

Evaluation of Extended Recursive Queries in Deductive Databases¹

Sang Ho Lee

Database Section

Electronics and Telecommunications Research Institute

Dea Duk Science Town

Dea-Jeon, KOREA

Lawrence J. Henschen

Dep. of Electrical Eng. and Computer Sci.

Technological Institute

Northwestern University

Evanston, IL 60208, U.S.A.

Abstract

In order to extend the expressive power of deductive databases, there have been efforts to allow existential quantifiers in the IDB rule of Prenex Normal Form to occur. A formula which can have an existential quantifier in front in a restricted way is defined as an *extended rule*. With the extended rule, we can easily define a virtual view which requires a division operation of relational algebra. Even though there have been numerous efforts to answer recursive queries, the formula under investigation so far is assumed to be free of existential quantifiers in the beginning so that no Skolem functions are allowed to occur in the formula. This paper addresses an extended recursive query evaluation where at least one formula in a recursive rule set is of an extended rule. We investigate reducible recursion as well as four cases of non-reducible recursion of transitive closure and linear recursion type. We have found that occurrences of existentially quantified variables in the extended recursive body predicate dramatically limit the level of recursive search. The number of iterations to answer extended recursive queries can be determined by the structure of the extended recursive rule set.

1 Introduction

Handling recursive queries in the deductive database has drawn a lot of attention over the past decade among researchers in the database and A.I. community, and as a result there are a number of techniques proposed [8,7,15,2,17]. A major portion of the research is focused on dealing with the transitive closure and the linear recursive formula which are generally believed to fall into the

practical class of recursion. To our knowledge, all works assume that the recursive formula under investigation is free of existential quantifiers in the beginning, therefore do not allow Skolem functions to occur in the formula. The formula in a database has been assumed to be of the form $(\forall \vec{x})(\forall \vec{y}) [P(\vec{x}) \leftarrow \vec{A}(\vec{x}, \vec{y})]$.

On the other hand, it has been noted [13,11] that a formula with universal quantifiers only in front has a limited expressive power, and that we need to allow an existential quantifier to occur in the formula in front. The necessity of existential quantifiers in the IDB is evident when we try to define a view which requires a division operation of relational algebra [16]. For example, a virtual view, a Ph/D candidate which can be defined as a student who has taken all core courses and has passed a qualifying examination is expressed as below.

$(\forall x)(\exists y) [Candidate(x) \leftarrow Taken(x, y) \wedge Passed(x)]$.

Such a view cannot be properly expressed by a formula with universal quantifiers in front only. Controlled occurrences of existential quantifiers in the IDB rule can enhance the expressive power of deductive databases without introducing a null value even though existential quantifiers have been generally proposed to represent the null value (exists but unknown).

In order to extend the expressive power of deductive databases, Lee and Henschen [11] introduce two types of IDB, namely an *acceptable rule* and an *extended rule*², which can have existential quantifiers in the IDB rule in a restricted way such that no null value is introduced. The precious semantics of existential quantifiers in a IDB rule is also identified by the name of *expansion form* [11].

As for the inference rule of deductive databases, the resolution principle [14] or its refinements like SLD have been used so far. If we apply resolution or its refinements to an extended rule, an existentially quantified variable of the rule is transformed into a Skolem function in Skolemization phase. We have found that the resolution-based deduction is not suitable for an extended rule due to Skolem functions [10]. Lee and Henschen [12] propose a new inference rule, namely *substitution rule* which replaces the role of the resolution principle in deduction phase. We have shown that the substitution rule is sound and (answer) complete [12].

¹This work was supported in part by the National Science Foundation under Grant DCR-860-8311

²In this paper, only an *extended rule* is considered.

Another technique to handle existential quantifiers in a formula of Prenex Normal Form is suggested by Lloyd and Topor [13]. Their handling mechanism basically transforms an extended program into a general program which is to be executed under an ordinary Prolog system. The transformation is done by use of the equivalence $(\exists x)\alpha \equiv \neg(\forall x)\neg\alpha$ (where α denotes variables for expression [9]). The transformed clause always has a negative literal in the body, so evaluation is carried out under Negation as Failure [6] combined with the backtracking control strategy. Their evaluation approach is believed not to be suitable for the general database environment where the data reside in the second-memory device.

In this paper, we describe a recursive query evaluation strategy when extended rules are present in the database. The deduction mechanism being adopted is the substitution rule. We consider four types of recursive extended rules (one transitive closure and three linear recursion cases) as well as reducible recursion. Reducible recursion is a case where one of existing techniques is directly applicable after appropriate transformations. As for four non-reducible recursion cases, none of existing techniques is applied. One interesting conclusion of our work is that the termination condition of the extended recursive query evaluation except the reducible recursion is independent on the contents of EDB. That is, the number of iterations in extended recursive query answering can be determined by the structure of the recursive rule set. We believe that those four classes are enough to see how existentially quantified variables behave in a recursive formula.

The rest of the paper is organized as follows: Section 2 presents basic materials on the extended rule and the substitution rule. Section 3 considers the reducible recursion as well as an query evaluation of extended rules of transitive closure type. In section 4, we discuss an recursive query evaluation of linear recursion type. We conclude the paper in section 5.

2 Preliminaries

We assume that the readers are familiar with the basic notations in [5] such as unifications, resolution, paramodulation and factoring. Following Lee and Henschen [11] we have the following definition.

Definition 2.1 An *extended* IDB rule is of the form (or equivalent to),

$$(\forall \vec{x}) [P(\vec{x}) \leftarrow (\vec{Q}\vec{y})\{A_1(\vec{x}, \vec{y}) \wedge \dots \wedge A_n(\vec{x}, \vec{y})\}],$$

where $\vec{Q}\vec{y}$ denotes a list of quantified variables.

By an extended rule we can easily define a virtual view which involves a division operation of relational algebra, therefore enhance the expressive power of deductive databases. The precious semantics of existentially quantified variables in a formula is viewed by the name of *expansion form*.

Definition 2.2 Let R be an extended rule with existential quantifiers. An *intermediate expansion form* (IEF) of R is a clause in which all Skolem functions of R are elim-

inated due to repeated application of paramodulation [5] from the appropriate DCA (Domain Closure Axiom). A clause which is a subsuming factor of an IEF and does not have a subsuming factor for itself is called an *expansion form* (EF) of R .

Example 2.1 Expansion Form

$$R : (\forall x)(\exists z)(\forall y)(\forall v) [P(x, y) \leftarrow A(z, v) \wedge B(y) \wedge C(x, v)]$$

Let D_z be $\{c_1, \dots, c_n\}$. Then the expansion form is

$$\begin{aligned} & [P(x, y^1) \neg A(c_1, v^1) \neg B(y^1) \neg C(x, v^1) \\ & \quad \neg A(c_2, v^2) \quad \neg C(x, v^2) \\ & \quad \dots \\ & \quad \neg A(c_n, v^n) \quad \neg C(x, v^n)]. \end{aligned}$$

Note that the existentially quantified variable z is replaced by each element of the domain of z , and all occurrences of variable v are distinct due to renaming. $\square$

Properties of the expansion form and how to construct the expansion form are found in [11]. As for the deduction step of deductive databases, we rely on a new inference rule, the substitution rule, instead of ordinary resolution. Here is the formal definition of the substitution rule [12].

Definition 2.3 Suppose we have two clauses

1. $(\forall \vec{x}) [P(\vec{x}, \vec{a}) \leftarrow (\vec{Q}\vec{y}) \vec{A}(\vec{x}, \vec{y}, \vec{b})]$
2. $(\forall \vec{u}) [R(\vec{u}, \vec{c}) \leftarrow (\vec{Q}\vec{v}) \{B_1(\vec{u}, \vec{v}, \vec{d}_1) \wedge \dots \wedge B_n(\vec{u}, \vec{v}, \vec{d}_n)\}]$

such that $P(\vec{x}, \vec{a})$ and $B_i(\vec{u}, \vec{v}, \vec{d}_i)$ are unifiable in Skolemized form. Assume no variables in common. Let λ be a MGU such that $\{P(\vec{x}, \vec{a})\}\lambda = \{B_i(\vec{u}, \vec{v}, \vec{d}_i)\}\lambda$ in the un-Skolemized clauses. The substitution rule generates clause 3 from clause 1 and clause 2

3. $[(\forall \vec{u}) [R(\vec{u}, \vec{c}) \leftarrow (\vec{Q}\vec{v}) \{B_1(\vec{u}, \vec{v}, \vec{d}_1) \wedge \dots \wedge B_{i-1}(\vec{u}, \vec{v}, \vec{d}_{i-1}) \wedge (\vec{Q}\vec{y}) \vec{A}(\vec{x}, \vec{y}, \vec{b}) \wedge B_{i+1}(\vec{u}, \vec{v}, \vec{d}_{i+1}) \wedge \dots \wedge B_n(\vec{u}, \vec{v}, \vec{d}_n)\}]]\lambda.$

In this case we mean λ to replace all occurrences of the variables both free and bound. When a variable in clause 3 is quantified more than once in the quantifier matrix, all quantifiers except the outermost one are removed. When a constant is quantified in the quantifier matrix, remove that quantifier. Clause 3 is called a conclusion. $\square$

The fundamental difference with the resolution principle is that the substitution rule does not Skolemize given clauses, i.e. universal quantifiers in the body of rules are allowed to occur when the substitution rule is applied. Note that the MGU λ is constructed from the corresponding unSkolemized literals, while the MGU in resolution is obtained in Skolemized literals. It is shown that the substitution rule preserves the derivability of R where R is a head literal in conclusion [12]. We use the substitution rule in the deduction step and evaluate the derived formula by use of relational algebra. An existentially quantified variable is then implemented by a division operator. We do not rely on meta rules (such as Negation as Failure or Closed World Assumption) in order to evaluate the existentially quantified variable in the formula. For a detailed description on the substitution rule, see Lee and Henschen [12].

Example 2.2 Substitution Rule

Consider the two clauses.

1. $(\forall x_1)(\forall x_2) [P(x_1, x_2) \leftarrow (\forall y_1) A(x_1, x_2, y_1)]$

2. $(\forall u_1)(\forall u_2)\{R(u_1, u_2) \leftarrow (\forall v_1)\{P(u_1, v_1) \wedge B(v_1, u_2)\}\}$
 When clause 1 and clause 2 are Skolemized, we have

1'. $P(x_1, x_2) \neg A(x_1, x_2, f(x_1, x_2))$ and

2'. $R(u_1, u_2) \neg P(u_1, g(u_1, u_2)) \neg B(g(u_1, u_2), u_2)$.

The clauses are resolvable with MGU $= \{u_1/x_1, g(u_1, u_2)/x_2\}$. So, we can apply the substitution rule. λ for the unSkolemized literals is $\{u_1/x_1, v_1/x_2\}$. The conclusion is

$$(\forall u_1)(\forall u_2) [R(u_1, u_2) \leftarrow (\forall v_1)(\forall y_1)\{A(u_1, v_1, y_1) \wedge B(v_1, u_2)\}].$$

R is calculated by evaluating $\Delta_{v_1}((\Delta_{y_1} A(u_1, v_1, y_1)) \bowtie B(v_1, u_2))$ where $\Delta_{x_i} A(x_1, \dots, x_n)$ is an abbreviation for $A(x_1, \dots, x_n) \div \text{Domain}(x_i)$. $\square$

3 Reducible Recursion and Transitive Closure

There are certain cases in which a set of extended recursive rules can be transformed into a set of ordinary recursive rules so that the reduced recursive rule set can be handled by use of one of existing techniques found in [3,4]. Such recursion is said to be reducible.

Example 3.1 Reducible Recursion

(A) 1. $(\forall x)(\forall y) [P(x, y) \leftarrow \{(\forall w) A(x, w, z)\} \wedge P(z, y)]$

2. $(\forall x)(\forall y) [P(x, y) \leftarrow B(x, y)]$

The above case is easily solved by transforming the given rules into the following.

1'. $(\forall x)(\forall y) [P(x, y) \leftarrow A'(x, z) \wedge P(z, y)]$

2. $(\forall x)(\forall y) [P(x, y) \leftarrow B(x, y)]$

where $(\forall x)(\forall z) [A'(x, z) \leftarrow (\forall w) A(x, w, z)]$.

(B) 1. $(\forall x)(\forall y)(\forall z)(\forall w) [P(x, y) \leftarrow (\forall u)(\forall v)\{A(x, u, z) \wedge P(z, w) \wedge B(w, v, y)\}]$

2. $(\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)]$

This case is transformed into the following.

1'. $(\forall x)(\forall y)(\forall z)(\forall w) [P(x, y) \leftarrow A'(x, z) \wedge P(z, w) \wedge B'(w, y)]$

2. $(\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)]$

where $(\forall x)(\forall z) [A'(x, z) \leftarrow (\forall u) A(x, u, z)]$ and $(\forall w)(\forall y) [B'(w, y) \leftarrow (\forall v) B(w, v, y)]$.

$\square$

The transformed recursive rule set is clearly handled by an existing technique, because the existentially quantified variables no longer exist and the primed relations can be derived separately. Let R be a set of extended recursive formulas. Let $P(x_1, \dots, x_n)$ be a recursive predicate occurring in the body of some of the rules. If one of x_i is existentially quantified in Prenex Normal Form, then R is not reducible.

As for non-reducible recursion, we present four cases of transitive closure and linear recursion type. For convenience, we introduce a new notation A^* . $A^*(x_1, \dots, x_n)$ denotes an instance of a relation $A(x_1, \dots, x_n)$ such that $A^*(x_1, \dots, x_n)$ has all possible combinations (i.e. tuples) over its domains. Let $m_1, \dots, m_n$ be a size of the domain of each attribute $x_1, \dots, x_n$. The size of relation $A^*(x_1, \dots, x_n)$ is $m_1 \times \dots \times m_n$.

Let us consider the typical transitive closure which we call TC1.

1. $(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z) \{A(x, z) \wedge P(z, y)\}]$

2. $(\forall x)(\forall y) [P(x, y) \leftarrow B(x, y)]$

Let us apply the substitution rule in order to expand the recursive formula

3. $(2 \rightarrow 1)$

$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z) \{A(x, z) \wedge B(z, y)\}]$

4. $(1 \rightarrow 1)$ All variables in the first clause 1 are subscripted by 1.

$\lambda = \{z/x_1, y/y_1\}$

$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z) [A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge P(z_1, y)\}]]$

5. $(2 \rightarrow 4)$

$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z) [A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge B(z_1, y)\}]]$

6. $(1 \rightarrow 4)$ All variables in clause 1 are subscripted by 2. $\lambda = \{z_1/x_2, y_1/y_2\}$

$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z) [A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge (\forall z_2)\{A(z_1, z_2) \wedge P(z_2, y)\}\}]]$

7. $(2 \rightarrow 6)$

$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z) [A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge (\forall z_2)\{A(z_1, z_2) \wedge B(z_2, y)\}\}]]$

Clauses 2, 3, 5 and 7 are expanded formulas which will generate some of the answers. There is fortunately a pattern³ in the expanded formulas just like the ordinary case [8]. Therefore, it is possible to predict the general expanded formula, i.e.

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)\{A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge (\forall z_2)\{A(z_1, z_2) \wedge \dots \wedge (\forall z_n)\{A(z_{n-1}, z_n) \wedge B(z_n, y)\}\}\}\}].$$

Now, let us apply the expanded formulas to particular instances of database in order to see how an existentially quantified variable behaves in a recursive formula.

Case 1: Let D_x, D_y and D_z be $\{1, 2, 3\}$.

Relation $A = A^*$, and Relation B :

x	y
2	2

From clause 2, the instance of P is $\{< 2, 2 >\}$.

From clause 3,

$A \bowtie B = \{< 1, 2, 2 >, < 2, 2, 2 >, < 3, 2, 2 >\}$.

Then, $\Delta_{z_2}(A \bowtie B) = \emptyset$, i.e. no new tuple is derived.

Case 2: Let D_x, D_y and D_z be $\{1, 2, 3\}$.

Relation $A = A^*$, and Relation B :

x	y
1	1
2	2
3	3

From clause 2, the instance of P is

$\{< 1, 1 >, < 2, 2 >, < 3, 3 >\}$.

From clause 3, $A \bowtie B =$

$\{< 1, 1, 1 >, < 2, 1, 1 >, < 3, 1, 1 > \\ < 1, 2, 2 >, < 2, 2, 2 >, < 3, 2, 2 > \\ < 1, 3, 3 >, < 2, 3, 3 >, < 3, 3, 3 >\}$.

³When the substitution rule is performed, variables need to be re-named appropriately in order to see whether or not there is a pattern in the expanded formulas. Otherwise it would be hard to find a pattern in the expanded formulas.

Then, $\Delta_{\mathcal{S}_2}(A \bowtie B) = \emptyset$, i.e. no new tuple is derived.
Case 3: Let D_x , D_y and D_z be $\{1,2,3\}$.

Relation $A = A^*$ and Relation B :

x	y
1	1
2	1
3	1

From clause 2, the instance of P is
 $\{ \langle 1, 1 \rangle, \langle 2, 1 \rangle, \langle 3, 1 \rangle \}$.

From clause 3, $A \bowtie B =$

$$\{ \langle 1, 1, 1 \rangle, \langle 2, 1, 1 \rangle, \langle 3, 1, 1 \rangle, \\ \langle 1, 2, 1 \rangle, \langle 2, 2, 1 \rangle, \langle 3, 2, 1 \rangle, \\ \langle 1, 3, 1 \rangle, \langle 2, 3, 1 \rangle, \langle 3, 3, 1 \rangle \}.$$

Then, $\Delta_{\mathcal{S}_2}(A \bowtie B) = \{ \langle 1, 1 \rangle, \langle 2, 1 \rangle, \langle 3, 1 \rangle \}$.

Again, no new tuple is derived because these are already added by rule 2.

Interesting enough, the existentially quantified variable tends to limit the search space as the above cases show. Formally stated, one important property associated with TC1 is that the iteration process is not necessary and that only one execution of clause 2 (the exit rule) is enough to derive all answers regardless of the contents of database.

Theorem 3.1 Let R be a set of recursive extended rules such that

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z) \{A(x, z) \wedge P(z, y)\}] \quad (1)$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow B(x, y)]. \quad (2)$$

Then only one execution of the exit rule (i.e. clause 2) generates all instances of predicate P .

Proof: Let us recall the general expanded formula,

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z_1)\{A(x, z_1) \wedge (\forall z_2)\{A(z_1, z_2) \\ \wedge (\forall z_3)\{A(z_2, z_3) \wedge \dots \wedge (\forall z_n)\{A(z_{n-1}, z_n) \\ \wedge B(z_n, y)\} \dots \}\}].$$

Consider $n = 1$ case, i.e.

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z_1)\{A(x, z_1) \wedge B(z_1, y)\}]. \quad (3)$$

Because variable z_1 is universally quantified in the rule body, in order to derive a tuple $\langle a_i, a_j \rangle$ from clause (3), the instances of relation A and relation B must be,

Relation A:	<table border="1"> <thead> <tr> <th>x</th> <th>z</th> </tr> </thead> <tbody> <tr><td>...</td><td>...</td></tr> <tr><td>a_i</td><td>a_1</td></tr> <tr><td>a_i</td><td>a_2</td></tr> <tr><td>...</td><td>...</td></tr> <tr><td>a_i</td><td>a_n</td></tr> <tr><td>...</td><td>...</td></tr> </tbody> </table>	x	z	...	...	a_i	a_1	a_i	a_2	...	...	a_i	a_n	...	...
x	z														
...	...														
a_i	a_1														
a_i	a_2														
...	...														
a_i	a_n														
...	...														
Relation B:	<table border="1"> <thead> <tr> <th>x</th> <th>z</th> </tr> </thead> <tbody> <tr><td>...</td><td>...</td></tr> <tr><td>a_1</td><td>a_j</td></tr> <tr><td>a_2</td><td>a_j</td></tr> <tr><td>...</td><td>...</td></tr> <tr><td>a_n</td><td>a_j</td></tr> <tr><td>...</td><td>...</td></tr> </tbody> </table>	x	z	...	...	a_1	a_j	a_2	a_j	...	...	a_n	a_j	...	...
x	z														
...	...														
a_1	a_j														
a_2	a_j														
...	...														
a_n	a_j														
...	...														

Note that the tuple $\langle a_i, a_j \rangle$ is already in relation B .

For $n = 2$ case,

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z_1)\{A(x, z_1) \wedge (\forall z_2) \\ \{A(z_1, z_2) \wedge B(z_2, y)\}\}] \quad (4)$$

Note that all variables in the second occurrence of A in formula (4), $A(z_1, z_2)$, are existentially quantified in Prenex Normal Form. Consider an expansion form of formula (4). Variables z_1 and z_2 are replaced by each element of the domain, so $A(z_1, z_2)$ will be expanded to A^* in the expansion form. Hence, in order to derive a tuple $\langle a_i, a_j \rangle$ from formula (4), the instance of relation A must be A^* , and

x	z
...	...
a_1	a_j
a_2	a_j
...	...
a_n	a_j
...	...

the instance of relation B must be

Tuple $\langle a_i, a_j \rangle$ is already in B . For $n \geq 3$ cases, similar arguments are applied. **Q.E.D.**

4 Linear Recursion

In this section, we consider three kinds of extended linear recursions. The basic approach to linear recursion is similar to the previous one illustrated in section 3. For the sake of simplicity and readability, we present only the general expanded formula and the associated theorem of each case.

One typical non-reducible class of linear recursion looks like this.

1. $(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w)(\forall z)\{A(x, z) \wedge P(z, w) \wedge B(w, y)\}]$
2. $(\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)]$

We call it LR1. Let us apply the substitution rule to get expanded formulas.

3. $(2 \rightarrow 1) (\forall x)(\forall y) [P(x, y) \leftarrow (\exists w)(\forall z) \\ \{A(x, z) \wedge C(z, w) \wedge B(w, y)\}]$
4. $(1 \rightarrow 1)$ All variables in the first clause 1 are subscripted by 1. $\lambda = \{z/x_1, w/y_1\}$.
 $(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w)(\forall z)\{A(x, z) \wedge (\exists w_1)(\forall z_1) \\ \{A(z, z_1) \wedge P(z_1, w_1) \wedge B(w_1, w)\} \wedge B(w, y)\}]$
5. $(2 \rightarrow 4)$
 $(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w)(\forall z)\{A(x, z) \wedge (\exists w_1)(\forall z_1) \\ \{A(z, z_1) \wedge C(z_1, w_1) \wedge B(w_1, w)\} \wedge B(w, y)\}]$
6. $(1 \rightarrow 4)$ All variables in clause 1 are subscripted by 2. $\lambda = \{z_1/x_2, w_1/y_1\}$
 $(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w)(\forall z)\{A(x, z) \wedge (\exists w_1) \\ (\forall z_1)\{A(z, z_1) \wedge (\exists w_2)(\forall z_2)\{A(z_1, z_2) \wedge P(z_2, w_2) \\ \wedge B(w_2, w_1)\} \wedge B(w_1, w)\} \wedge B(w, y)\}]$
7. $(2 \rightarrow 6)$
 $(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w)(\forall z)\{A(x, z) \wedge (\exists w_1) \\ (\forall z_1)\{A(z, z_1) \wedge (\exists w_2)(\forall z_2)\{A(z_1, z_2) \wedge C(z_2, w_2) \\ \wedge B(w_2, w_1)\} \wedge B(w_1, w)\} \wedge B(w, y)\}]$

There is a pattern in the expanded formulas. The general expanded formula is

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w_1)(\forall z_1)\{A(x, z_1) \wedge (\exists w_2)(\forall z_2) \\ \{A(z_1, z_2) \wedge \dots \wedge (\exists w_n)(\forall z_n)\{A(z_{n-1}, z_n) \wedge C(z_n, w_n) \\ \wedge B(w_n, w_{n-1})\} \wedge \dots \wedge B(w_2, w_1)\} \wedge B(w_1, y)\}].$$

Theorem 4.1 Let P be a recursive predicate defined by

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w)(\forall z)\{A(x, z) \wedge P(z, w) \\ \wedge B(w, y)\}]$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)].$$

Then if the instance of A in EDB is not A^* , all tuples of P are generated by

$$(\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)] \text{ and}$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w_1)(\forall z_1)\{A(x, z_1) \wedge C(z_1, w_1)\}]$$

$$\wedge B(w_1, y)\},$$

independent of the contents of B and C .

Proof: Recall that the general expanded formula is
 $(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w_1)(\forall z_1)\{A(x, z_1) \wedge (\exists w_2)(\forall z_2)\{A(z_1, z_2) \wedge \dots \wedge (\exists w_n)(\forall z_n)\{A(z_{n-1}, z_n) \wedge C(z_n, w_n) \wedge B(w_n, w_{n-1})\} \wedge \dots \wedge B(w_2, w_1)\} \wedge B(w_1, y)\}].$

Consider the $n = 2$ case, i.e.

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\exists w_1)(\forall z_1)\{A(x, z_1) \wedge (\exists w_2)(\forall z_2)\{A(z_1, z_2) \wedge C(z_2, w_2) \wedge B(w_2, w_1)\} \wedge B(w_1, y)\}].$$

In order to derive $\langle a_i, a_j \rangle$ from the above formula, the instance of A must be A^* because all variables in the second occurrence of A (i.e. $A(z_1, z_2)$) are existentially quantified in Prenex Normal Form and existentially quantified variables are to be replaced by each element of the corresponding domain in the corresponding expansion form. Because we assume that the instance of A is a proper subset of A^* , no tuple is derived from the above formula. For $n \geq 3$, a similar argument is applied. **Q.E.D.**

Another typical non-reducible class of linear recursive formulas looks like this.

$$1. (\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)(\exists w)\{A(x, z) \wedge P(z, w) \wedge B(w, y)\}]$$

$$2. (\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)]$$

We call it LR2. Let us apply the substitution rule to get expanded formulas as we did for the case of LR1.

$$3. (2 \rightarrow 1) (\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)(\exists w)\{A(x, z) \wedge C(z, w) \wedge B(w, y)\}]$$

4. $(1 \rightarrow 1)$ All variables in the first clause 1 are subscripted by 1. $\lambda = \{z/x_1, w/y_1\}$.

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)(\exists w)\{A(x, z) \wedge (\forall z_1)(\exists w_1)\{A(z, z_1) \wedge P(z_1, w_1) \wedge B(w_1, w)\} \wedge B(w, y)\}]$$

$$5. (2 \rightarrow 4)$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)(\exists w)\{A(x, z) \wedge (\forall z_1)(\exists w_1)\{A(z, z_1) \wedge C(z_1, w_1) \wedge B(w_1, w)\} \wedge B(w, y)\}]$$

$$6. (1 \rightarrow 4) \text{ All variables in clause 1 are subscripted by 2. } \lambda = \{z_1/x_2, w_1/y_1\}$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)(\exists w)\{A(x, z) \wedge (\forall z_1)(\exists w_1)\{A(z, z_1) \wedge (\forall z_2)(\exists w_2)\{A(z_1, z_2) \wedge P(z_2, w_2) \wedge B(w_2, w_1)\} \wedge B(w_1, w)\} \wedge B(w, y)\}]$$

$$7. (2 \rightarrow 6)$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)(\exists w)\{A(x, z) \wedge (\forall z_1)(\exists w_1)\{A(z, z_1) \wedge (\forall z_2)(\exists w_2)\{A(z_1, z_2) \wedge C(z_2, w_2) \wedge B(w_2, w_1)\} \wedge B(w_1, w)\} \wedge B(w, y)\}]$$

There is a pattern in the expanded formulas. The general expanded formula is

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z_1)(\exists w_1)\{A(x, z_1) \wedge (\forall z_2)(\exists w_2)\{A(z_1, z_2) \wedge \dots \wedge (\forall z_n)(\exists w_n)\{A(z_{n-1}, z_n) \wedge C(z_n, w_n) \wedge B(w_n, w_{n-1})\} \wedge \dots \wedge B(w_2, w_1)\} \wedge B(w_1, y)\}].$$

Theorem 4.2 Let P be a recursive predicate defined by

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)(\exists w)\{A(x, z) \wedge P(z, w) \wedge B(w, y)\}]$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)].$$

Then if the instance of A in EDB is not A^* , all tuples of P are generated by

$$(\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)] \text{ and}$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z_1)(\exists w_1)$$

$$\{A(x, z_1) \wedge C(z_1, w_1) \wedge B(w_1, y)\}],$$

independent of the contents of B and C .

Proof: Similar to the proof of Theorem 4.1.

Now, consider another type of linear recursive formulas which we call LR3.

$$1. (\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)\{A(x, z) \wedge (\forall w)\{P(z, w) \wedge B(w, y)\}\}]$$

$$2. (\forall x)(\forall y) [P(x, y) \leftarrow C(x, y)]$$

Apply the substitution rule to get the general expanded formula.

$$3. (2 \rightarrow 1)$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)\{A(x, z) \wedge (\forall w)\{C(z, w) \wedge B(w, y)\}\}]$$

$$4. (1 \rightarrow 1) \lambda = \{z/x_1, w/y_1\}$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)\{A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge (\forall w_1)\{P(z_1, w_1) \wedge (\forall w)\{B(w_1, w) \wedge B(w, y)\}\}\}\}]$$

$$5. (2 \rightarrow 1)$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)\{A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge (\forall w_1)\{C(z_1, w_1) \wedge (\forall w)\{B(w_1, w) \wedge B(w, y)\}\}\}\}]$$

$$6. (1 \rightarrow 4) \lambda = \{z_1/x_2, w_1/y_2\}$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)\{A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge (\forall z_2)\{A(z_1, z_2) \wedge (\forall w_2)\{P(z_2, w_2) \wedge (\forall w_1)\{B(w_2, w_1) \wedge (\forall w)\{B(w_1, w) \wedge B(w, y)\}\}\}\}\}\}]$$

$$7. (2 \rightarrow 6)$$

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)\{A(x, z) \wedge (\forall z_1)\{A(z, z_1) \wedge (\forall z_2)\{A(z_1, z_2) \wedge (\forall w_2)\{C(z_2, w_2) \wedge (\forall w_1)\{B(w_2, w_1) \wedge (\forall w)\{B(w_1, w) \wedge B(w, y)\}\}\}\}\}\}]$$

Similarly there is a pattern. The general expanded formula is

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z_1)\{A(x, z_1) \wedge (\forall z_2)\{A(z_1, z_2) \wedge \dots \wedge (\forall z_n)\{A(z_{n-1}, z_n) \wedge (\forall w_n)\{C(z_n, w_n) \wedge (\forall w_{n-1})\{B(w_n, w_{n-1}) \wedge \dots \wedge (\forall w_1)\{B(w_2, w_1) \wedge B(w_1, y)\}\}\}\}\}\}].$$

Theorem 4.3 Let P be a recursive predicate of class LR3. Then, one execution of the exit rule generates all tuples of P .

Proof: Consider the $n = 1$ case, i.e.

$$(\forall x)(\forall y) [P(x, y) \leftarrow (\forall z)\{A(x, z) \wedge (\forall w)\{C(z, w) \wedge B(w, y)\}\}]. \quad (5)$$

In order to derive a tuple $\langle a_i, a_j \rangle$ from formula (5), the instance of relation A , B and C are

Relation A:	...	...
	a_i	a_1
	...	...
	a_i	a_n
Relation B:	...	...
	a_1	a_j
	...	...
	a_n	a_j
	...	...

Because relation C has all possible combinations of each element of the domain, the tuple $\langle a_i, a_j \rangle$ is already in relation C . For $n > 1$, a similar argument is clearly applied. **Q.E.D.**

Corollary 4.1 Let R be a set of recursive extended rules in the class TC1, LR1, LR2, or LR3. The recursive predicate P in R has a least fixed point.

Proof: immediate from Theorem 3.1, Theorem 4.1, Theorem 4.2 and Theorem 4.3. **Q.E.D.**

One of factors we have to take into account when we evaluate a recursive query is a termination condition. The termination condition is based on a *least fixed point* concept [1]. In the database context Aho and Ullman [1] show that any relational algebra expression that does not use a set difference operator is additive in all its variables, and that a monotonic expression can have a least fixed point. The existential quantified variables correspond to the division operator of relational calculus and the division operator is not independent (i.e. not a prime operator) [16]. When a division operator is expressed by independent operators, a set difference operator should be used. This implies that there may be no least fixed point of a recursive predicate in an extended rule. However, as Corollary 4.1 says, the four classes of recursion we have considered have a least fixed point. This phenomenon is consistent with Aho and Ullman's remark that not every expression involving a difference operator fails to be monotone.

5 Closing Remarks

We have discussed the extended recursive query evaluation based on the substitution rule. First, we present reducible recursion where a given recursive rule set is transformed into ordinary recursion so that an existing method is applied. Secondly, we have provided expanded forms for some of the major non-reducible cases. We have shown that non-reducible recursions have almost always fixed iteration bounds independent of the contents of the EDB.

Theorems 3.1, 4.1, 4.2 and 4.3 cover virtually all of the transitive closure and linear recursion cases where the body variables are of linear variable pattern. We have considered extensions to Han's classification [7] to allow universal quantifiers in the body. We believe the addressed techniques can be easily extended to these cases. We also believe that the occurrence of existentially quantified variables in the recursive body predicate will tend to limit the level of recursive search dramatically just as it did for our cases.

References

- [1] Aho, A.V. and Ullman, J.D., Universality of Data Retrieval Languages, *Proceedings of the 6th ACM Symposium on Principles of Programming Languages*, Jan. 1979, pp. 100-120.
- [2] Bancilhon, F., Maier, D., Sagiv, Y., and Ullman, J., Magic Sets and Other Strange Ways to Implement Logic Programs, *Proceedings of the 5th ACM SIGMOD-SIGART News Symposium on Principles of Database Systems*, 1986, pp. 1-15.
- [3] Bancilhon, F. and Ramakrishnan, P., An Amateur's Introduction to Recursive Query Processing Strategies, *Proceedings of ACM SIGMOD '86*, May 1986, pp. 16-52.
- [4] Bancilhon, F. and Ramakrishnan, P., Performance Evaluation of Data Intensive Logic Programs, in: J. Minker (ed.), *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*, August 1986, pp 284-314.
- [5] Chang, C.L. and Lee, R.C.T., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.
- [6] Clark, K.L., Negation as Failure, in: H. Gallaire and J. Minker (ed.), *Logic and Databases*, Plenum, 1978, pp 293-322.
- [7] Han, J., Pattern-Based and knowledge-Directed Query Compilation for Recursive Data Bases, Ph/D Dissertation, University of Wisconsin at Madison, Wisconsin, 1985.
- [8] Henschen, L. and Naqvi, S., On Compiling Queries in Recursive First-Order Databases, *J. Assoc. Comput. Mach.*, 31(1):45-85 (Jan.1984).
- [9] Hermes, H., *Introduction to Mathematical Logic*, Springer-Verlag, 1973.
- [10] Lee, S.H., Extending the Expressive Power of Deductive Databases, Ph/D Dissertation, Northwestern University, 1989.
- [11] Lee, S.H. and Henschen, L., Semantics and Indefiniteness of Existential Quantifiers in the Deductive Database, *Proceedings of the Annual Conference of the International Association of Knowledge Engineers*, 1989, pp 132-151.
- [12] Lee, S.H. and Henschen, L., Substitution-based Compilation of Extended Rules in Deductive Databases, *Proceedings of the 10th International Conference on Automated Deduction (CADE)*, M.E. Stickel, (Eds.) Springer-Verlag Lecture Notes in Artificial Intelligence, No. 449, 1990, pp 57-71.

- [13] Lloyd, J.W. and Topor, R.W., Making Prolog More Expressive, *Journal of Logic Programming* 1(3):225-240 (Oct. 1984).
- [14] Robinson, J., A Machine-Oriented Logic Based on the Resolution Principle, *J. Assoc. Comput. Mach.* 12(1):23-41 (January 1965).
- [15] Ullman, J.D., Implementation of Logical Query Languages for Databases, *ACM TODS* 10(3):289-321 (Sept. 1985).
- [16] Ullman, J.D., *Principles of Database and Knowledge-Base Systems*, Vol. 1 and 2, Computer Science Press, 1982.
- [17] Vieille, L., Recursive Axioms in Deductive Databases: The Query/Subquery Approach, *Proceedings of the First International Conference on Expert Database Systems*, 1986, pp 179-194.