Magic Counting Methods

Domenico Saccà †
University of Calabria, Rende, Italy
Carlo Zaniolo
MCC, Austin, Texas, USA

ABSTRACT

The problem considered is that of implementing recursive queries, expressed in a logic-based language, by efficient fixpoint computations In particular, the situation is studied where the initial bindings in the recursive predicate can be used to restrict the search space and ensure safety of execution Two key techniques previously proposed to solve this problem are (i) the highly efficient counting method, and (ii) the magic set method which is safe in a wider range of situations than (i) In this paper, we present a family of methods, called the magic counting methods, that combines the advantages of (i) and (ii) This is made possible by the similarity of the strategies used by the counting method and the magic set method for propagating the bindings This paper introduces these new methods, examines their computational complexity, and illustrates the trade-offs between the family members and their superiority with respect to the old methods

1. Introduction

The Counting Method [BSMU,SZ1,SZ2] and the Magic Set Method [BSMU, SZ1] are among the most significant techniques [BR] developed for supporting logic-based data languages such as Nail! [Ul] or LDL [TZ] The former method is normally more efficient than the latter, but the latter is safe in a wider range of situations than the former In this paper, we present a family of methods, called *Magic Counting Methods,* that combine the strengths of both The combination is made possible by the fact that the counting method and magic set method use similar strategies for propagating bindings

In this paper, we first introduce a general framework whereby the counting method and magic set method are compared and their computational behavior analyzed (Sections 2 and 3) This framework is then used to introduce and prove the correctness of the Independent Magic Counting Methods, and a refinement of these called the Integrated Magic Counting Methods (Sections 4 and 5) The rest of the paper discusses the the implementation of these methods and their computational complexities We find that there exists a well-defined efficiency hierarchy among them

Throughout the paper, we consider the query program Q composed by the query goal

$$P(a, Y)?$$

and the following rules

$$P(X, Y) - E(X, Y)$$

$$P(X, Y) - L(X, X_1), P(X_1, Y_1), R(Y, Y_1)$$

where P is a ***recursive predicate*** and E, L and R are ***database predicates*** The first rule, which does not contain any recursive predicate in the body, is called an ***exit rule,*** whereas the second rule is called a ***recursive rule*** The ***answer*** of the query is the set of pairs (a, b) that can be inferred from the logic program and from the facts stored in the (finite) database relations corresponding to E, L and R

The above query is an abstraction of a large class of similar queries, for instance to obtain the well-known same-generation example [BMSU], we can assume that both L and R correspond to a relation ***Parent*** storing the facts that the person X_1 (or Y_1) is parent of X (or

† Part of this work was done while this author was visiting at MCC

Y), and if we simplify the exit rule as

$$P(X, X)$$

thus, every person is of the same generation of himself, and the query asks for all persons who are of the same generation as a "

It is also easy to generalize this query by letting L and R be derived predicates or conjuncts of these, and X and Y be replaced by several arguments On the other hand, the above query has a clear graph interpretation where the methods and results can be shown without too heavy a notation Moreover, since a similar graph interpretation holds for a large class of queries, notably all queries that can be expressed using only one linear recursive rule, it turns out that all results found for the query at hand can immediately be extended to this important class, which was studied in [SZ1] under the name of canonical strongly linear queries Furthermore, further extensions to more general queries are possible, although more difficult and outside the scope of this paper

2. The Counting Method and the Magic Set Method

The counting and the magic set methods rewrite the rules and the query goal so that the answer can be constructed efficiently by means of two fixpoint computations For the example at hand, the counting method produces the following modified query program Q_C

COUNTING METHOD

$$CS(0, a) \quad (1)$$

$$CS(J+1, X_1) - CS(J, X), L(X, X_1) \quad (2)$$

$$P_C(J, Y) - CS(J, X), E(X, Y) \quad (3)$$

$$P_C(J-1, Y) - P_C(J, Y_1), R(Y, Y_1) \quad (4)$$

$$Answer(Y) - P_C(0, Y) \quad (5)$$

$$Answer(Y)?$$

Rules 1 and 2, are called counting rules as they define the counting set CS, while (3) and (4) are called modified rules ($J+1$ is an obvious notation of convenience, in actual Prolog we should write $J1$ instead and have a goal "$J1$ is $J+1$")

On the other hand, the magic set method produces the following query Q_M

MAGIC SET METHOD

$$MS(a) \quad (1)$$

$$MS(X_1) - MS(X), L(X, X_1) \quad (2)$$

$$P_M(X, Y) - MS(X), E(X, Y) \quad (3)$$

$$P_M(X, Y) - MS(X), L(X, X_1), \quad (4)$$
$$P_M(X_1, Y_1), R(Y, Y_1)$$

$$Answer(Y) - P_M(a, Y) \quad (5)$$

$$Answer(Y)?$$

Rules 1 and 2, are called the magic set rules as they define the the magic set MS, while (3) and (4) are called modified rules

Two queries are said to be *equivalent* when they have the same answer (as per the well defined semantics of Horn clause queries [VK]) Then, we have the following result that establishes the conceptual correctness of the magic set and counting methods

FACT 1 [SZ1] *The queries Q, Q_C and Q_M are equivalent* □

Let us next examine the problem of implementing these methods using a least fixpoint computation To that end, we use a convenient mixture of Horn clause and procedural programming notation

The counting set CS is computed by the following fixpoint computation (the semicolon is used as end-delimiter for both rules and statements)

COUNTING SET COMPUTATION

```
begin
    $CS(0, a)$,
    $J = 0$,
    while $CS(J, X_1)$ do
    begin
    $CS(J+1, X_1) - CS(J, X), L(X, X_1)$,
        $J = J+1$,
    end,
end
```

Thus, we assume that we are working with a variable two-column relation CS corresponding to the predicate In the fixpoint computation, we perform relational algebra operations on relations, which, for simplicity and expressivity of notation, we represent by their equivalent Horn-clause form Thus, in the first step, our variable relation is assigned the relation containing only the tuple $(0, a)$ The fixpoint computation step consists of taking the semijoin of the current relation with that representing $L(X, X_1)$ and incrementing the first column by one We thus assume that the new tuples so generated are added to the variable relation (e g , using Prolog assert) Thus, the test $CS(J, X_1)$ fails only after the last fixpoint iteration has failed to produce any new tuple As we will discuss extensively later, there are situations (i e , in the presence of cycles), where the loop test never fails and the fixpoint computation never terminates In these situations, we will say that the counting method is not *safe,* i e , the answer cannot be computed in a finite amount of time [Ul, SZ2]

Using the seminaive fixpoint approach [Ban, BaR], we can express the fixpoint computation of magic sets as follows (to implement the fixpoint computation we have introduced an index that records the steps at which a value is first introduced)

SEMINAIVE MAGIC SET COMPUTATION

```
begin
    $\overline{MS}(0, a)$,
    $I = 0$,
    while $\overline{MS}(I, X_1)$ do
    begin
        $\overline{MS}(I+1, X_1)$ -
            $\overline{MS}(I, X), L(X, X_1), not(\overline{MS}(\_, X_1))$,
        $I = I+1$,
    end,
end
```

The final magic set can be computed as

$$MS(Y) - \overline{MS}(\_, Y),$$

i e , by projecting the index out

The difference between the magic set computation and counting set fixpoint is thus clear, this difference reduces to the presence of $not(\overline{MS}(\_, X_1))$ in the magic set fixpoint, i e , to the condition that $(I+1, X_1)$ is not added to $\overline{MS}$ if (J, X_1) is already in $\overline{MS}$ (for some $0 \leq J \leq I$) Such negation can be implemented using the set difference operation of relational algebra

3. Computational Complexity of Methods

In order to characterize the data underlying the query and to obtain insight on the behavior of the methods, we need to associate a graph G with the query, in the following way

a) For each value in the domains underlying L or R, there is a corresponding node in G If the same value appears both in L and R then there will be two distinct associated nodes in G (one may think they have some label to distinguish them) Therefore, the nodes of G are partitioned into L*-nodes* and R*-nodes*

b) For each pair (b, c) in L, there is an arc (b, c) in G, where both b and c are L-nodes

c) For each pair (b, c) in E such that b and c are also database constants of L and R, respectively, there is an arc (b, c) in G where b is an L-node and c is an R-node

d) For each pair (b, c) in R, there is an arc (c, b) in G, where both b and c are R-nodes

The ***query graph*** $G_Q = <N, A>$ is the subgraph of G, induced by all nodes that are reachable from a (recall that a is the constant in the query goal) We call a the ***source node*** of G_Q The query graph G_Q, in turn, is composed by the three subgraphs $G_L = <N_L, A_L>$, $G_E = <N_E, A_E>$ and $G_R = <N_R, A_R>$, such that A_L, A_E and A_R are all the arcs in A corresponding to pairs in L, E and R, respectively It is easy to see that N_L and N_R are disjoint and contain L-nodes and R-nodes, respectively, and $N_L \cup N_R = N$ On the other hand, G_E is a bipartite graph having arcs from L-nodes to R-nodes Finally, A_L, A_R and A_E are disjoint and $A_L \cup A_R \cup A_E = A$ The number of respective nodes of G_Q, G_L, G_R and G_E will be denoted by n, n_L, n_R and n_E, while the number of respective arcs is denoted by m, m_L, m_R and m_E

Let b and c be two nodes in the query graph If there is a path from b to c with length k, we say that c has a distance k from b

Consider the graph G_L It follows directly from the definitions that the nodes of this graph are the magic set values, $N_L = MS$ Thus we will call G_L the ***Magic Graph*** and refer to magic graph nodes and magic set values as synonyms The counting set CS consists of pairs (j, b) where b is a node in the magic graph and j is its distance from the source node a, as it will be shown below The set of values obtained from CS by projecting the indices out will be denoted CS_{-1}, obviously $CS_{-1} = MS$ Moreover, let I_b denote the set of indices j such that (j, b) is in CS Then a node b will, respectively, be called

a) ***single*** if I_b is a singleton set ,

b) ***multiple*** if I_b has a finite cardinality greater than one,

c) ***recurring*** if I_b is infinite

The magic graph of a query will be called ***regular,*** when all its nodes are single and non-regular otherwise

For the query graph of Figure 1, G_L is the subgraph induced by the nodes a, a_1, , a_5, while G_R is the subgraph, represented by darker arcs, induced by the nodes b_1, , b_9 The graph G_E is composed by the dashed arcs in Figure 1 The magic graph is regular since all nodes in G_L are single, i e , they have a unique distance from a If we add the tuple $<a_2, a_5>$ to the relation L then the query becomes acyclic and the node a_5 becomes multiple, instead, if we add the tuple $<a_5, a_2>$ then the query becomes cyclic and the nodes a_2, a_3 and a_5 become recurring

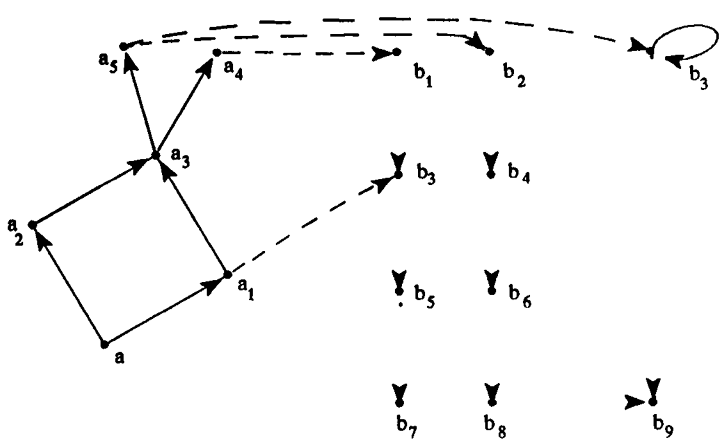

Fig 1. Query Graph

Proposition 1 summarizes the properties of query graphs

PROPOSITION 1 *Let Q be a query and G_Q be the query graph Then $MS = CS_{-1} = N_L$ In addition, given a node b in G_L,*

a) *b is single if and only if all directed paths from the source node a to b in G_L have the same distance,*

b) *b is multiple if and only if there at least two directed paths from a to b in G_L with different length,*

c) *b is recurring if and only if there is a cyclic directed path from a to b in G_L, and*

d) *the set I_b of indices associated to b coincides with the set of all distances of b from a*

As shown in [SZ1,MPS], there is also a simple graph based interpretation of the query answer

FACT 2 *A node b is in the answer of Q if there is a (possibly cyclic) directed path from the source node a to b in G_Q such that this path is composed by exactly k arcs from A_L, one arc from A_E and k arcs from A_R, where k is any non-negative integer* ◻

Consider the query instance whose graph is shown in Figure 1 Then b_5 is in the answer because of the path a, a_1, b_3, b_5 The other answer nodes are b_7, b_8, b_9 and b_3 Note that the latter node is in the answer because of the cyclic path

$$a, a_1, a_3, a_5, b_3, b_3, b_3, b_3$$

The path from a to b_9 is cyclic as well

Within the graph formalism, interesting complexity results about the magic set and the counting methods have been found in [MPS] The costs of the two methods are summarized in Table 1 for the different kinds of Magic Graphs (MG) The basic cost unit is the cost of retrieving a tuple in a database relation In this table and throughout the paper, we use the notations O and Θ for describing asymptotic time complexity If the cost function of an algorithm is $f(n)$, where n is the problem size, and $g(n)$ is another function of n, then

a) $f(n) = O(g(n))$, if there exists a constant d such that $f(n) \leq d \times g(n)$ for all but some finite (possibly empty) set of non-negative values for n, and

b) $f(n) = \Theta(g(n))$, if both $f(n) = O(g(n))$ and $g(n) = O(f(n))$

MG	Counting	Magic Set
Regular	$\Theta(m_L + n_L \times m_R)$	$\Theta(m_L \times m_R)$
Acyclic	$\Theta(n_L \times m_L + n_L \times m_R)$	$\Theta(m_L \times m_R)$
Cyclic	*unsafe* †	$\Theta(m_L \times m_R)$

Tab 1 *Costs of the counting and magic set methods*

PROPOSITION 2 *Let C and Ms be the costs of the counting method and of the magic set method, respectively Then*

a) *If the magic graphs are regular then $C = O(Ms)$*

b) *If the magic graphs are acyclic and $m_L = O(m_R)$, then $C = O(Ms)$*

PROOF Since the number of arcs is always greater than the number of nodes, we have that $n_L = O(m_L)$ Hence, $n_L \times m_R = O(m_L \times m_R)$, thus, part a) of the proposition holds Furthermore, if $m_L = O(m_R)$, then obviously $n_L \times m_R = O(m_L \times m_L)$ Therefore, also part b) of the proposition is proved ◻

Proposition 2 says that the counting method always works better than the magic set method when the magic graphs are regular In addition, since it is realistic to assume that m_R is, on the average, of the same order of m_L, it is fair to say that the counting method, on the average, works better than the magic set method when there is no cycle In fact, in the average, we have that $C = \Theta(n \times m)$ and $Ms = \Theta(m \times m)$ Note that having $m_L >> m_R$ is not sufficient for the magic set

† In [MPS] it has been shown that the counting method can be extended to deal with cyclic graphs and its cost is $\Theta(m \times n^3)$ Also note that the costs for the magic set method are actually higher than those given in Table 1, since arcs not in G_Q can be developed at each step of fixpoint computation using the modified rules For simplicity, these costs can be neglected since they simply reinforce the superiority of magic counting methods

method to work better than the countıng method

Thus the countıng method ıs superıor to the magıc set method ın terms of worst case behavıor This superıorıty ıs even more dramatıc when typıcal behavıor ıs consıdered, ın the comparatıve study presented ın [BR] the countıng method was shown to be more effıcıent than all other methods (ıncludıng the magıc set method but excludıng the [HN] method whıch ıs comparable performance-wıse) Unfortunately, the potentıal presence of cycles ın the database compromıses the applıcabılıty of the countıng method ın many sıtuatıons Note that, a database beıng logıcally acyclıc (e g , a non-ıncestuous famıly tree) does not guarantee that the physıcal database ıs cycle free, sınce checkıng acyclıcıty upon updates ıs very expensıve and not often done ın practıce -- thus there could be accıdental cycles that throw the countıng method astray Therefore, a method that combınes the performance of the countıng set method wıth the safety of the magıc set method ıs hıghly desırable

4. Magic Counting Methods

We now propose a famıly of methods that combıne the magıc set and the countıng methods and that are, therefore, called *magıc countıng* methods All methods ın the famıly make use of a ***restrıcted magıc set*** and a ***restrıcted countıng set*** A ***restrıcted*** magıc set, denoted by RM, ıs any (possıbly empty and not necessarıly proper) subset of MS Lıkewıse, RC wıll denote any (possıbly empty and not necessarıly proper) subset RC of CS, whıle $RC_{-\iota}$ denotes the set of values ın RC wıthout theır ındıces In addıtıon, for each b ın $RC_{-\iota}$, RI_b ıs the set of all ındıces assocıated to b ın RC (obvıously $RI_b \subseteq I_b$)

The general structure of the magıc countıng methods consısts of two steps In the fırst step, a restrıcted magıc set RM and a restrıcted countıng set RC ıs constructed, ın the second step, both the magıc countıng method and the magıc set method are applıed usıng the restrıcted sets Thıs second step ıs ımplemented as follows

MODIFIED RULES & QUERY FOR INDEPENDENT MC METHODS

$$P_C(J, Y) - RC(J, X), E(X, Y) \qquad (1)$$

$$P_C(J\text{-}1, Y) - P_C(J, Y_1), R(Y, Y_1) \qquad (2)$$

$$P_M(X, Y) - RM(X), E(X, Y) \qquad (3)$$

$$P_M(X, Y) - MS(X), L(X, X_1), P_M(X_1, Y_1), R(Y, Y_1) \qquad (4)$$

$$Answer(X) - P_C(0, X) \qquad (5)$$

$$Anwer(X) - P_M(a, X) \qquad (6)$$

$Answer(X)?$

Notıce that the predıcate RC has replaced CS ın the exıt rule of P_C (Rule 1), whıle RM has replaced the orıgınal MS ın the exıt rule for P_M (Rule 3) The recursıve rules for both P_C and P_M (Rules 2 and 4) have remaıned as ın the orıgınal magıc set and countıng methods It ıs easy to see that the P_C and the P_M rules operate ındependently from each other Therefore, these magıc countıng methods wıll be called *ındependent*

Let us next consıder the ıssue of correctness of the magıc countıng methods A method wıll be saıd to be *correct* ıf ıt generates the same answer as the orıgınal query *for all possıble databases* (ı e , by FACT 1, the same answer as the magıc set method or the countıng method) We then have thıs ımportant result

THEOREM 1 ***An ındependent magıc countıng method ıs correct ıf and only ıf the followıng two condıtıons hold***

a) $RM \cup RC_{-\iota} = MS$, and

b) for each b ın $RC_{-\iota}-RM$, $RI_b = I_b$

PROOF *Only-ıf part* Let M be a correct ındependent magıc countıng method We fırst prove that, for each ınstance of the query Q, $RM \cup RC_{-\iota} = MS$, where MS ıs the magıc set of the query and RM and RC are the reduced sets constructed by M We carry out the proof by contradıctıon Suppose that, for a gıven query ınstance $\hat{Q}$, a node b ıs ın MS but not ın $RM \cup RC_{-\iota}$ Let k be the length of any (non-cyclıc) path from the source node a to b (by Proposıtıon 1, such a path exısts) We construct another ınstance of Q by modıfyıng the query graph as follows We add the new nodes b_k, b_{k-1}, , b_0 to G_R, and also ıntroduce the arc (b, b_k) ın G_E and the arcs (b_k, b_{k-1}), , (b_1, b_0) ın G_R It ıs easy to see that the new graph corresponds to a new query ınstance $\bar{Q}$ Moreover, by Fact 2, the node b_0 ıs ın the answer of $\bar{Q}$ Sınce the reduced sets are constructed by M ındependently from the database predıcates E and R (ı e , G_E and G_R), the reduced sets for $\bar{Q}$ remaın unchanged and do not contaın the node b Hence, sınce only those arcs of A_E startıng from a node ın MS or $RC_{-\iota}$ are used ın the second step of the method M (see Rules 1 and 3 of Step 2 for ındependent methods), the arc (b, b_k) and, hence, the path from a to b_0 ıs not taken ınto account Therefore, the method M does not generate the node b_0 -- a contradıctıon Let us now agaın proceed ab absurdo to prove that for each b ın $RC_{-\iota}-RM$, $RI_b = I_b$ Suppose not and let k be an ındex ın I_b but not ın RI_b We modıfy the query graph of the query instance $\hat{Q}$ as before The magic set does not use the new arc (b, b_k) because b is not in RM and it cannot generate the pair (a, b_0) The counting method uses the node b but, since RI_b does not contain the index k, the node b_k does not have the index k Hence, a path from b_k to b_0 with length k cannot be constructed by repeatedly using Rule 2, i e, b_0 is not marked as an answer (contradiction)

If part Let us now suppose that, for any query instance $\hat{Q}$, the independent magic counting method M constructs the reduced sets in such a way that $RM \cup RC_{-i} = MS$ and for each b in RC_{-i}-RM, $RI_b = I_b$ We have to prove that M is correct Consider any node b_0 in the query answer By Proposition 1, there exists a path from a to b_0 composed by k arcs in G_L, the arc (b, b_k) in G_E and the arcs (b_k, b_{k-1}) , , (b_1, b_0) in G_R If b is in RC_{-i}-RM, the index k is in I_b by Proposition 1 and, then, in RI_b since $RI_b = I_b$ by hypothesis The index k is passed to b_k by Rule 1 Hence, the counting method assigns the index 0 to b_0 by repeatedly applying Rule 2 Therefore, the node b_0 is included in the answer by Rule 5 Suppose now that b is not in RC_{-i}-RM We have that b is in N_L because it is the source node of an arc in G_E Hence, since $N_L = MS$ by Proposition 1 and $RM \cup RC_{-i} = MS$ by hypothesis, b is in RM Rules 3 and 4 generate all pairs (c, d) such that (b, e) is in A_E and there are paths from c to b and from e to d with the same length Hence, the pair (a, b_0) is also generated and the node b_0 is included in the answer by Rule 6 This concludes the proof □

Theorem 1 allows us to divide the nodes of the magic graph into the set RC that uses the counting method and the set RM that uses the magic set method Since the counting method is better than the magic method for all nodes but the recurring ones, the ideal solution assigns the recurring nodes to RM and all others to RC However, this ultimate goal is not easy to reach, because of the added complexity of detecting recurring nodes, thus, we present three alternative methods that approximate the ultimate goal with solutions that offer practical advantages of their own Since detecting non-regular graphs is easier than detecting cyclic ones, these methods use the regularity of the magic graph as their decision criterion

The simplest method to implement is the basic method, as follows

a) *Basic Method* If the graph G_L is regular then $RM = \emptyset$ and $RC = CS$, otherwise $RM = MS$ and $RC = \emptyset$ The basic method coincides with the counting method in the former case and with the magic set method in the latter case

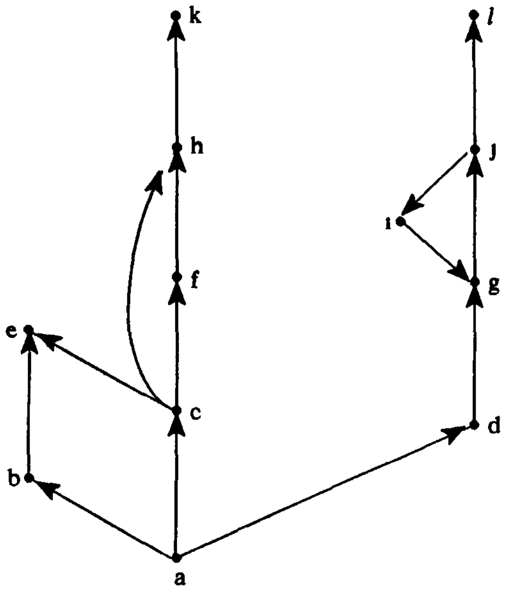

Fig 2. Magic Graph

For instance, the graph G_L of Figure 2, is not regular, thus $RM = MS = \{a, b, \quad , l\}$ and $RC = \emptyset$

While the basic method removes the compile-time dilemma of having to chose between counting and magic sets, it is clearly suboptimal in the sense that the counting method should still be used for the parts of the graph which do not contain any multiple or recurring nodes The next method accomplishes that by recording the level at which non-regular nodes are first found

b) *Single Method* Let i_s be the maximum index such that all nodes in CS_{-i} having an index less then i_s are single Then, RC_{-i} is the set of all (single) nodes with index less than i_s, and $RM = MS$-RC_{-i}

In Figure 2, for example, we have i_s=2, $RC_{-i} = \{a, b, c, d\}$ and $RM = \{e, f, \quad , l\}$

Using an index to partition the graph horizontally represents too coarse a criterion, since nodes in different vertical branches of the graph are smeared together For the example of Figure 2, for instance, the nodes $e, \quad , h$ are assigned to RM, although they are single The next method solves this problem

c) *Multiple Method* RC_{-i} is the set of all single nodes and $RM = MS$-RC_{-i} (i e, RM contains all multiple and recurring nodes)

For the example of Figure 2, we have $RC_{-i} = \{a, b, c, d, e, f\}$ and $RM = \{g, h, i, j, k, l\}$

Our final method uses counting for both single and multiple nodes

d) *Recurring Method* $RC_{-\iota}$ is the set of all single and multiple nodes and $RM = MS - RC_{-\iota}$ (thus, RM contains all recurring nodes)

For the magic graph G_L in Figure 2, the Recurring Method will produce, $RC_{-\iota} = \{a, b, c, d, e, f, h, k\}$ and $RM = \{g, i, j, l\}$

5. Integrated Magic Counting Methods

Before turning to the actual computation of the reduced set, let us observe how, in the last three methods, the RM nodes have been relegated to the part of the graph most remote from the source-- i e, to the upper part of Figure 2 As the magic set computation for these nodes progresses, it moves to the lower part of the graph (i e, closer to the source node) where no recurring node exists -- thus it can be improved by using the counting method The integrated magic counting methods embody this improvement

MODIFIED RULES & QUERY FOR INTEGRATED MC METHODS

$$P_M(X, Y) - RM(X), E(X, Y) \qquad (1)$$

$$P_M(X, Y) - RM(X), L(X, X_1), P_M(X_1, Y_1), R(Y, Y_1) \qquad (2)$$

$$P_C(J, Y) - RC(J, X), L(X, X_1), P_M(X, Y), R(Y, Y_1) \qquad (3)$$

$$P_C(J, Y) - RC(J, X), E(X, Y) \qquad (4)$$

$$P_C(J{-}1, Y) - P_C(J, Y_1), R(Y, Y_1) \qquad (5)$$

$$Answer(X) - P_C(0, X) \qquad (6)$$

$Answer(X)?$

Rules 1 and 2 are those of the magic set method, whereas Rules 4 and 5 are those of the counting method Rule 3 contains the recursive predicate P_M in its body Nevertheless, if the magic set method (first two rules) are applied before the counting method, then P_M is already solved and Rule 3 can be considered as an exit rule Indeed, this rule transfers the results of the magic set method to the counting method Therefore, these magic counting methods are called *integrated*

THEOREM 2 *An integrated magic counting method is correct if and only if the following conditions hold a)* $RM \cup RC_{-\iota} = MS$, *b) for each* b *in* $RC_{-\iota} - RM$, $RI_b = I_b$, *and c) the pair* $(0,a)$, *where* a *is the source node of the query graph, is in* RC

PROOF It follows the lines of the proof of Theorem 1 □

Also for integrated methods, we may have basic, single, multiple and recurring methods, according to the way the reduced sets are constructed However, by Theorem 2, the reduced counting set RC cannot be empty Then we shall assume that an empty RC actually means that RC only contains the pair $(0,a)$ The integrated single method coincides with the magic counting method proposed in [SZ1]

We are interested in magic counting methods that are not only correct but also *safe*, i e, the answer to the query can be found in a finite amount of time via a fixpoint computation Let Step 1 denote the computation of the sets RM and RC The following proposition states that the safety of a magic counting method only depends on Step 1

PROPOSITION 3 *An (integrated or independent) magic counting method is safe if and only if for each instance of the query* Q, *the computation of Step 1 is safe* □

In the remaining sections, we discuss the four approaches for computing the reduced sets and we compare the corresponding methods for efficiency To this end, we shall denote the cost function of the basic method by B, single method by S, multiple method by M, or recurring method by R A subscript denotes whether the method is independent (IND) or integrated (INT) For instance M_{IND} is the cost function of the independent multiple method and S_{INT} is the cost function of the integrated single method Since the integrated basic method practically coincides with the independent one, we denote by B the cost function of the two of them Finally, we continue to denote the cost functions of the counting method and the magic set method by C and Ms, respectively Also, if M' and M are two methods, then $M' \leq_q M$ will denote that $O_{M'} = O(O_M)$ for magic graphs of type q, where $q = R$ stands for regular magic graphs, $q = A$ stands for non-regular acyclic ones and $q = C$ stands for non-regular cyclic ones When $M' \leq M$ and vice versa, we write $M' = M$ When $M' \leq M$ holds only on the average case (i e, if the bound $m_L = O(m_R)$ holds as it will happen on the average), then we use the notation $M' \lesssim M$ For instance, the part a) and b) of Proposition 2 are expressed as $C \leq_R Ms$ and $C \lesssim_A Ms$, respectively We have $Ms \leq_C C$ as well

6. Basic Magic Counting Methods

The basic magic counting methods just detect whether there is some multiple node in the magic graph When no such a node occurs, they use the counting method, otherwise they use the magic set method Thus, Step 1 is implemented as follows we extend the magic set predicate with an additional argument that records whether this is the first occurrence of a node (1), or a successive one (2) Only first occurrences are used in the following steps, the multiple occurrences are not

```
begin
  $\overline{MS}(0, 1, a)$,
  $I = 0$,
  while $\overline{MS}(I, 1, X_1)$ do
  begin
    $\overline{MS}(I+1, C, X_1)$ - $\overline{MS}(I, 1, X), L(X, X_1)$,
      if $\overline{MS}(\_, 1, X_1)$ then $C=2$ else $C=1$,
    $I = I+1$,
    end,
end
```

(note the use of *if -then -else* construct with the obvious meaning as per any implementation of Prolog)

At the end of the above fixpoint computation, if all nodes are single (i e , there are no tuples $(I, 2, Y)$ in $\overline{MS}$), then $RM = \emptyset$ and RC is computed as follows

$$RC(I, Y) - \overline{MS}(I, 1, Y)$$

On the other hand, if there is at least one multiple (or recurring) node (i e , there are at least one tuple $(I, 2, Y)$ in $\overline{MS}$), then $RC = \emptyset$ (or $RC = \{(0,a)\}$, if the method is integrated) and RM is computed as follows

$$RM(Y) - \overline{MS}(\_, 1, Y)$$

Then the following proposition follows directly from the definitions

PROPOSITION 4 *The basic magic counting methods are correct and safe and their costs are the ones shown in Table 2 Furthermore,* $B =_R C$, $B =_{A,C} Ms$, $B \leq_C C$, *and* $C \lesssim_A B$

□

MG	Independent or Integrated
Regular	$\Theta(m_L + n_L \times m_R)$
Non-Regular	$\Theta(m_L \times m_R)$

Tab 2 *Costs of basic magic counting methods*

7. Single magic counting methods

The single magic counting methods represent a simple extension of the basic methods They perform the same fixpoint computation as basic methods, but, at the end, they construct the reduced sets in a different way If all nodes are single then $RM = \emptyset$ and RC is constructed from $\overline{MS}$ by projecting out the second index (as basic methods do) If there is at least one multiple (or recurring) node then the methods select the maximum index i_s for which all nodes with index $i \leq i_s$ are single Then, RM and RC are computed in the following way

$$RM(I, Y) - \overline{MS}(I,1, Y), I \geq i_s$$

$$RC(I, Y) - \overline{MS}(I,1, Y), I < i_s$$

(Again, if RC turns out to be empty, then the integrated method adds the pair $(0,a)$)

In order to present the complexity of single methods, we characterize the graph G_L as follows Consider the subgraph of G_L induced by all single nodes b having a *distance* from a less than i_s We denote the number of nodes and arcs of this subgraph by n_s and m_s, respectively Moreover, we denote by $n_{\hat{s}}$ the number of all single nodes b such that b has a distance from a less than i_s and there is no directed path from b to any node with distance from a greater or equal to i_s Le $m_{\hat{s}}$ be the total number of arcs entering the above nodes Obviously, $n_s \geq n_{\hat{s}}$ and $m_s \geq m_{\hat{s}}$ For the magic graph G_L in Figure 2, we have $i_s = 2$, $n_s = 4$, $n_{\hat{s}} = 1$, $m_s = 3$, $m_{\hat{s}} = 1$

Thus we have the following proposition

PROPOSITION 5 *The single magic counting methods are correct and safe and their costs are the ones shown in Table 3 Furthermore,* $S_{IND} =_R S_{INT} =_R B$, $S_{INT} \leq_{A,C} S_{IND}$ *and* $S_{IND} \leq_{A,C} B$

PROOF It is easy to see that $RC_{-i} \cup RM = MS$ In addition, RC_{-i} contains only single nodes Hence, every node in RC_{-i} has only one index, thus, condition b) of Theorem 1 or 2 is satisfied Finally, if RC is not empty, then it contains the pair (o,a) since obviously $0 \leq i_s$ In the other case, we have added the above pair It follows that also condition c) of Theorem 2 holds, so the single methods are correct Let us now discuss their complexity If the magic graph is regular, RC_{-i} contains all nodes and the single methods coincides with the counting method Let us now suppose that the query is not regular It is easy to see that also in this case the reduced sets are constructed in $O(m_L)$ time Let us now consider the implementation of Step 2 The integrated single method works as the magic set method for all nodes in N_L that have a distance from the source node a greater or equal to i_s Thus, the method finds all possible paths from such nodes to all nodes in N_R Hence, the magic set part of the integrated method works in $\Theta((m_L - m_s) \times m_R)$ time On the other hand, the method works as the counting method for all nodes with distance from a less than i_s Thus, its cost is $\Theta(n_s \times m_R)$ Note that the cost of the Rule 3 (see the implementation of Step 2 for integrated methods), where the results of the magic set method is passed to the counting method, has been already included in the cost of the magic set part of the integrated method Using a similar argument, it is easy to see that the cost functions of the independent method for non-regular magic graphs, with or without cycles, are the ones shown in Table 3 Hence, all cost functions in Table 3 are correct and, therefore, the single methods are safe The cost function of the independent method is an upper bound for the cost function of the integrated method for non-regular graphs, since $m_{\check{s}} \geq m_{\hat{s}}$ The other relationships among cost functions can be easily derived from Table 3 and Table 2 □

MG	Independent	Integrated
Regular	$\Theta(m_L + n_L \times m_R)$	$\Theta(m_L + n_L \times m_R)$
Non-Regular	$\Theta(m_L + +(m_L - m_{\hat{s}}) \times m_R + n_{\hat{s}} \times m_R)$	$\Theta(m_L + +(m_L - m_{\check{s}}) \times m_R + n_{\check{s}} \times m_R)$

Tab 3 *Costs of the single magic counting methods*

Proposition 5 says that the single methods work better than the basic ones and that the integrated single method works better than the independent one We note that the integrated single method coincides with the magic counting method presented in [SZ1]

8. Multiple Magic Counting Methods

We now propose multiple magic counting methods that fully exploit single nodes by including all of them in the reduced counting set To this end, the second argument of the magic set predicate records whether this is the first occurrence of a node or the second This time, both first and second occurrences are used to generate other nodes in order to identify all multiple or recurring nodes (thus, we may need to use the same path twice)

```
begin
  $\overline{MS}(0, 1, a)$,
  $I = 0$,
  while $\overline{MS}(I, C, X_1)$, do
  begin
    $\overline{MS}(I+1, C, X_1)$ -
       $\overline{MS}(I, \hat{C}, X)$, $L(X, X_1)$, $not(\overline{MS}(\_, 2, X_1))$,
       if $\overline{MS}(\_, 1, X_1)$ then $C=2$ else $C=1$,
    $I = I+1$,
  end,
end
```

At the end of the above fixpoint computation, the multiple methods compute RC_{-1} as the set of all single nodes and RM as the set of all multiple/recurring nodes in the following way (recall that the magic set MS is also needed for independent methods)

$$MS(Y) \ \text{-} \ \overline{MS}(\_, 1, Y)$$

$$RM(Y) \ \text{-} \ \overline{MS}(\_, 2, Y)$$

$$RC(I, Y) \ \text{-} \ \overline{MS}(I, 1, Y), \ not(RM(Y))$$

If RC happens to be empty (i e , there are no single nodes), then the integrated method adds the pair $(0, a)$ In order to discuss the complexity of multiple magic counting methods, we denote by n_s the number of all simple nodes in G_L and by m_s the number of arcs in the subgraph of G_L induced by the simple nodes Besides, we denote by $n_{\hat{s}}$ the number of all simple nodes b in G_L such that there is no directed path from b to any multiple or recurring node Let $m_{\hat{s}}$ be the total number of arcs entering the above nodes It is easy to see that $n_s \geq n_{\hat{s}}$, $m_s \geq m_{\hat{s}}$, $n_s \geq n_{\check{s}}$, $m_s \geq m_{\check{s}}$, $n_{\hat{s}} \geq n_{\check{s}}$ and $m_{\hat{s}} \geq m_{\check{s}}$ For the magic graph in Figure 2, we see that $n_s = 6$, $n_{\hat{s}} = 2$, $m_s = 6$ and $m_{\hat{s}} = 3$

PROPOSITION 6 *The multiple magic counting methods are correct and safe and their costs are the ones shown in Table 4 Furthermore,* $M_{INT} \leq_{A,C} M_{IND}$, $M_{IND} \leq_{A,C} S_{IND}$, $M_{INT} \leq_{A,C} S_{INT}$ *and* $M_{IND} =_R M_{INT} =_R S_{INT} =_R S_{IND}$ □

MG	Independent	Integrated
Regular	$\Theta(m_L + n_L \times m_R)$	$\Theta(m_L + n_L \times m_R)$
Non-Regular	$\Theta(m_L + +(m_L - m_{\hat{s}}) \times m_R + n_{\hat{s}} \times m_R)$	$\Theta(m_L + +(m_L - m_s) \times m_R + n_s \times m_R)$

Tab 4 *Costs of multiple magic counting methods*

From Proposition 6, we infer that the independent (resp , integrated) multiple counting method works better than the independent (resp , integrated) single method Besides, the integrated multiple method works better than the independent one

9. Recurring Magic Counting Methods

In order to cope with cycles in the reduced set computation, we observe that, in a graph with K nodes, any path with length $>2 \times K - 1$, is cyclic Thus, we have the following algorithm, which also uses a set-cardinality function (an efficient operation that does not change our complexity bounds)

```
begin
    $\overline{MS}(0, 1, a)$,
    $I = 0$,
    $K = 1$,
    while $\overline{MS}(I, X_1), I < 2 \times K - 1$ do
    begin
        $\overline{MS}(I+1, X_1) - \overline{MS}(I, X), L(X, X_1)$,
        $K = cardinality(\overline{MS}(\_, Y))$,
        $I = I + 1$,
    end,
end
```

At the end of the above fixpoint computation, the recurring methods compute RC_-, as the set of all single and multiple nodes and RM as the set of all recurring nodes in the following way

$$MS(Y) - \overline{MS}(\_, Y)$$

$$RM(Y) - \overline{MS}(I, Y), I \geq K$$

$$RC(I, Y) - \overline{MS}(I, Y), not(RM(Y))$$

If RC happens to be empty (i e , all nodes are recurring) then the integrated method adds the pair $(0, a)$ to it

In order to perform the complexity analysis of recurring methods, we denote by n_m and m_m the number of multiple and single nodes and the number of arcs among such nodes Besides, we denote by $n_{\hat{m}}$ the number of all simple or multiple nodes b in G_L such that there is no directed path from b to any recurring node Let $m_{\hat{m}}$ be the total number of arcs entering the above nodes It is easy to see that $n_m \geq n_s$, $m_m \geq m_s$, $n_m \geq n_{\hat{m}}$, $m_m \geq m_{\hat{m}}$, $n_{\hat{m}} \geq n_{\hat{s}}$ and $m_{\hat{m}} \geq m_{\hat{s}}$ For the magic graph G_L in Figure 2, we see that $n_m = 8$, $n_m = 7$, $m_m = 9$ and $m_{\hat{m}} = 8$

PROPOSITION 7 *The recurring magic counting methods are correct and safe and their costs are the ones shown in Table 5 Furthermore,* $R_{INT} \leq_{A,C} R_{IND}$, $R_{IND} \lesssim_{A,C} M_{IND}$, $R_{INT} \lesssim_{A,C} M_{INT}$ *and* $R_{IND} =_R R_{INT} =_R M_{INT} =_R M_{IND}$ □

MG	Independent	Integrated
Regular	$\Theta(m_L + n_L \times m_R)$	$\Theta(m_L + n_L \times m_R)$
Acyclic	$\Theta(n_L \times m_L + n_L \times m_R)$	$\Theta(n_L \times m_L + n_L \times m_R)$
Cyclic	$\Theta(n_L \times m_L + +(m_L - m_{\hat{m}}) \times m_R + n_{\hat{m}} \times m_R)$	$\Theta(n_L \times m_L + +(m_L - m_m) \times m_R + n_m \times m_R)$

Tab 5 *Costs of recurring magic counting methods*

Proposition 7 states that again the integrated method works better than the corresponding independent method, and both methods work better than the counting method However, the reduced sets are not computed any longer in $O(m_L)$ time This means that the recurring methods work better than the corresponding multiple methods only on the average One could object that the implementation of Step 1 for recurring methods is a bit naive, and some other implementation could work better Indeed, we have a smarter implementation that computes the reduced set in $O(m_L + n_m \times m_m)$, thus the increase of complexity is only limited to the multiple nodes, whereas recurring nodes are detected in linear time (Due to the space constraints of this paper, this algorithm is not given here, but see [Tar] for a depth algorithm that detects strongly connected components in linear time) We point out that the cost component $n_m \times m_m$ is due to the fact that we need to associate with every multiple node all possible distances from the source node This means that we cannot expect the same tangible improvement in passing from multiple methods to recurring ones as we had in moving from basic to single methods or from single to multiple methods

10. Conclusion

We have presented a family of methods that combine the strengths of two well-known methods for implementing logic queries for databases, namely, the magic set method and the counting method We have given necessary and sufficient conditions for the new algorithms, called magic counting methods, to be correct and safe In addition, we have divided the family of magic counting methods into independent methods and integrated methods, according to whether the magic set part and the counting part run independently from each other or co-operate Within each family, four methods were introduced, these are the basic method, the single method, the multiple method and the recurring method Every method is identified by two coordinates, one distinguishing between basic methods, single methods, multiple methods and recurring ones, and the second coordinate establishing whether the method is independent or integrated

A detailed efficiency analysis of the methods was performed and it was found that that there exists a clear hierarchy among them In fact, if the first coordinate is fixed, then the integrated method works better than the independent ones, if the second coordinate is fixed, then a recurring method works better than a multiple one, a multiple one works better than a single one, and a single one works better than a basic one Finally, all magic counting methods are safe and work better than the magic set method, and they coincide with the counting method when the query is regular In Figure 3 we present this hierarchy in details The relationship

$\leq_q$ is denoted by a a solid arc labelled q, while the relationship $\lesssim_q$ is denoted by a dotted arc The integrated basic method and the the independent basic method have the same cost, thus, they are represented by a one node, *B* Since all magic counting methods have the same cost function for regular cases, the corresponding arcs are not included in Figure 3

These results were obtained for simple kinds of queries However, their extension to a larger class of queries, called canonical strongly linear in [SZ1], is reasonably simple Their extension to the completely general case is harder, but possible, and constitutes a topic for future research

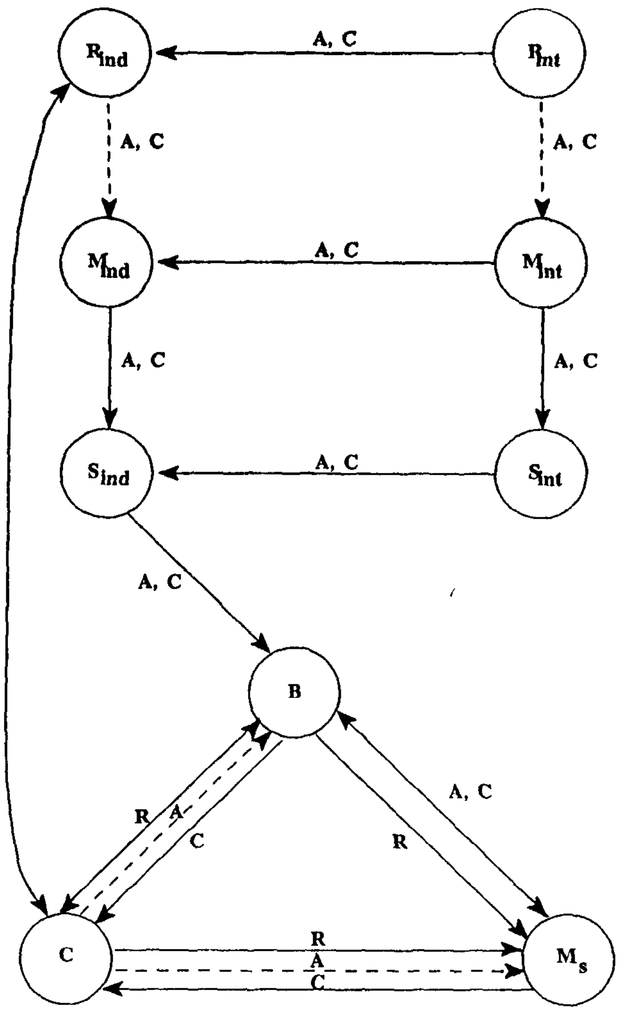

Fig 3 Efficiency Hierarchy Among Magic Counting Methods

References

[Ban] Bancilhon, F , "Naive evaluation of Recursively Defined Relations," Unpublished Manuscript, 1985

[BaR] Balbin Isaac, K Ramamohanarao, "A Differential Approach to Query Optimization in Recursive Deductive Databases," Journal of Logic Programming, to appear

[BMSU] Bancilhon, F , Maier, D , Sagiv, Y , Ullman, J D, "Magic sets and other strange ways to implement logic programs", *Proc 5th ACM SIGMOD-SIGACT Symp on Principles of Database Systems,* 1986, pp 1-15

[BR] Bancilhon, F , Ramakrishnan, R , "An amateur's introduction to recursive query processing strategies", *Proc ACM SIGMOD Conf ,* 1986, pp 16-52

[HN] Henschen, L J, Naqvi, S A, "On compiling queries in recursive first-order databases", *JACM 31,* 1, 1984, pp 47-85

[MK] McKay, D , Shapiro, S , "Using active connection graphs for reasoning with recursive rules", *Proc 7th IJCAI, 1981, pp 368-374*

[MPS] Marchetti-Spaccamela, A , Pelaggi, A , Saccà, D , "Worst-case complexity analysis of methods for logic query implementation", *Proc ACM SIGMOD-SIGACT Symp on Principles of Database Systems,* San Diego, Cal , 1987

[SZ1] Saccà, D , Zaniolo, C , "On the implementation of a simple class of logic queries for databases", *Proc 5th ACM SIGMOD-SIGACT Symp on Principles of Database Systems,* 1986, pp 16-23

[SZ2] Saccà, D , Zaniolo, C , "The generalized counting method for recursive queries, *Proc 1st International Conf on Database Theory,* Rome, 1986

[SZ3] Saccà, D , Zaniolo, C , "Implementation of Recursive Queries for a Data Language Based on Pure Horn Logic", *Procs Fourth Conference on Logic Programming,* Melburne, May 25-29, 1987

[Tar] Tarjan, R E , "Depth first search and linear graphs algorithms," *SIAM J Computing,* vol 1, no 2, 146-160

[TZ] Tsur, Shalom, and Zaniolo, Carlo "LDL A Logic-based Data Language," Proc 12th Int Conference on Very Large Data Bases, 1986

[Ul] Ullman, J D , "Implementation of logical query languages for databases", *TODS 10,* 3, 1985, pp 289-321

[VK] van Emden, M H , Kowalski, R , "The semantics of predicate logic as a programming language", *JACM 23,* 4, 1976, pp 733-742