevich, DATALOG vs First-Order Logic, *J. Comput. System Sci.* **49** (1994), 562–588.

[AU79] A. V. Aho and J. D. Ullman, Universality of data retrieval languages, *in* "Proceedings, 6th ACM Symp. on Principles of Programming Languages, 1979," pp. 110–117.

[Co74] S. A. Cook, An observation of time-storage trade-off, *J. Comput. System Sci.* **9** (1974), 308–316.

[CGKV88] S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi, Decidable optimization problems for database logic programs, *in* "Proc. 20th ACM Symp. on Theory of Computing," pp. 447–490.

[CM77] A. K. Chandra and P. M. Merlin, Optimal implementation of conjunctive queries in relational databases, *in* "Proc. 9th ACM Symp. on Theory of Computing, New York, 1977," pp. 77–90.

[C93] S. Chaudhuri, Finding nonrecursive envelopes for recursive predicates, *in* "Proceedings, 12th ACM Symp. on Principles of Database Systems, Washington, DC, 1993."

[Fa75] R. Fagin, Monadic generalized spectra, *Z. Math. Logik* **21** (1975), 89–96.

[FSV93] R. Fagin, L. Stockmeyer, and M. Y. Vardi, On Monadic NP vs. Monadic co-NP, *in* "Proc. 8th IEEE Conf. on Structure in Complexity Theory, 1993," pp. 19–30.

[GMSV87] H. Gaifman, H. Mairson, Y. Sagiv, and M. Y. Vardi, Undecidable optimization problems for database logic programs, *in* "Proc. 2nd IEEE Symp. on Logic in Computer Science, Ithaca, 1987," pp. 106–115.

[GJ79] M. R. Garey and D. S. Johnson, "Computers and Intractability —A Guide to the Theory of *NP*-Completeness," Freeman, San Francisco, 1979.

[Io86] Y. E. Ioannides, Bounded recursion in deductive databases, *Algorithmica* **1** (1986), 391–385.

[KA89] P. Kanellakis and S. Abiteboul, Deciding bounded recursion in database logic programs, *SIGACT News* **20** (4), 1989.

[KV90] P. G. Kolaitis and M. Y. Vardi, On the expressive power of Datalog: Tools and a case study, *in* "Proc. 9th ACM Symp. on Principles of Database Systems, 1990," pp. 61–71.

[MUV84] D. Maier, J. D. Ullman, and M. Y. Vardi, On the foundations of the universal relation model, *ACM Trans. on Database Systems* **9** (1984), 283–308.

[Na89] J. F. Naughton, Data independent recursion in deductive databases, *J. Comput. Syst. Sci.* **38** (1989), 259–289.

[PS82] C. H. Papadimitriou and K. Steiglitz, "Combinatorial Optimization-Algorithms and Complexity," Prentice–Hall, Englewood Cliffs, NJ, 1982.

[Pa94] C. H. Papadimitriou, "Computational Complexity," Addison–Wesley, Reading, MA, 1994.

[SY81] Y. Sagiv and M. Yannakakis, Equivalences among relational expressions with the union and difference operators, *J. Assoc. Comput. Mach.* **27**, No. 4 (1981), 633–655.

[U89] J. D. Ullman, "Principles of Database and Knowledge Base Systems," Vol. 2, Computer Science Press, New York, 1989.

[Va82] M. Y. Vardi, The complexity of relational query languages, *in* "Proc. 14th ACM Symp. on Theory of Computing, San Francisco, 1982," pp. 137–146.

Printed in Belgium