A NECESSARY CONDITION FOR A DOUBLY RECURSIVE RULE TO BE EQUIVALENT TO A LINEAR RECURSIVE RULE

by

Weiming Zhang and C T Yu
Dept of Electrical Engineering and Computer Science
University of Illinois at Chicago
Chicago, Illinois 60680

Abstract

Nonlinear recursive queries are usually less efficient in processing than linear recursive queries It is therefore of interest to transform non-linear recursive queries into linear ones We obtain a necessary and sufficient condition for a doubly recursive rule of a certain type to be logically equivalent to a single linear recursive rule obtained in a specific way

1. Introduction

Many proposals [BMSU, BaRa, GaMa, HaLu, HeNa, IoWo, Ioan, LaYa, Naug, Sacc, Ullm, Vard, YuZh, Zani] have been made to optimize recursive queries in deductive databases Among these proposals, some [BSMU, Chan, HaLu, HeNa, TrYu, YuZh, etc] are intended to process linear recursive queries and others [BaRa, CeGL, Ullm] are intended to process more general nonlinear recursive queries One of the reasons for optimizing linear recursive queries is that, in the "real world", such queries are encountered most frequently [BaRa] On the other hand, nonlinear recursive queries also arise [Vard] and the algorithms proposed to solve nonlinear recursive queries are usually less efficient than those proposed for linear recursive queries We observe that, many nonlinear recursive queries are in fact logically equivalent to some linear recursive queries A famous example would be the ancestor query which has appeared in many papers [BaRa, Chan, etc] By identifying such nonlinear recursive queries and converting them into equivalent linear recursive queries, we are able to utilize the more efficient linear-recursive-query-processing algorithms

In a previous paper [TrYu], we have given a method to convert a nonlinear recursive rule into a sequence of linear recursive rules (A similar but not identical convertion appears in [CeGL]) and a sufficient condition on the placement of variables in a rule to convert a doubly recursive rule into a single linear recursive rule

This paper is an extension of [TrYu] We give a necessary condition on the placement of variables in the

rule to convert a doubly recursive rule of a certain type into a single linear recursive rule The necessary condition stated in this paper is a slight generalization of the sufficient condition given in [TrYu] By modifying the proof given in [TrYu], it can be shown that the necessary condition is also a sufficient condition Thus, for the doubly recursive rule of the type, the condition turns out to be both necessary and sufficient

An analogy of the result obtained in this paper is as follows In relational database, a distinction is made between tree queries and cyclic queries [BeCh, YuOz] where tree queries can be optimized "easily" but cyclic queries can not The result is useful not only in query optimization, but also in database design and graph theory [BFMY, GoSh]

In this paper, we attempt to obtain a similar result for deductive databases i e we attempt to differentiate the "simple" linear recursive queries from the more complicated nonlinear recursive queries The first step is to find a necessary and sufficient condition for a doubly recursive rule to be equivalent to a linear recursive rule

It has recently been shown [BKBR, Shma] that the problem of whether a non linear recursive rule is equivalent to some linear recursive rule is undecidable In Section 2, we give the definitions and notations used in this paper and briefly review the previous results in [TrYu] In Section 3, the necessary condition for a doubly recursive rule of certain type to be logically equivalent to a single linear recursive rule obtained in a specific way is stated along with a number of comprehensive examples The necessary (and sufficient) condition can be easily checked A proof of the result is given in Section 4

2 Definitions, Notations and Previous Results

A deductive database consists of two types of relations or predicates (predicates and relations will be used interchangeably throughout the paper), the base relations and the deductive relations The set of all base relations form the extensional database, EDB, in which, the base relations are stored physically The set of deductive relations form the intensional database, IDB, in which, the deductive relations are deduced from the base relations and deductive relations by using the sets of rules defining them

A rule defining a deductive predicate is a function-free, Horn clause of the form $B - A_1A_2 \quad A_k$ with head B and body $A_1A_2 \quad A_k$

A (directly) recursive rule is a rule with the head predicate B appearing in the body at least once If B appears in the body exactly once, then, the rule is called a linear recursive rule Otherwise , the rule is nonlinear recursive A doubly recursive rule is a nonlinear recursive rule with two occurrences of the head predicate appearing in the body of the rule

Let s be a recursive predicate of arity n The recursive rule defining s will involve n distinct universally quantified variables, denoted by $X = (x_1\ x_2 \quad x_n)$ In the body of this rule, there will be $p \geq 0$ distinct existentially quantified variables, denoted by $U = (u_1\ u_2 \quad u_p)$ The universally quantified vari-

ables, X, which appear in the head of the rule (and possibly in the body of the rule) are called the *distinguished* variables, the existentially quantified variables, U, which occur in the body of the rule are the *non-distinguished* variables For each predicate, of arity g, in the body of the rule, g arguments are selected from the variables of X and U Such selection is denoted by a formal matrix product (X,U)H = (x_1 x_2 x_n u_1 u_2 u_p)H, where H is a (n+p)×g 0-1 matrix with exactly one 1 in each column, and is called a selector If x_j (u_i) is at the k^{th} argument position in the matrix product, we say that H selects x_j (u_i) and places it at position k

Let $s(x_1\ x_2\ \ x_n)$ be a recursive predicate defined by a set of rules For a constant vector A = (a_1 a_2 a_n), (a_1 a_2 a_n) is a tuple of s in IDB if $s(a_1\ a_2\ \ a_n)$ can be established by a finite rule/goal tree similar to that defined in [Ullm] by trimming unnecessary "or" branches, i e if there exists a substitution θ which assigns constants to all distinguished and non-distinguished variables, such that, by applying θ to all nodes in the rule/goal tree, each leaf node unifies with a tuple in EDB and the root of the tree unifies with $s(a_1\ a_2\ \ a_n)$

Example 2 1 Let s be defined by

r_e $s(x_1\ x_2\ x_3)$ – $f(x_1\ x_2\ x_3)$

r_r $s(x_1\ x_2\ x_3)$ – $s(x_1\ x_2\ u_1)$ $r(u_1\ x_2\ u_2)$ $s(u_2\ x_2\ x_3)$

where r_e is an exit rule and r_r is a recursive rule

If the EDB contains $f(a_1\ a_2\ b_1)$, $f(b_2\ a_2\ c_1)$, $f(c_2\ a_2\ a_3)$, $r(b_1\ a_2\ b_2)$ and $r(c_1\ a_2\ c_2)$, with a's, b's and c's being constants, then $s(a_1\ a_2\ a_3)$ can be established by the rule/goal tree shown in Fig 2 1 since by applying

$$\theta = \{a_1/x_1, a_2/x_2, a_3/x_3, b_1/u_1, b_2/u_2, c_1/v_1, c_2/v_2\}$$

to the rule/goal tree, each leaf node unifies with a tuple in the EDB and we obtain $s(a_1\ a_2\ a_3)$ at the root

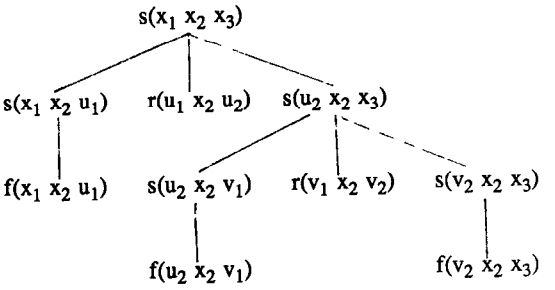

Fig 2 1

Note that, the tree in Fig 2 1 is constructed by applying a rule to each s-node Each time when a rule is applied to an s-node, the head of the rule unifies with the s-predicate and each predicate in the body of the rule becomes a child node of the s-node The non-distinguished variables may have to be renamed □

Let s(X) and t(X) be two predicates of the same arity and defined over the same domains We say that, s(X) implies t(X), denoted by s(X)==>t(X), if every tuple in s is a tuple in t s(X) and t(X) are logically equivalent, denoted by s(X)<==>t(X), if s(X)==>t(X) and t(X)==>s(X)

Our previous results show that, it is always possible to convert any nonlinear recursive rule into a sequence of linear recursive rules However, when the number of linear recursive rules in the sequence is large, the computation could be very inefficient. Therefore, it is of interest to determine the conditions which make a non-

linear recursive rule be logically equivalent to a single linear recursive rule

In [TrYu], we considered two deductive predicates defined by the following sets of rules

(I) r_e $s(X) - f(X)$
r_r $s(X) - s((X,U)Z^1)\ r((X,U)W)\ s((X,U)Z^2)$

(II) r_e $s_1(X) - f(X)$
r_r $s_1(X) - f((X,U)Z^1)\ r((X,U)W)\ s_1((X,U)Z^2)$

where r_e is the exit rule, r_r is the recursive rule, f and r are base predicates and Z^1, Z^2 and W are selectors The r_r of (I) is a doubly recursive rule and the r_r of (II) is a linear recursive rule (II) is obtained from (I) by replacing the first occurrence of the recursive predicate in the body of the recursive rule by the body of the exit rule It can be shown $s_1(X) ==> s(X)$

One of the results reported in [TrYu] is a sufficient condition for s and s_1 to be logically equivalent

If s and s_1 are defined by (I) and (II), respectively, and (i) each distinguished variable, x_j, is selected by at least one of the selectors Z^1 or Z^2, and the selected variable is placed at position j, and (ii) W selects only non-distinguished variables, then $s_1(X) <==> s(X)$ independent of the data

Example 2 2 Let s and s_1 be defined by

r_e $s(x\ y) - f(x\ y)$
r_r $s(x\ y) - s(x\ u)\ r(u\ v)\ s(v\ y)$

r_e $s_1(x\ Y) - f(x\ y)$
r_r $s_1(x\ Y) - f(x\ u)\ r(u\ v)\ s_1(v\ y)$

then s and s_1 satisfy the sufficient condition, thus $s_1(X) <==> s(X)$ for any EDB

In the next Section, we show a slightly more general condition to be necessary for a restricted type of rules

3 The Main Result

In this paper, the rules which we consider are defined by (I) and (II) with the following additional assumptions

Assumption 1.

The rules are range restricted, i e every distinguished variable has to appear in the body of each rule This assumption is commomly used [Reit, BaRa]

Assumption 2.

The recursive rule, r_r, is of the following type There exist two non-distinguished variables, u_i and u_j, such that, (i) u_i is selected and placed at some position i_1 by Z^1, placed at some position i_2 by W, but it is not selected by Z^2 (ii) u_j is selected and placed at some position j_1 by Z^2, placed at some position j_2 by W, but it is not selected by Z^1 The restriction on the rule may be illustrated by Fig 3 1

Z^1	W	Z^2
(u_i)	(u_j u_i)	(u_j)
i_1	j_2 i_2	j_1

Fig 3 1

Assumption 3

For every distinguished variable, x_j, selected by Z^1 or Z^2 or both, x_j is placed at exactly one position by each selector, i e a Z-selector is not

allowed to place a distinguished variable to two or more positions in the same predicate

Example 3 1 The following rule satisfies these assumptions

$$s(x_1\ x_2\ x_3) \leftarrow s(x_1\ u_1\ u_2)\ r(u_1\ u_3)\ s(x_2\ u_3\ x_3)$$

where $u_i = u_1$, $i_1 = 2$, $i_2 = 1$, $u_j = u_3$, $j_1 = 2$ and $j_2 = 2$

The main result we obtain is the following proposition

Proposition 3 1

Let s and s_1 be defined by (I) and (II), respectively, and satisfy Assumption 1, 2 and 3 A necessary condition for $s(X) <==> s_1(X)$ independent of data is that, in rule r_r,

(i) each distinguished variable, x_j, is selected by at least one of the selectors Z^1 or Z^2, and the selected variable is placed at position j, and

(ii) if x_j is selected by W, then both Z^1 and Z^2 select x_j and place it at position j

A proof of the result is in Section 4 Here, we give a comment on the connection of the Proposition 3 1 with the previous result in [TrYu]

The necessary condition in Proposition 3 1 is a slight generalization of the sufficient condition given in [TrYu], in the sense that, distinguished variables may appear in W However, in spite of the generalization, the proof of the sufficient condition supplied in [TrYu] can be generalized to show that the necessary condition stated here is actually sufficient Thus, the conditions (i) and (ii) stated in Proposition 3 1 are necessary and sufficient for the rules satisfying Assumptions 1, 2 and 3 A number of examples are given below to illustrate the result

Example 3 2 Let s and s_1 be defined by

r_e $s(x_1\ x_2\ x_3) \leftarrow f(x_1\ x_2\ x_3)$
r_r $s(x_1\ x_2\ x_3) \leftarrow s(x_1\ x_2\ u_1)\ r(u_2\ x_2\ u_1)\ s(u_2\ x_2\ x_3)$

r_e $s_1(x_1\ x_2\ x_3) \leftarrow f(x_1\ x_2\ x_3)$
r_r $s_1(x_1\ x_2\ x_3) \leftarrow f(x_1\ x_2\ u_1)\ r(u_2\ x_2\ u_1)\ s_1(u_2\ x_2\ x_3)$

It is easy to see that both the conditions (i) and (ii) stated in Proposition 3 1 are satisfied Therefore, by using the same techniques used in [TrYu], we can show that $s(X) <==> s_1(X)$ independent of data

Example 3 3 Let s and s_1 be defined by

r_e $s(x_1\ x_2\ x_3) \leftarrow f(x_1\ x_2\ x_3)$
r_r $s(x_1\ x_2\ x_3) \leftarrow s(x_3\ x_2\ u_1)\ r(u_1\ x_2\ u_2)\ s(x_1\ x_2\ u_2)$

r_e $s_1(x_1\ x_2\ x_3) \leftarrow f(x_1\ x_2\ x_3)$
r_r $s_1(x_1\ x_2\ x_3) \leftarrow f(x_3\ x_2\ u_1)\ r(u_1\ x_2\ u_2)\ s_1(x_1\ x_2\ u_2)$

Note that these rules satisfy the Assumptions But necessary condition (i) is violated, since x_3 is selected, but is not placed at position 3 by Z^1 If the EDB contains only the following tuples $f(a_1\ a_2\ b_2)$, $f(a_3\ a_2\ c_2)$, $f(b_1\ a_2\ c_1)$, $r(b_1\ a_2\ b_2)$ and $r(c_1\ a_2\ c_2)$, then, $(a_1\ a_2\ a_3)$ is a tuple of s but not a tuple of s_1, as shown in Fig 3 2

Example 3 4 Let s and s_1 be defined by

r_e $s(x_1\ x_2\ x_3) \leftarrow f(x_1\ x_2\ x_3)$
r_r $s(x_1\ x_2\ x_3) \leftarrow s(x_1\ u_1\ u_2)\ r(u_2\ x_2\ u_3)\ s(u_3\ x_2\ x_3)$

r_e $s_1(x_1\ x_2\ x_3) \leftarrow f(x_1\ x_2\ x_3)$
r_r $s_1(x_1\ x_2\ x_3) \leftarrow f(x_1\ u_1\ u_2)\ r(u_2\ x_2\ u_3)\ s_1(u_3\ x_2\ x_3)$

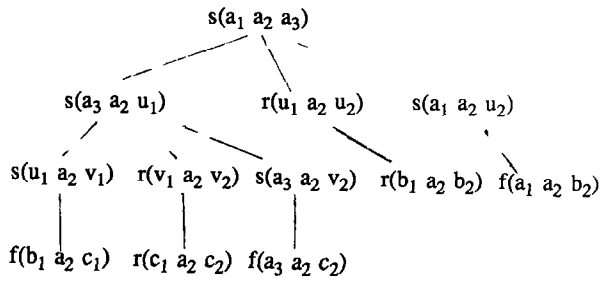

Fig 3 2

The necessary condition (ii) is not satisfied because that x_2 is selected by W, but not by Z^1 If the EDB contains only $f(b_3\ a_2\ a_3)$, $f(c_3\ b_1\ b_2)$, $f(a_1\ c_1\ c_2)$, $r(b_2\ a_2\ b_3)$ and $r(c_2\ a_2\ c_3)$, then $(a_1\ a_2\ a_3)$ is a tuple of s in IDB, but is not a tuple of s_1

4 Proof of Proposition 3 1

In this Section, we give a proof of Proposition 3 1 Let s(X)<==>s_1(X) for any X and the equivalence is independent of data contained in EDB We now show that the rule has the properties (i) and (ii) as stated in Proposition 3 1 by setting s(A)<==>s_1(A) in a particular EDB with the following properties

(1) there exist three constant vectors

$$A = (a_1\ a_2 \quad a_n)$$
$$B = (b_1\ b_2 \quad b_p)$$
$$C = (c_1\ c_2 \quad c_p)$$

with all a's, b's and c's distinct to each other, and

(2) there are two relations, f and r, which contain the following tuples only $f((A,B)Z^2)$, $f((((A,B)Z^1),C)Z^1)$, $f((((A,B)Z^1),C)Z^2)$, $r((A,B)W)$ and $r((((A,B)Z^1),C)W)$ (In order to avoid writing too many parentheses, in the remaining part of this Section, we use a shorthand which leaves out all parentheses and commas, e g $(((A,B)Z^1),C)Z^2$ is written as ABZ^1CZ^2, and

(3) A is a tuple of s in IDB and the minimal rule/goal tree to establish s(A) is the one shown in Fig 4 1

Since s and s_1 are equivalent, s_1(A) can be established by a finite rule/goal tree of the form as shown in Fig 4 2 That is, there exists a substitution which let each leaf node of the tree unify with a tuple in EDB, so that, s_1(A) can be obtained at the root

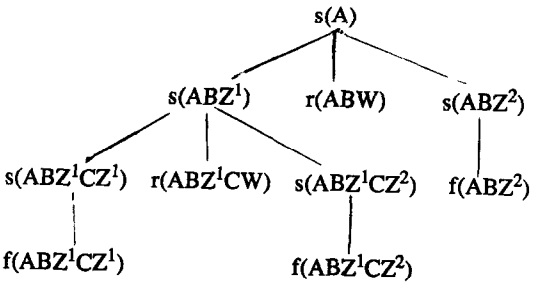

Fig 4 1

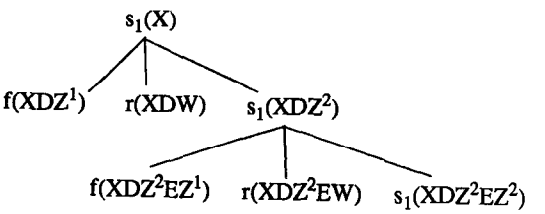

Fig 4 2

The substitution requires that in the s_1 tree, each variable in X, D, E, , is substituted by exactly one value from $\{a_1 \quad a_n\ b_1 \quad b_p\ c_1 \quad c_p\}$ In the rest of the proof, we do not differentiate X from A, since x_i=a_i for $1 \le i \le n$

In order to obtain the requirements on Z^1, Z^2 and W of the set of rules, we will unify each leaf node in s_1 tree with some node in s tree, which is a tuple in the EDB

The crucial facts that we frequently reference in our proof are

Fact 1 There is at least one tuple in EDB which unifies with a leaf node in the tree of s_1

Fact 2 The tree of s_1 is finite

Fact 3 By Assumption 2, the restrictions on the tuples in our EDB can be stated as follows

3 1 c_i is at position i_1 in $f(ABZ^1CZ^1)$, but is not in any other f tuples

3 2 c_j is at position j_1 in $f(ABZ^1CZ^2)$, but is not in any other f tuples

3 3 In $r(ABZ^1CW)$, c_i and c_j are at positions i_2 and j_2, respectively

3 4 b_j is at position j_1 in $f(ABZ^2)$, but is not in any node of the subtree rooted by $s(ABZ^1)$

3 5 In r(ABW), b_i and b_j are at positions i_2 and j_2, respectively

3 6 b_i is at position i_1 in $s(ABZ^1)$, but is not in $f(ABZ^2)$

Before we proceed any further, there are some notes about the structure of the trees of s and s_1 that we would like to make

1 We choose the tree of s to be of the form as shown in Fig 4 1, because it is the smallest non-trivial tree for $s(X)<==>s_1(X)$

2 Since $s_1(X)==>s(X)$, the tree of s_1 has at least 4 levels, as shown in Fig 4 2, otherwise it can easily be shown that s(A) will have a smaller tree

In the proof of Proposition 3 1, we will find a unique unification for each leaf node in s_1 tree As a result, we shall establish a set of properties for Z^1, Z^2 and W Then, conditions (i) and (ii) of Proposition 3 1 will be easily derived from these properties

Let the EDB, s and s_1 be defined as given above We start to find the unifications for leaf nodes in s_1 tree

Consider the node r(XDW) There are only two r tuples, $r(ABZ^1CW)$ and r(ABW), which may unify with it

Case 1 r(XDW) unifies with r(ABW)

By Assumption 2 and Fact 3 5, $d_i=b_i$ and b_i is at position i_1 in node $f(XDZ^1)$, $d_j=b_j$ and b_j is at position j_1 in $s_1(XDZ^2)$ Thus by Fact 3 1 and 3 6, $f(XDZ^1)$ must unify with $f(ABZ^1CZ^2)$ By Assumption 1, $d_j=b_j$ must appear in some child node of $s_1(XDZ^2)$

Subcase 1 $d_j=b_j$ and b_j is at some position i_3 in node $f(XDZ^2EZ^1)$

That is, Z^1 selects x_{j_1} and places it at position i_3 By Fact 3 4, $f(XDZ^2EZ^1)$ may possibly unify with $f(ABZ^2)$ only But, if that is the case, there is no tuple to unify with $r(XDZ^2EW)$, because the value of e_i in $r(XDZ^2EW)$ can only be b_i or c_i, but none of them is in $f(ABZ^2)$, by Fact 3 1 and

Fact 3 6 Thus we obtain a contradiction to Fact 1

Subcase 2 $d_j=b_j$ and b_j is at some position t in node $r(XDZ^2EW)$

Thus W selects x_{j_1} and places it at position t. Therefore, a_{j_1} is at position t in r(ABW) Thus $r(XDZ^2EW)$ can not unify with r(ABW) By Fact 3 4, $r(ABZ^1CW)$ can not unify with $r(XDZ^2EW)$ Thus there is contradiction to Fact 1

Subcase 3 $d_j=b_j$ and b_j is at some position j_3 in $s_1(XDZ^2EZ^2)$

That is, Z^2 selects x_{j_1} and places it at position j_3 The tree structure below $s_1(XDZ^2EZ^2)$ depends on whether the rule r_e or the rule r_r is applied

Subcase 3 1 r_e is applied There is a single leaf node, $f(XDZ^2EZ^2)$, under the s_1 node, and b_j is at position j_3 in it Since a_{j_1} is now at position j_3 in $f(ABZ^2)$ and by Fact 3 4, b_j is not in the other two f tuples, there is no tuple to unify with $f(XDZ^2EZ^2)$

Subcase 3 2 r_r is applied By Assumption 1, $d_j=b_j$ must appear in some child node of $s_1(XDZ^2EZ^2)$ Then similar arguments in the three subcases can be applied repeatly Thus we obtain contradictions to either Fact 1 or Fact 2

Case 2 Since Case 1 is shown to be impossible, we must have r(XDW) unify with $r(ABZ^1CW)$

Let this unification be denoted by U1 By Fact 3 3, U1 implies that $d_i=c_i$ and $d_j=c_j$

We first consider $d_i=c_i$ Now, c_i is at position i_1 in $f(XDZ^1)$ by Assumption 2 By Fact 1 and Fact 3 1, $f(XDZ^1)$ must unify with $f(ABZ^1CZ^1)$

We denote this unification by U2 From U1 and U2, we immediately obtain several properties of the selectors

Prop 1 For every distinguished variable, x_i, selected by Z^1, the variable is placed at position i

This is because that, if Z^1 selects x_i and places it to some position k, then a_i is at position k in $s(ABZ^1)$ (see s tree in Fig 4 1), and U2 implies that, a_i is at position k in $f(ABZ^1CZ^1)$, thus, it is also at position i in $s(ABZ^1)$ Therefore, a_i is at both position i and position k in $s(ABZ^1)$ By Assumption 3, k=i

Prop 2 For every distinguished variable, x_i, selected by W, x_i is also selected by Z^1

This can be shown by letting W select x_i and place it to some position p Then, U1 implies that a_i is at position p in $r(ABZ^1CW)$, thus it is also in $s(ABZ^1)$

Prop 3 Any x_i not selected by Z^1 is selected by Z^2

This is because that, Z^1 can not select all x's, since otherwise, by Prop 1, the recursive rule of s induces an infinite loop Therefore, this property follows from Prop 1, 2 and Assumption 1

Consider the vector X By Prop 1, 2 and 3, the set of positions in X can be partitioned into three sets, $P(X, \overline{Z^1})$, $P(X, Z^1W)$ and $P(X, Z^1-W)$,

such that, position j is in $P(X, \overline{Z^1})$ iff x_j is not selected by Z^1, it is in $P(X, Z^1W)$ iff x_j is selected by both Z^1 and W, it is in $P(X, Z^1-W)$ iff x_j is selected by Z^1 but not by W

Prop 4 All non-distinguished variables selected by Z^1 are placed at positions in $P(X, \overline{Z^1})$, and Z^1 fills all positions in $P(X, \overline{Z^1})$ with non-distinguished variables

This follows from Prop 1 and Prop 2 directly

We now return to U1 which implies $d_j=c_j$ Now, c_j is at position j_1 in $s_1(XDZ^2)$ We want to determine in which set of $P(X, \overline{Z^1})$, $P(X, Z^1W)$ and $P(X, Z^1-W)$ the position j_1 is

Claim 1 Position j_1 is not in $P(X, Z^1W)$

To show this, we assume the contrary and let W select x_{j_1} and place it at some position p Thus, c_j is at position p in $r(XDZ^2EW)$, a_{j_1} is at position p in r(ABW) and at position j_1 in $s(ABZ^1)$ and therefore at position p in $r(ABZ^1CW)$ Since both tuples of r in the s tree have a_{j_1} at position p, $r(XDZ^2EW)$ can not unify with any of them.

Claim 2 Position j_1 is not in $P(X, \overline{Z^1})$

To see this, we let position j_1 be in $P(X, \overline{Z^1})$ By Prop 3, Z^2 selects x_{j_1} and places it to some position h_1, which may or may not be in $P(X, \overline{Z^1})$

Case 1 h_1 is not in $P(X, \overline{Z^1})$

Consider node $s_1(XDZ^2EZ^2)$, c_j is now at position h_1

Subcase 1 rule r_e is applied to $s_1(XDZ^2EZ^2)$

A leaf nod $f(XDZ^2EZ^2)$ is obtained and c_j is at position h_1 in it By Fact 3 2, $f(XDZ^2EZ^2)$ can not unify with $f(ABZ^1CZ^1)$ nor with $f(ABZ^2)$ By Prop 4, some b is at position h_1 in $f(ABZ^1CZ^2)$ Thus, there is no tuple to unify with $f(XDZ^2EZ^2)$

Subcase 2 r_r is applied to node $s_1(XDZ^2EZ^2)$

We obtain a new f node $f(XDZ^2EZ^2GZ^1)$ as a child of $s_1(XDZ^2EZ^2)$ for some non-distinguished variable vector G By Prop 1, c_j is at position h_1 in $f(XDZ^2EZ^2GZ^1)$ Thus, we can apply the same argument as in Subcase 1 to obtain a contradiction to Fact 1

Case 2 h_1 is in $P(X, \overline{Z^1})$

Therefore, by Prop 3, we let Z^2 select x_{h_1} and place it at some position h_2, which, again, may or may not be in $P(X, \overline{Z^1})$ Consider $s_1(XDZ^2EZ^2)$, a_{j_1} is at position h_2 now If h_2 is not in $P(X, \overline{Z^1})$, the similar arguments as in Case 1 are applicable to obtain a contradiction If h_2 is in $P(X, \overline{Z^1})$, then, the argument here can be applied repeatly Since there is a finite number of positions in $P(X, \overline{Z^1})$, a contradiction to Fact 1 must occur

Prop 5 Position j_1 is in $P(X, Z^1-W)$

This is from Claim 1 and Claim 2

Prop 6 For every position i in $P(X, \overline{Z^1})$, Z^2 selects x_i and places it to some position in $P(X, \overline{Z^1})$ and all positions in $P(X, \overline{Z^1})$ are filled, by Z^2, with x's previously at positions in $P(X, \overline{Z^1})$

This is because the follows If i is in $P(X, \overline{Z^1})$, then Z^2 selects x_i, by Assumption 1 and Prop 3 Let Z^2 place x_i at some position q If position q is not in $P(X, \overline{Z^1})$, by Prop 1 and 5, x_i (equivalently, a_i) is at position q and c_j is at position j_1 in $f(XDZ^2EZ^1)$ By the definition of $P(X, \overline{Z^1})$, a_i is not in any node of the subtree rooted by $s(ABZ^1)$ (see Fig 4 1) By Fact 3 2, c_j is not in $f(ABZ^2)$ Therefore, no tuple can unify with $f(XDZ^2EZ^1)$ Thus q must be in $P(X, \overline{Z^1})$ Let X' be the set of x's at positions in $P(X, \overline{Z^1})$ Since the number of x's in X' is the same as that of positions in $P(X, \overline{Z^1})$, each position in $P(X, \overline{Z^1})$ is occupied by some x in X'

Now, we go on to find the unifications for remaining leaf nodes By Prop 1 and 5 and unification U1, c_j is now in $f(XDZ^2EZ^1)$ By Fact 1 and Fact 3 2, this implies that, $f(XDZ^2EZ^1)$ must unify with $f(ABZ^1CZ^2)$, denoted by U3 By Prop 4, the positions in $P(X, \overline{Z^1})$ in $f(XDZ^2EZ^1)$ are occupied by the e's and the positions in $P(X, \overline{Z^1})$ in $s(ABZ^1)$ are occupied by the b's Thus, by Prop 6, the positions in $P(X, \overline{Z^1})$ in $f(ABZ^1CZ^2)$ are occupied by the b's Since $f(XDZ^2EZ^1)$ unifies with $f(ABZ^1CZ^2)$, each e is equal to some b By Assumption 2, e_i at position i_2 in $r(XDZ^2EW)$ is equal to some b By Fact 3 3, $r(XDZ^2EW)$ can not unify with $r(ABZ^1CW)$ Thus, it must unify with r(ABW) We denote this unification by U4

Prop 7 For every x_i selected by W, x_i is selected by Z^2 and is placed at position i

This is because that, if W selects x_i and places it at some position p, then a_i is at position p in r(ABW) Consider node $r(XDZ^2EW)$, let the variable at position p be y By Prop 2, y appears at position i in node $f(XDZ^2EZ^1)$ Thus U4 implies $y=a_i$ and U3 implies that a_i is at position i in $f(ABZ^1CZ^2)$ Thus, by Prop 2 and Assumption 3, Z^2 selects a_i (equivalently, x_i) and places it at position i

By Assumption 2, U4 also implies that $e_j=b_j$ is at position j_1 in $s_1(XDZ^2EZ^2)$ If rule r_r is applied to it, we obtain three child nodes, $f(XDZ^2EZ^2GZ^1)$, $r(XDZ^2EZ^2GW)$ and $s_1(XDZ^2EZ^2GZ^2)$ for some non-distinguished variable vector G By Prop 1 and 5, b_j is at position j_1 in $f(XDZ^2EZ^2GZ^1)$ By Fact 3 4, $f(XDZ^2EZ^2GZ^1)$ can not unify with $f(ABZ^1CZ^1)$ nor with $f(ABZ^1CZ^2)$ Thus, $f(ABZ^2)$ is the only remaining candidate However, if the two nodes unify, then the positions in $P(X, \overline{Z^1})$ in $f(ABZ^2)$ are occupied by a's by Prop 6 and the corresponding positions in $f(XDZ^2EZ^2GZ^1)$ are occupied by g's, by Prop 4 Thus each g is equal to some a By Assumption 2, $r(XDZ^2EZ^2GW)$ has the a value at position i_2

Thus there is no tuple to unify with $r(XDZ^2EZ^2GW)$ By Fact 3 3 and 3 5 Therefore, only the rule r_e can be applied and the unification, denoted by U5, is to unify $f(XDZ^2EZ^2)$ with $f(ABZ^2)$

Prop 8 For every x_i selected by Z^2, x_i is placed at position i

This can be shown as the follows By Prop 7, Prop 8 is true for all x's selected by both Z^2 and W For each x_i selected by Z^2 but not by Z^1, let x_i be placed at some position p Then, a_i is at position p in $f(ABZ^2)$ and x_i is at position p in $s_1(XDZ^2)$ By Prop 6, position p is in $P(X, \overline{Z^1})$ Also by Prop 6, all positions in $P(X, \overline{Z^1})$ in $s_1(XDZ^2)$, and therefore the same positions in $f(XDZ^2EZ^2)$ are occupied by x's Thus, U5 requires x_i to be at position p in $f(XDZ^2EZ^2)$, and which in turn, implies that x_i is at position i in $s_1(XDZ^2)$ Since x_i is at both position i and position p, by Assumption 3, p=i For each x_i selected by both Z^1 and Z^2, but not by W, let Z^2 place x_i to some position j Note that position i is in $P(X, Z^1-W)$ Position j is also in $P(X, Z^1-W)$, since it has just been shown that Z^2 places each variable x_k in $P(X, \overline{Z^1}) \cup P(X, Z^1W)$ to position k Let the variable at position j in $f(XDZ^2EZ^2)$ be y Then, y is at position i in $s(XDZ^2)$ and by Prop 1, it is also at position i in $f(XDZ^2EZ^1)$ U5 implies $y=a_i$ If y is a non-distinguished variable, then, U3 implies y=c, for some c, and therefore, we have a contradiction since there is no c in $f(ABZ^2)$ Thus, y must be a distinguished variable and $y=x_i$

Now, we obtain all unifications of the leaf nodes in s_1 tree and a set of properties of Z^1, Z^2 and W It is easy to see that, Prop 1, 3 and 8 imply condition (i) and Prop 1, 2 and 7 imply condition (ii) of Proposition 3 1 □

5 Conclusion

This paper is an extension of our previous work We give a necessary condition on the placement of variables to convert a doubly recursive rule of restricted type into a single linear recursive rule This condition is a slight generalization of the sufficient condition that we obtained previously and it can also be proved sufficient by using the techniques in [TrYu] What we have achieved is the identification of a condition which differentiates a (doubly) nonlinear recursive rule from a trivial nonlinear recursive rule (one that is logically equivalent to a linear recursive rule) We believe that the techniques employed in this paper and in [YuZh] will be useful in analyzing recursive rules

We now comment on the Assumption 1, 2 and 3 Assumption 1 is essential for the analysis

The motivation of using Assumption 2 is that, by letting $s(A)<==>s_1(A)$ in a particular EDB, with a non-trivial minimal tree of s(A), it can be proved that, the recursive rule must have the following property

Z^1 (Z^2) shares some non-distinguished variable(s) with either Z^2 (Z^1) or W

By this property, rules are divided into different types according to how the variables are shared Assumption 2 corresponds to the type of rules in which, Z^1 and W share some non-distinguished variable(s) and Z^2 and W share some other non-distinguished variable(s) Thus, rules satisfying Assumption 2 satisfy the property given above We intend to generalize the result of this paper to include other types of rules

Assumption 3 is not really required With a further generalization of the necessary condition of Proposition 3 1, we can release Assumption 3 and allow a distinguished variable to be placed at more than one positions by Z^1 and/or Z^2

6 References

[BaRa] F Bancilhon and R Ramakrishnan, "An Amateur's Introduction to Recursive Query Processing Strategies", ACMSIGMOD, 1986

[BFMY] C Beeri, R Fagin, D Maier and M Yannakakis, "On the Desirability of Acyclic Database Schemas", JACM, 30(3), July, 1983

[BeCh] P Bernstein and D-M Chiu, "Using Semi-join to Solve Relational Queries", JACM, 1981

[BMSU] F Bancilhon, D Maier, Y Sagiv and J Ullman, "Magic Sets and Other Strange Ways to Implement Logic Programs", Principles of Data Base Conference, 1986

[BKBR] Beeri, C Kanellakis, P Bancilhon, F and Ramakriahnan, R "Bounds on the Progagation of Selection into Logic Programs", Principles of Data Base Conference, 1987 (to appear)

[CeGL] S Ceri, G Gottlob and L Lavazza, "Translation and Optimization of Logic Queries the Algebraic Approach", Proc of Twelfth Inter Conf on VLDB, Aug 1986

[Chan] C Chang, "On the Evaluation of Queries Containing Derived Relations in Relational Databases", In Advances in Data BAse Theory, Vol 1 H Gallaire, J Minker and J Nicholas, Plenum Press, N Y pp 235-260

[GoSh] N Goodman and O Shmueli, "Systactic Characterization of Tree Database Schemas", JACM, 30(4), Oct 1983

[HaLu] J Han and H. Lu, "Some Performance Result on Recursive Query Processing in Relational Database Systems", IEEE Proc of Inter Conf on Data Engineering, 1986

[HeNa] L Henschen and S Naqvi, "On Compiling Queries in Recursive First-Order Databases", JACM, 31(1), 1984

[Ioan] Y Ioannidis, "A Time Bound on the Materialization of Some Recursively Defined Views", Proc of VLDB, 1985

[IoWo] Y Ioannidis and E Wong, "An Algebraic Approach to Recursive Inference", Expert Database Systems, 1986

[LaYa] P Larson and H Yang, "Computing Queries from Derived Relations", VLDB, 1985, pp 259-269

[Lloy] J Lloyd, "Foundation of Logic Programming", Springer-Verlag, 1984

[Naug] J Naughton, "Data Independent Recursive in Deductive Databases", Proc of the Fifth ACM SIGACT-SIGMOD, March 1986

[NaSa] Naughton, J and Sagiv, Y "A decidable class of bounded recursions", Priciples of Data Base Conference 1987, to appear

[Reit] R Reiter, "Deductive Question Answering on Relational Database", In Logic and Databases, H Gallaire and J Minker, Plenum Press, N Y 1978, pp 149-177

[Sacc] D Sacca, "On the Implementation of a Simple Class of Logic Queries for Databases", Principles of Data Base Conference, 1986

[Shmu] Shmueli, O "Decidability and expressiveness aspects of logic queries", Principles of Data Base Conference 1987, to appear

[TrYu] D Troy and C Yu, "Transforming Nonlinear Recursive Rules to Linear Recursive Rules", Technical report, Univ of Ill at Chicago, 1986

[Ullm] J Ullman, "Implementation of Logical Query Languages for Databases", ACM Trans on Database Systems, Vol 10, Sep 1985

[Ullm2] J Ullman, "Principles of Database Systems", Computer Science Press, Rockville, M d , 1982

[Vard] M Vardi, "Complexity of Relational Queries", ACMSIGACT, 1982, pp 137-145

[YuOz] C Yu and M Ozsoyoglu,"An Algorithm for Tree-Query Membership of a Distributed Query", IEEE COMPSAC, 1979

[YuZh] C Yu and W Zhang, "Efficient Recursive Query Processing Using Wavefront Methods", IEEE Data Engineering, 1987 (to appear)

[Zani] C Zaniolo, "Safety and Compilation of Non-recursive Horn Clauses", Expert Database Systems, 1986, pp 167-178