On Impossibility of Decremental Recomputation of Recursive Queries in Relational Calculus and SQL*

Guozhu Dong
Department of Computer Science
University of Melbourne
Parkville, Vic. 3052, Australia
Email: dong@cs.mu.oz.au

Leonid Libkin
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974, USA
Email: libkin@research.att.com

Limsoon Wong
Institute of Systems Science
Heng Mui Keng Terrace
Singapore 0511
Email: limsoon@iss.nus.sg

Abstract

We study the problem of maintaining recursively-defined views, such as the transitive closure of a relation, in traditional relational languages that do not have recursion mechanisms. In particular, we show that the transitive closure cannot be maintained in relational calculus under deletion of edges. We use new proof techniques to show this result. These proof techniques generalize to other languages, for example, to the language for nested relations that also contains a number of aggregate functions. Such a language is considered in this paper as a theoretical reconstruction of SQL. Our proof techniques also generalize to other recursive queries. Consequently, we show that a number of recursive queries cannot be maintained in an SQL-like language. We show that this continues to be true in the presence of certain auxiliary relations. We also relate the complexity of updating transitive closure to that of updating the same-generation query and show that the latter is strictly harder than the former. Then we extend this result to that of updating queries based on context-free sets.

1 Problem Statement and Summary

It is well known that relational calculus (equivalently, first-order logic) cannot express recursive queries such as transitive closure [1]. However, in a real database system, it is reasonable to store both the relation and its transitive closure and update the latter whenever edges are added to or removed from the former. Doing this is known under the name of *view maintenance*. In this paper we consider the problem of whether the above update problem for maintaining transitive closure and other recursive queries can be accomplished using relational calculus or using its practical SQL-like extensions. We also compare the complexity of maintaining transitive closure against the complexity of maintaining "same generation" and context-free chain queries.

In this paper, we use the letter R to denote a binary relation, and R^+ to denote its transitive closure. It can be proved [6, 2] that given R, R^+, and a new edge (x, y) to be added to R, the transitive closure

*To appear in *Proceedings of 5th International Workshop on Database Programming Languages*, Gubbio, Italy, September 1995

$R^+_{+(x,y)}$ of $R \cup \{(x,y)\}$ can be expressed in first-order logic and thus in relational calculus. In particular, for all u and v, $R^+_{+(x,y)}(u,v)$ iff $R^+(u,v)$, or $R^+(u,x)$ and $y = v$, or $u = x$ and $R^+(y,v)$, or $R^+(u,x)$ and $R^+(y,v)$. Thus transitive closure can be incrementally maintained in a relational database.

The problem of updating the transitive closure after an edge has been removed is more difficult. The best positive solution so far is that of Dong and Su [5]. They proved that if R is acyclic, then the transitive closure $R^+_{-(x,y)}$ of R with the edge (x,y) removed can be defined in first-order logic in terms of R, R^+, and (x,y). Thus transitive closure can be decrementally maintained in a relational database provided the relation involved is acyclic. But this is not satisfactory because acyclicity cannot be tested in relational calculus [10].

Another solution is that of Immerman and Patnaik [14]. They proved that transitive closure of undirected graphs can always be maintained, provided some auxiliary ternary relations can be used. Dong and Su [7] strengthened this result further by showing that transitive closure of undirected graphs can be maintained using only auxiliary binary relations. They also showed that it cannot be done using only auxiliary unary relations.

In Section 2, we prove that transitive closure cannot be decrementally maintained in a relational database in general. That is, $R^+_{-(x,y)}$ cannot be expressed in relational calculus in terms of R, R^+, and (x,y) when R is a directed graph that is not necessarily acyclic. We also consider the problem of maintaining transitive closure in a context where some auxiliary relations are available. Dong and Su [7] also obtained results that are similar to ours. However, the proof techniques involved are very different. Most importantly, their proof technique is only applicable to the particular case of maintaining transitive closure in relational calculus. Ours is much simpler and can be generalized to more expressive languages and other recursive queries. In particular, instead of transitive closure, any query complete for DLOGSPACE can be used.

In Section 3 we show that our technique extends naturally to prove that transitive closure cannot be decrementally maintained using query languages having the power of SQL. That is, we show that the availability of arithmetic operations and GROUP-BY does not help at all. We also extend this result in the presence of simple auxiliary relations. In addition, we exhibit a query that illustrates the additional power of using an SQL-like language incrementally. This query, which is inexpressible in SQL, is expressible incrementally in SQL with certain auxiliary relations but is not expressible incrementally in first-order logic with the same auxiliary relations.

In Section 4, we look at the complexity of maintaining transitive closure against the complexity of maintaining other queries. We prove that it is strictly more difficult to maintain the "same generation" query than to maintain transitive closure. We are also able to generalize this result and show that maintaining context-free chain queries (in a certain sense to be defined) is at least as hard as maintaining transitive closure.

In Section 5 we extend our basic technique to show that the same-generation query cannot be maintained (incrementally or decrementally) in SQL-like languages.

2 Recomputation of Recursive Queries in Relational Calculus

The purpose of this section is to show that the transitive closure of a relation cannot be decrementally maintained in relational calculus or first-order logic. That is,

Theorem 2.1 *There is no relational calculus expression that defines the transitive closure $R^+_{-(x,y)}$ of $R - \{(x,y)\}$ in terms of a binary relation R, its transitive closure R^+, and an edge (x,y).*

We introduce a new proof technique that is different from [7]. In particular, our technique does not rely on games and can be readily extended to other queries and languages. For example, we will show that the analog of Theorem 2.1 holds for a language having the expressive power of SQL. Another proof of Theorem 2.1 based on the locality property of first-order logic [10] is given in Appendix A.

2.1 The Proof

Definition 2.2 A **single cycle** is a graph of the form $\{(v_1,v_2),(v_2,v_3),\ldots,(v_{n-1},v_n),(v_n,v_1)\}$, where all v_i are distinct. A **chain** is the $\{(v_1,v_2),(v_2,v_3),\ldots,(v_{n-1},v_n)\}$ part of a single cycle. □

Our first proof is based on a result in first-order logic, called the bounded degree property [13], that formalizes the intuition behind the fact that recursive queries are not first-order definable. It says that it is not possible to define in first-order logic a function that transforms a graph having small in- and out-degrees into a graph having a large number of in- and out-degrees.

Let $G = \langle V, E\rangle$ be a graph (binary relation). Throughout the paper, when we speak of graph queries, we assume that the nodes come from a countably infinite domain, and that equality is the only predicate available on the nodes. (Note the absence of order.) Define $\mathsf{in\text{-}deg}(v) = card(\{v' \mid (v',v) \in E\})$ and $\mathsf{out\text{-}deg}(v) = card(\{v' \mid (v,v') \in E\})$. The degree set $\mathsf{deg}(G)$ of G is defined as $\{\mathsf{in\text{-}deg}(v) \mid v \in V\} \cup \{\mathsf{out\text{-}deg}(v) \mid v \in V\}$.

Definition 2.3 Suppose for any function f from graphs to graphs that is definable in a language $\mathcal{L}$ it is the case that for any number k there is a number c, depending on f and k, such that $card(\mathsf{deg}(f(G))) \leq c$ for any graph G satisfying $\mathsf{deg}(G) \subseteq \{0,1,\ldots,k\}$. Then $\mathcal{L}$ is said to have the **bounded degree property**. □

Theorem 2.4 (see [13]) *First-order logic has the bounded degree property.* □

One may prove that certain general recursive queries, such as transitive closure, cannot be expressed in first-order logic. However, one may find that a very different approach is needed to prove the same result for certain special cases of these queries, such as transitive closure of a chain. (This is in fact a *class* of queries that have the following property: when the input is a chain, they return its transitive closure). Using Theorem 2.4 many first-order inexpressibility results and their special cases can be proved in a simple uniform manner. For example,

Corollary 2.5 *First-order logic cannot express the transitive closure of a chain.*

Proof. The degree set of a chain is $\{0,1\}$. But the degree set of the transitive closure of a chain is $\{0,1,\ldots,n\}$, where n is the length of the chain. Then the corollary follows by Theorem 2.4. □

Making use of Corollary 2.5, we can now present our

Proof of Theorem 2.1. Let R be a single cycle and (x,y) be the edge to be removed. Assume that $R^+_{-(x,y)}$ is definable in first-order logic. That is, there is a formula $\Theta[R, R^+, x, y, u, v]$ such that for all u and v:

$$R^+_{-(x,y)}(u,v) \text{ iff } \Theta[R, R^+, x, y, u, v] \tag{1}$$

In this formula Θ every quantifier is of form $\forall z \in V$ or $\exists z \in V$ where V is the set of nodes of R (and R^+). As the transitive closure of a single cycle is a complete graph, every $R^+(x', y')$ in $\Theta[R, R^+, x, y, u, v]$ can be replaced by *true* so that (1) continues to hold. This replacement results in a formula $\Theta'[R, x, y, u, v]$ in which R^+ does not appear. So we have

$$R^+_{-(x,y)}(u,v) \text{ iff } \Theta'[R, x, y, u, v] \tag{2}$$

Now, we prove that (2) implies that transitive closure of a chain is first-order definable. Let P be a binary relation symbol to be interpreted as a chain. Let $end(y')$ be a first-order formula in terms of P saying that y' has out-degree zero in P and let $start(y')$ be a first-order formula in terms of P saying that y' has in-degree zero in P. Define $\Theta''[P, x, y, u, v]$ as $\Theta'[R, x, y, u, v]$ in which $R(z, z')$ is replaced by $P(z, z') \vee (end(z) \wedge start(z'))$. Let $\Delta[P, u, v]$ be $\exists x \exists y. end(x) \wedge start(y) \wedge \Theta''[P, x, y, u, v]$. According to (2), if P is interpreted as a chain, then $\Delta[P, u, v]$ holds iff there is an edge from u to v in the transitive closure of a graph obtained from P by first inserting an edge from end to start and then deleting it. That is, iff there is an edge in the transitive closure of P. Hence, (2) implies that the transitive closure of a chain can be expressed in first-order logic, contradicting Corollary 2.5. Thus $R^+_{-(x,y)}(u,v)$ is not first-order definable. □

2.2 Generalizations

Our proof is based on three assumptions. First, a query Q, whose unmaintainability we must prove, has the following property: whenever Q is applied to a chain, it produces the transitive closure of that chain. Second, the language must contain a sublanguage as expressive as first-order logic. Third, the transitive closure of a chain is not definable in the language. Summing up, we see from the proof of Theorem 2.1 that the following is true.

Theorem 2.6 *Let $\mathcal{L}$ be a language that contains a sublanguage as expressive as relational calculus, but that cannot express the transitive closure of a chain. Let Q be any graph query that computes the transitive closure of a chain. Then it is impossible to write a query in $\mathcal{L}$ that, when applied to a binary relation R, an edge (x,y) in R, and $Q(R)$, will produce $Q(R - \{(x,y)\})$. In other words, Q cannot be decrementally maintained in $\mathcal{L}$.* □

One immediate application of this result is the impossibility of decremental recomputation of queries that are simpler than transitive closure. Transitive closure together with first-order logic captures the complexity class NLOGSPACE. Deterministic transitive closure, which closes paths whose nodes have outdegree one, captures the class DLOGSPACE, when added to the first-order logic [11]. Since deterministic and the usual transitive closures are equivalent on chains, we obtain

Corollary 2.7 *It is impossible to recompute deterministic transitive closure decrementally using relational calculus.* □

2.3 Using Auxiliary Relations

Our negative result merely says that the transitive closure R^+ of a relation R cannot be maintained using relational calculus when an edge (x,y) is removed from that relation. It says nothing about whether it is possible to maintain R^+ when some additional auxiliary relations are present.

The availability of auxiliary relations – which must themselves be maintainable using relational calculus under edge insertion and deletion – can make a difference. For example, Dong and Su [5] show that for a special kind of cyclic graphs R, where there is at most one path connecting any two vertices, R^+ can be maintained when an auxiliary relation storing a maximal acyclic subgraph of R is provided. However, the general situation remains open.

Open Problem 2.8 Is there a k-ary property A_R of a binary relation R such that both $R^+_{-(x,y)}$ and $A_{R-\{(x,y)\}}$ can be expressed using first-order formulae in terms of R, R^+, A_R, and (x, y)?

Note that a property is required to be name-independent or *generic*: if $A_R(x_1, \ldots, x_n)$ holds, then $A_{\pi(R)}(\pi(x_1), \ldots, \pi(x_n))$ is required to hold for any automorphism π of R's nodes.

For $k = 1$ (unary predicates), this problem is easily resolved.

Theorem 2.9 *There is no unary property A_R of a binary relation R such that both $R^+_{-(x,y)}$ and $A_{R-\{(x,y)\}}$ can be expressed using first-order formulae in terms of R, R^+, A_R, and (x, y).*

Proof. Assume that R is a single cycle. Because of genericity, either A_R holds for each node of R, or A_R fails for each node of R, for any single cycle R. Now, suppose the unary property A_R exists and $\Psi[R, R^+, A_R, x, y, u, v]$ is a first-order formula such that $R^+_{-(x,y)}(u, v)$ iff $\Psi[R, R^+, A_R, x, y, u, v]$. Since R is a cycle, we can replace each occurrence of $R^+(v, w)$ by *true* obtaining $\Phi[R, A_R, x, y, u, v]$ such that

$$R^+_{-(x,y)}(u, v) \text{ holds} \quad \text{iff} \quad \Phi[R, A_R, x, y, u, v] \text{ holds} \tag{3}$$

Since A_R is generic, either $\{m \mid \exists$ single cycle R of length m such that A_R holds for every node in $R\}$ or $\{m \mid \exists$ single cycle R of length m such that A_R does not hold for any node in $R\}$ is infinite. Assume without loss of generality that the former is the case. Then, for any m, there is a single cycle R of length $\geq m$ such that A_R holds for every node in R. Let Φ' be obtained from Φ be replacing $A_R(v)$ by *true*. Then, according to (3), for infinitely many non-isomorphic single cycles, $R^+_{-(x,y)}(u, v)$ holds iff $\Phi'[R, x, y, u, v]$ where Φ' now is a first-order formula.

Using this fact and the same argument as in the second proof of Theorem 2.1, we obtain a first-order formula that, for infinitely many non-isomorphic chains, computes their transitive closure. But this contradicts the bounded degree property, thus proving the theorem. □

For the situation where the arity of auxiliary relations is 2 or greater, we can only resolve certain special cases. For example, let $E_R(x, y, u, v)$ be a 4-ary property saying that every path in R from u to v must go through the edge (x, y), where $x = u$ and $y = v$ do not hold simultaneously. It is possible to maintain the transitive closure of some cyclic relations using this E_R. That is, $R^+_{-(x,y)}$ can be defined in terms of R, R^+, E_R, and (x, y). However, we have to address the decremental maintenance of E_R itself. Again, relational calculus does not provide us with sufficient expressive power to do so.

Theorem 2.10 *There exists a class of relations R such that neither R^+ nor E_R can be maintained decrementally.*

Proof. Consider the following relation R. Let R_0 be a single cycle. R is obtained from it by adding two new nodes, x and y, the edge (x, y), and edges (z, x), (x, z), (z, y), (y, z) for all z on the cycle R_0. Note that E_R is empty because there are at least two alternative paths between any two nodes in R that are not connected by an edge.

The inability to maintain R^+ decrementally, given R, R^+ and E_R follows immediately. Since R^+ is a complete graph and E_R is empty, they can be safely replaced by constants. If the transitive closure of R were decrementally maintainable, we could have written a first-order formula for calculating the transitive closure of a chain obtained from R by deleting (x, y) and then any edge (z, z') on the cycle. To see that this impossible, consider an arbitrary chain C with endpoints x and y, and define the new graph R by making $C - \{x, y\}$ into a cycle and adding edges (z, x), (x, z), (z, y), (y, z) for all $z \in C - \{x, y\}$. Maintainability of R^+ now implies that the transitive closure of $C - \{x, y\}$ is first-order definable, but this contradicts the bounded degree property.

To show that E_R cannot be maintained decrementally, note that if it were, we would be able to define, in relational calculus, $E_{R'}$, for R' a single cycle. Indeed, if such $E_{R'}$ were definable, we would be able to define the transitive closure of a chain by adding the edge from the end to the start to make a single cycle and then, for any two nodes x and y, checking if this added edge is on the path from x to y. □

3 Recomputation of Recursive Queries in SQL

In the preceding sections, first-order logic was used as the ambient language. However, real database query languages are richer and more powerful than relational algebra. In particular, the de-facto practical query language SQL has a GROUP-BY operator, arithmetic operators, and aggregate functions. These extra primitives give these languages extra power; for example, they can test whether a set of numbers has even or odd cardinality [12]. In this section, we consider decremental recomputation of transitive closure in these more practical languages. First, we define a "theoretical SQL", which is a formally presented language that has the main features that distinguish SQL from relational calculus. Then we prove analogs of our negative results for this language.

3.1 A Theoretical Reconstruction of SQL

As we mentioned, there are two main features that make SQL more expressive that first-order logic. First, SQL allows nesting by using the GROUP-BY operator. Second, SQL has arithmetic operations and a number of aggregate functions.

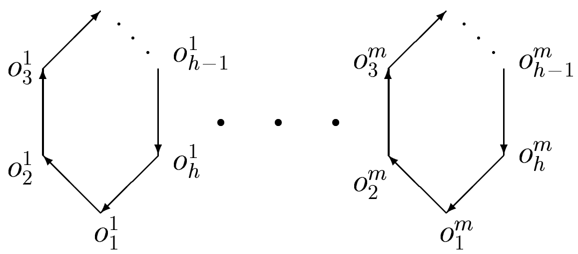
It was proposed in [12] to use the language $\mathcal{NRL}^{\text{aggr}}$ as a theoretical language to study the expressive power of SQL. It is obtained from the nested relational algebra (which is a standard language for manipulating complex objects, see [3, 4, 15]) by adding simple rational arithmetic operators $+$, $-$, $\cdot$, and $\div$; a summation construct $\sum(f)(\{x_1, \ldots, x_n\}) = f(x_1) + \ldots + f(x_n)$; and the usual linear order on rational numbers $\leq^{\mathsf{rat}}$. It was shown [12] that $\mathcal{NRL}^{\text{aggr}}$ can express aggregate functions and grouping operations found in SQL. Moreover, the same paper showed that expressibility in $\mathcal{NRL}^{\text{aggr}}$ is independent of the depth of nesting of sets in intermediate data.

This implies, under the normal semantics used in database query languages, that a query in $\mathcal{NRL}^{\text{aggr}}$ over flat relations can always be translated into a query in first-order logic augmented with the same operators. In other words, $\mathcal{NRL}^{\text{aggr}}$ and (most variations of) SQL coincide in expressible power, as far as flat relations are concerned. Thus we can investigate decremental recomputation of recursive queries in the context of $\mathcal{NRL}^{\text{aggr}}$ instead.

3.2 Expressive Power of SQL

The main result of this section is that decremental recomputation of transitive closure remains inexpressible in $\mathcal{NRL}^{\text{aggr}}$. Our proof uses a finite-cofiniteness result in nested relational algebra [13], which says that it is impossible to define a query in $\mathcal{NRL}^{\text{aggr}}$ that distinguishes members of a family of graphs called k-multi-cycles.

Definition 3.1 A binary relation is called a **k-multi-cycle** if it is nonempty and is of the form shown below, where $h \geq k$ and o_i^j are all distinct. That is, it is a graph containing $m \geq 1$ unconnected cycles of equal length $h \geq k$. Recall that the vertices come from a countably infinite domain on which only the equality test is available. □



Theorem 3.2 (see [13]) *Let f be a Boolean query on graphs definable in $\mathcal{NRL}^{\text{aggr}}$. Then there is a k such that either $f(R)$ is true for every k-multi-cycle R, or $f(R)$ is false for every k-multi-cycle R.* □

In essence, this theorem says that we cannot use $\mathcal{NRL}^{\text{aggr}}$ to distinguish one k-multi-cycle from another, provided the cycles are sufficiently long. Note that this result remains valid even when we restrict ourselves to subclasses of k-multi-cycles having some fixed number of cycles. For example, it continues to hold when we replaced k-multi-cycles by single cycles of length at least k. Many inexpressibility results can be obtained from this theorem in a trivial manner. In particular,

Corollary 3.3 $\mathcal{NRL}^{\text{aggr}}$ *is unable to express the transitive closure of a chain.*

Proof. Suppose the transitive closure of a chain can be expressed in this language. Then testing whether a graph is a chain can also be expressed in this language; see [13]. Then testing whether a k-multi-cycle contains exactly one cycle can be expressed, too. However, this contradicts Theorem 3.2. Therefore, the transitive closure of a chain is inexpressible in $\mathcal{NRL}^{\text{aggr}}$. □

The main result of this section follows immediately from Theorem 2.6 and Corollary 3.3.

Theorem 3.4 *It is impossible to recompute (deterministic) transitive closure decrementally in* $\mathcal{NRL}^{\text{aggr}}$. □

One has to be careful when formulating an analog of Open Problem 2.8 for $\mathcal{NRL}^{\text{aggr}}$. If we simply ask whether it is possible to maintain transitive closure decrementally using additional space, the answer to this question is positive. Indeed, as auxiliary object we store a (nested) relation that contains transitive closure for each subrelation of R, together with a list of edges deleted from R to obtain this subrelation. Then transitive closure can easily be recomputed. However, this naive approach requires exponential space. We still do not know whether the following is true.

Open Problem 3.5 *Is it possible to recompute transitive closure decrementally in SQL using polynomial auxiliary space?*

One instance of this problem can be resolved using the methods as in the proof of Theorem 2.9.

Theorem 3.6 *There is no unary property A_R of a binary relation R such that both $R^+_{-(x,y)}$ and $A_{R-\{(x,y)\}}$ can be expressed using $\mathcal{NRL}^{\text{aggr}}$ in terms of R, R^+, A_R, and (x,y).* □

3.3 SQL vs Relational Calculus

In this subsection we compare the incremental evaluation power of $\mathcal{NRL}^{\text{aggr}}$ with that of first-order logic and with that of $\mathcal{NRL}^{\text{aggr}}$ without incremental evaluation. In particular, we provide an example query that is expressible in the first language but not in the other two languages.

The example is as follows. Let S and R be two arbitrary graphs. Let x and y be two arbitrary nodes in S and R respectively. Does x reach more nodes than y? We denote this query by $\mathsf{moreprolific}_{S,R}(x,y)$.

Proposition 3.7 *Let S and R be two graphs. Let (u,v) be an edge to be inserted in S. Then*

1. $\mathsf{moreprolific}_{S,R}(s,t)$ *is inexpressible in $\mathcal{NRL}^{\text{aggr}}$.*
2. $S^+_{+(u,v)}$ *and* $\mathsf{moreprolific}_{S\cup\{(u,v)\},R}$ *can be expressed in $\mathcal{NRL}^{\text{aggr}}$ in terms of S, R, S^+, R^+, and* $\mathsf{moreprolific}_{S,R}$.
3. $\mathsf{moreprolific}_{S\cup\{(u,v)\},R}$ *cannot be expressed in first-order logic in terms of S, R, S^+, R^+, and* $\mathsf{moreprolific}_{S,R}$.

Proof.

1. If R is a chain, then $R^+(x,y)$ iff $\mathsf{moreprolific}_{R,R}(x,y)$.
2. We note that $S^+_{+(u,v)}$ is expressible in $\mathcal{NRL}^{\text{aggr}}$ using (u,v) and S^+. Then we use the following steps to test if (x,y) should be included in $\mathsf{moreprolific}_{S\cup\{(u,v)\},R}$. First, select from $S^+ + (u,v)$ the set of nodes reachable from x. Second, select from R^+ the set of nodes reachable from y. Third, test if the cardinality of the first set exceeds that of the second step. This third and last step is expressible in $\mathcal{NRL}^{\text{aggr}}$.
3. Let S and R be two complete graphs with S being smaller or equal in size to R. Then S^+ and R^+ are complete graphs and $\mathsf{moreprolific}_{S,R}$ is empty. Let u be a node in S and v be new. Then $\mathsf{moreprolific}_{S\cup\{(u,v)\},R}$ is nonempty iff S is equal in size to R. If the latter can be expressed in first-order logic in terms of (u,v), S, and R then so can the test of whether two sets are equal in cardinality, which is known to be inexpressible [9] in first-order logic. □

Therefore, using $\mathcal{NRL}^{\text{aggr}}$ in an incremental/decremental way does provide some desirable extra expressive power.

4 Context-Free Chain Queries

A frequently considered class of queries is the class of chain queries. Consider databases as labeled graphs. A chain query $Q_{L,G}$ retrieves pairs of nodes between which there exists a walk (edge sequence, not necessarily a path) in G having some particular kind of label patterns specified by a context-free set L. For example, for the transitive closure query the patterns are specified by the regular set $\{A^n \mid n > 0\}$, where A is a unique label.

We also consider the *same-generation* query over a graph G having two label symbols A and B. Such a graph can be conveniently represented by two relations, one for edges labeled A and the other for edges labeled B, which need not be disjoint. We use A and B to name these two relations. Then we write $\mathsf{SG}_{A,B}(x,y)$ iff there is a z such that there is a walk from x to z in A and a walk from z to y in B that are equal in length. The same-generation is a chain query; it is specified by the context-free set $\{A^n B^n \mid n > 0\}$.

It is known that every chain query whose label patterns are specified by a regular set allows query recomputation after insertion using first-order logic [6, 8]. It is open whether this is true for any chain query whose label patterns are specified by an arbitrary context-free set. The same-generation query, as a special case of a context-free set, is known to be unmaintainable using first-order logic under insertion of edges if auxiliary relations are not allowed [8] and this remains true when unary auxiliary relations are allowed [7]. A generalization of these two results to SQL-like languages is given in Section 5. It is an open problem if incremental maintenance of the same-generation query is possible when polynomial auxiliary space are allowed.

The purpose of this section is to compare the complexity of decremental recomputation of transitive closure with that of context-free chain queries, in particular, the same-generation query.

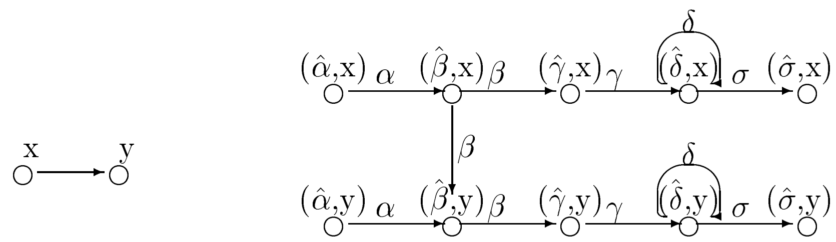
Definition 4.1 *Given two graph queries, Q_1 and Q_2, we say that*

1. *Q_1 is* **strictly harder** *than Q_2 if, whenever Q_1 can be decrementally maintained in a language $\mathcal{L}$ having relational calculus as a sublanguage, then so can Q_2, and there is an infinite class of graphs on which Q_2 can be decrementally maintained in relational calculus, but Q_1 cannot.*
2. *Q_1 is* **linearly easier** *than Q_2 if there exists a linear function f such that Q_1 can be decrementally maintained with auxiliary relations of arity up to $f(k)$ whenever Q_2 can be decrementally maintained with auxiliary relations of arity up to k.* $\Box$

Proposition 4.2 *The same-generation query is strictly harder than the transitive closure query.*

Proof. The same-generation query over acyclic databases cannot be decrementally maintained using first-order logic with at most unary auxiliary relations [7]. The transitive closure query over acyclic databases can be decrementally maintained using first-order logic with no auxiliary relations [5]. To conclude the proof, observe that $R^+(x,y)$ iff $\mathsf{SG}_{R,\{(v,v) \mid v \text{ is a node in } R\}}(x,y)$. That is, we use R as the A relation and $\{(v,v) \mid v$ is a node of $R\}$ as the B relation of $\mathsf{SG}_{A,B}$. If we can maintain $\mathsf{SG}_{A,B}$, then we can clearly maintain R^+. $\Box$

Theorem 4.3 *The transitive closure query is linearly easier than every chain query $Q_{L,G}$, where L is infinite.*



(a) An edge (b) Corresponding graph

Figure 1: Transforming graph for R^+ to graph for $Q_{L,G}$

Proof Sketch. Suppose L is an infinite context-free set. Then by the pumping lemma, there exist strings α, β, γ, δ, σ such that $\beta\delta$ is not empty and the set $\{\alpha\beta^n\gamma\delta^n\sigma \mid n \geq 1\}$ is contained in L. We maintain the transitive closure of R by maintaining the answer to $Q_{L,G}$ using the reduction described below and depicted in Figure 1.

First assume that α, β, γ, δ, and σ are letters. Then we use five constants, named $\hat{\alpha}$, $\hat{\beta}$, $\hat{\gamma}$, $\hat{\delta}$, and $\hat{\sigma}$, to tag the nodes in R. Corresponding to each edge (x, y) in R, we have the edges given in Part (b) of the figure. Let G be the resulting graph of this reduction.

Clearly, this reduction can be maintained in first-order logic. Furthermore, $R^+(x, y)$ iff (1) $x \neq y$ and $Q_{L,G}((\hat{\alpha}, x), (\hat{\sigma}, y))$, (2) $x = y$ and $R(x, y)$, or (3) $x = y$ and there is $z \neq x$ such that $Q_{L,G}((\hat{\alpha}, x), (\hat{\sigma}, z))$ and $Q_{L,G}((\hat{\alpha}, z), (\hat{\sigma}, y))$.

From the above, the following upper bounds are clear. If the first-order logic formulae that maintain $Q_{L,G}$ do not use auxiliary relations, then the formulae for maintaining transitive closure constructed by this reduction need only a 4-ary auxiliary relation. If the first-order logic formulae that maintain $Q_{L,G}$ use at most k-ary auxiliary relations, then the formulae for maintaining transitive closure constructed by this reduction need at most $2k$-ary auxiliary relations, plus a 4-ary auxiliary relation. Hence f can be taken to be a linear function dominating $\max(2k, 4)$.

For the situation where the length of $\alpha\beta\gamma\delta\sigma$ is m and some of α, β, γ, δ, σ is not a letter, we employ m different constants to tag the nodes of R using a similar construction, with minor adjustments if some of α, β, γ, δ, σ is empty. $\square$

Finally, we observe that all the transformations employed earlier are first-order and thus

Corollary 4.4 *Analog of Theorem 4.3 holds when we use the flat fragment of $\mathcal{NRL}^{\text{aggr}}$ instead of first-order logic.* $\square$

5 Impossibility of Recomputation of Same-Generation in SQL

It was previously known that the same-generation query cannot be maintained, without auxiliary space, using first-order logic when edges are inserted or deleted [7]. These previous results were proved using games and thus cannot be extended to SQL-like languages.

In this section, we give a further demonstration of the advantage of our technique. We extend the above results to any language that contains first-order logic as a sublanguage but is finite-cofinite on single-cycle queries. Then the inability of SQL to maintain the same-generation query without auxiliary space falls out as an immediate corollary.

Definition 5.1 A language $\mathcal{L}$ is called **cycle-simple** if it contains relational calculus as a sublanguage but cannot express properties of single cycles that are both infinite and coinfinite. That is, for any Boolean query f in $\mathcal{L}$, there exists a number n such that either f is true for any single cycle of length $\geq n$, or f is false for any single cycle of length $\geq n$. □

Let us first obtain a property on a special kind of graphs called two-chain graph that is true of any cycle-simple language.

Definition 5.2 A **two-chain graph** is a graph consisting of two unconnected chains. □

Let P be the following property of two-chain graphs: If lengths of two chains are n and m, then either $n = m$, or $n = m - 2$, or $n = m + 2$.

Proposition 5.3 *Let $\mathcal{L}$ be a cycle-simple language. Then $\mathcal{L}$ cannot test the property P of two-chain graphs.*

Proof. Note that a single cycle G of length more than 6 is of even length iff the following is true. There are non-consecutive edges in G such that removal of them from G results in a two-chain graph satisfying P. Thus, if P were $\mathcal{L}$-definable, being of even length can be tested on single cycles of length more than 6 as follows:

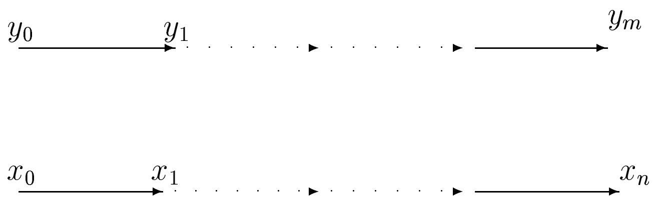
$$\exists x \exists y \exists u \exists v. E(x,y) \& E(u,v) \& \neg E(y,u) \& \neg E(v,x) \& \neg(x=u) \& \neg(y=v) \& P(E')$$

where $E'(z,w)$ is $E(z,w) \wedge \neg((z = x \& w = y) \vee (z = u \& w = v))$. But this contradicts the cycle-simplicity of $\mathcal{L}$. □

Using this result, we can prove that such a language $\mathcal{L}$ cannot increment the same-generation query.

Theorem 5.4 *Let $\mathcal{L}$ be a cycle-simple language. Then it cannot maintain the same-generation query under edge insertion without using auxiliary relations.*

Proof. Let R^c denote the graph obtained by reversing the edges in R. That is, $R^c(u,v)$ iff $R(v,u)$. Consider the two-chain graph R depicted below. Then SG_{R,R^c} is a diagonal graph. That is $\mathsf{SG}_{R,R^c}(u,v)$ iff $u = v$ and u is a node of R.



Suppose $\mathcal{L}$ can maintain the same-generation query when edges are inserted. Then we have a formula $\Theta[\mathsf{SG}_{R,R^c}, R, x, y, u, v]$ in $\mathcal{L}$ such that $\mathsf{SG}_{R\cup\{(x,y)\},R^c\cup\{(y,x)\}}(u,v)$ iff $\Theta[\mathsf{SG}_{R,R^c}, R, x, y, u, v]$.

Since SG_{R,R^c} is a diagonal, every occurrence of $\mathsf{SG}_{R,R^c}(x',y')$ in $\Theta[\mathsf{SG}_{R,R^c}, R, x, y, u, v]$ can be replaced by the formula $(x' = y') \wedge (\exists z'.R(x',z') \vee R(z',x'))$. So we obtain a formula $\Theta'[R, x, y, u, v]$ in which SG_{R,R^c} does not appear such that $\mathsf{SG}_{R\cup\{(x,y)\},R^c\cup\{(y,x)\}}(u,v)$ iff $\Theta'[R, x, y, u, v]$.

Now consider the following formula

$$\Psi(u,v) \;\equiv\; [\neg\exists w.E(u,w)] \;\&\; [\exists w.E(v,w)] \;\&\; [\forall w.(E(v,w) \rightarrow (\neg(w = u) \& (\neg\exists w'.E(w,w'))))]$$

This formula says that u is an endpoint (has outdegree zero), and v is a predecessor of an endpoint different from u. Let $start(x)$ be $\neg\exists w.E(w,x)$. Consider the sentence

$$\Phi \equiv \exists x_0 \exists y_0 \exists u \exists v.start(x_0) \ \& \ start(y_0) \ \& \ \neg(x_0 = y_0) \ \& \ \Psi(u,v) \ \& \ (\Theta'[R,x_0,y_0,u,v] \vee \Theta'[R,y_0,x_0,u,v])$$

This sentence is satisfied by the two-chain graph R shown above iff one of the following holds.

(a) In a graph obtained from R by adding (x_0,y_0), either x_{n-1} and y_m are of the same generation (then $n = m+2$) or y_{m-1} and x_n are of the same generation (then $n = m$); or

(b) In a graph obtained from R by adding (y_0,x_0), either x_{n-1} and y_m are of the same generation (then $n = m$) or y_{m-1} and x_n are of the same generation (then $m = n+2$).

In other words, Φ holds iff R has the property P. Since $\mathcal{L}$ contains a sublanguage equivalent to the relational calculus and all transformations above are first-order, we conclude that under the assumption that SG is incrementally maintainable, $\mathcal{L}$ can test the property P. But this contradicts proposition 5.3. This proves the theorem. □

Not only is $\mathcal{L}$ unable to maintain the same-generation query under edge insertion, it is also unable to maintain the same-generation query under edge deletion.

Theorem 5.5 *Let $\mathcal{L}$ be a language that contains relational calculus as a sublanguage but cannot express the transitive closure of a chain. Then it cannot maintain the same-generation query under edge deletion without using auxiliary relations.*

Proof sketch. Suppose R is a single cycle having an odd number of edges. Then it is easy to see that $\mathsf{SG}_{R,R}$ is a complete graph. Consider deleting an edge (x,y) from R. Notice that the transitive closure of the chain $R - \{(x,y)\}$ is expressible as $R^+_{-(x,y)}(u,v)$ iff $\mathsf{SG}_{R-\{(x,y)\},R-\{(x,y)\}}(u,v)$, or $\mathsf{SG}_{R-\{(x,y)\},R-\{(x,y)\}}(u,z)$ and $R(z,v)$ and $(z,v) \neq (x,y)$, or $R(u,v)$ and $(x,y) \neq (u,v)$.

If $\mathcal{L}$ can maintain the same-generation query when edges are deleted, then we have a formula $\Theta[R,\mathsf{SG}_{R,R},x,y,u,v]$ in $\mathcal{L}$ such that for all u and v:

(4) $$\mathsf{SG}_{R-\{(x,y)\},R-\{(x,y)\}}(u,v) \text{ iff } \Theta[R,\mathsf{SG}_{R,R},x,y,u,v]$$

Since $\mathsf{SG}_{R,R}$ is a complete graph, we can replace every occurrence of $\mathsf{SG}_{R,R}(x',y')$ by *true* so that (4) continues to hold. This replacement results in a formula $\Theta'[R,x,y,u,v]$ in which $\mathsf{SG}_{R,R}$ does not appear. So we have $\mathsf{SG}_{R-\{(x,y)\},R-\{(x,y)\}}(u,v)$ iff $\Theta'[R,x,y,u,v]$. This implies that the transitive closure of the chain $R - \{(x,y)\}$ is definable in $\mathcal{L}$. It is not hard to show that then transitive closure of any chain is definable in $\mathcal{L}$, which is a contradiction. □

Combining Theorem 5.4, Theorem 3.2, Theorem 5.5, and Corollary 3.3, we conclude that SQL-like languages cannot maintain the same-generation query under insertion and deletion of edges.

Corollary 5.6 $\mathcal{NRL}^{\mathrm{aggr}}$ *cannot maintain the same-generation query under insertion and deletion of edges without using auxiliary relations.* □

6 Remarks

Our proof of impossibility of decremental maintenance of transitive closure assumes the existence of a formula that decrements transitive closure. It then analyzes this formula under the assumption that the relation supplied is a single cycle. By making this assumption, it is allowed to simplify the formula. The resultant is a formula that expresses a query that is shown to be inexpressible in first-order logic.

An additional interest in this proof is its use of a recently-developed general result called the bounded degree property [13]. More significantly, it generalizes naturally to powerful and practical SQL-like languages by substituting the bounded degree property of first-order logic with the finite-cofiniteness of queries on k-multi-cycles in these augmented languages [13].

The ease of extension from proofs for first-order logic to SQL-like languages is a major distinguishing feature of the techniques in this paper from that of Dong and Su [7]. One advantage of the techniques here is that they extend to SQL-like languages having arithmetics and aggregate functions readily, whereas the games technique [7] does not. In Section 5 we demonstrated this advantage again and proved that the same-generation query cannot be maintained using an SQL-like language when edges are added or deleted.

Another proof of Theorem 2.1 is given in Appendix A. It also uses the idea of single cycle, but is quite different in flavor from the first. It assumes that there is a formula for decrementing transitive closure. It does not modify nor analyze this formula in any way. Instead, it uses this formula to construct a query on single cycle that is shown to be inexpressible in first-order logic.

Finally, the general case of maintaining transitive closure under edge deletion where no limit is placed on auxiliary relations (or polynomial space limit is placed in the case of nested relations) remains an interesting unsolved problem.

Acknowledgements. Wong was supported in part by the Real World Computing Novel Function Laboratory at the Institute of Systems Science. Wong would also like to thank the University of Melbourne and fellow coauthor Dong for their hospitality during this work. Dong would like to acknowledge the support of the Australian Research Council.

References

[1] A. Aho and J. Ullman. Universality of data retrieval languages. In *Proceedings 6th Symposium on Principles of Programming Languages, Texas, January 1979*, pages 110–120, 1979.

[2] V. Breazu-Tannen, P. Buneman, and S. Naqvi. Structural recursion as a query language. In *Proceedings of 3rd International Workshop on Database Programming Languages, Naphlion, Greece*, pages 9–19. Morgan Kaufmann, August 1991.

[3] V. Breazu-Tannen, P. Buneman, and L. Wong. Naturally embedded query languages. In *Proceedings of 4th International Conference on Database Theory, Berlin, Germany*, pages 140–154, October 1992.

[4] L. S. Colby. A recursive algebra for nested relations. *Information Systems*, 15(5):567–582, 1990.

[5] G. Dong and J. Su. Incremental and decremental evaluation of transitive closure by first-order queries. *Information and Computation*; to appear. Preliminary version appeared in 1993 in the *Proceedings of 16th Australian Computer Science Conference.*

[6] G. Dong and R. Topor. Incremental evaluation of datalog queries. In *LNCS 646: Proceedings of 4th International Conference on Database Theory, Berlin, Germany, October 1992*, pages 282–296.

[7] G. Dong and J. Su. Space-bounded FOIES. In *Proceedings of 14th ACM Symposium on Principles of Database Systems, San Jose, California*, May 1995, pages 139–150.

[8] G. Dong, J. Su, and R. Topor. Nonrecursive incremental evaluation of Datalog queries. *Annals of Mathematics and Artificial Intelligence*; to appear.

[9] H. Enderton. *A Mathematical Introduction to Logic.* Academic Press, San Diego, 1972.

[10] H. Gaifman. On local and non-local properties. In *Proceedings of the Herbrand Symposium, Logic Colloquium '81*, pages 105–135. North Holland, 1982.

[11] N. Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, 16:760–778, 1987.

[12] L. Libkin and L. Wong. Aggregate functions, conservative extension, and linear orders. In *Proceedings of 4th International Workshop on Database Programming Languages, New York*, pages 282-294, August 1993.

[13] L. Libkin and L. Wong. New techniques for studying set languages, bag languages, and aggregate functions. In *Proceedings of 13th ACM Symposium on Principles of Database Systems, Minneapolis, Minnesota*, pages 155–166, May 1994.

[14] S. Patnaik and N. Immerman. Dyn-FO: A parallel dynamic complexity class. In *Proceedings of 13th ACM Symposium on Principles of Database Systems, Minneapolis, Minnesota*, pages 210–221, May 1994.

[15] H.-J. Schek and M. H. Scholl. The relational model with relation-valued attributes. *Information Systems*, 11(2):137–147, 1986.

Appendix

A Another Proof of Theorem 2.1

Our second proof of Theorem 2.1 is based on a different set of techniques and proceeds in two steps. First, we use the locality result for first-order logic [10] to show that it is not possible to express whether a given edge lies on every path between two vertices in an extremely simple cyclic graph. Then we show that if transitive closure can be properly updated using relational calculus when edges are removed, it would contradict the first result.

Proposition A.1 *Let $E[R, x, y, u, v]$ be a first-order logic formula with x, y, u, and v as free variables and R as a binary relation symbol. Then there is a constant m such that for every single cycle R, for every pair of vertices u and v at least m edges apart in R, it is not the case that for every edge (x, y) in R, $E[R, x, y, u, v]$ iff (x, y) is an edge on every path from u to v in R.*

Proof. According to Gaifman [10], on all single cycles of a length $\geq m_0$, where m_0 is a constant depending on E, $E[R, x, y, u, v]$ is equivalent to a Boolean combination of formulae in free variables x, y, u, v such that all quantifiers are bounded. That is, they are of form $Qz \in N_r(w)$ where Q is $\forall$ or $\exists$, $w \in \{x, y, u, v\}$ and N_r denotes a neighborhood of radius r, i.e. the set of nodes reachable by path of length at most r.

Notice that in a single cycle an r-neighborhood consists of $2r + 1$ nodes. Let r_{max} be the maximal radius in all quantifiers used in E. Let $m = \max\{m_0, 8r_{max} + 4\}$. Then, on every single cycle O of length at least m, we can find four nodes n_1, n_2, n_3, n_4 such that their r-neighborhoods, $r \leq r_{max}$, are disjoint chains of length $2r + 1$. Then, given the form of E, either it must hold on O for all possible assignments of n_i's to x, y, u, v, or it must fail for all such assignments. Hence, $E[R, x, y, u, v]$ cannot express the fact that (x, y) is an edge on every path from u to v. Thus the proposition is proved. □

Now we conclude the proof of Theorem 2.1. Suppose to the contrary that there is a formula $\Delta[R, R^+, x, y, u, v]$ such that $\Delta[R, R^+, x, y, u, v]$ iff (u, v) is in the transitive closure $R^+_{-(x,y)}$, provided R^+ is the transitive closure of R and (x, y) is the edge in R being removed.

Then it is possible to define a formula $\Phi[R, R^+, x, y, u, v]$ such that $\Phi[R, R^+, x, y, u, v]$ iff every path from u to v in R goes through the edge (x, y), provided R^+ is the transitive closure of R and (x, y) is an edge in R. This can be done by noting that $\Phi[R, R^+, x, y, u, v]$ holds iff there is a path from u to v in R before the edge (x, y) is removed but there is no path from u to v in R after the edge (x, y) is removed. That is,

$$\Phi[R, R^+, x, y, u, v] \text{ holds} \quad \text{iff} \quad R^+(u, v) \wedge \neg\Delta[R, R^+, x, y, u, v] \text{ holds}$$

Now we observe that for any single cycle R, its transitive closure is just the cartesian product of all its vertices. Therefore R^+ is definable in first-order logic for any single cycle R.

This observation and the preceding fact imply that, for any single cycle R, we can define a formula that checks if every path from u to v in R contains the edge (x, y). However, this contradicts Proposition A.1. Therefore, the formula $\Delta[R, R^+, x, y, u, v]$ does not exist. □