One-Phase Magic-sets Transformation

(Extended Abstract)

Inderpal Singh Mumick
AT&T Bell Laboratories
mumick@research.att.com

Abstract

Magic-sets transformation is typically applied to a program in two phases – an adornment phase, and a magic-sets phase. A major drawback of the two-phase transformation is that join ordering, needed to do the adornment, must be done without knowledge of the actual predicates being pushed, and thereby without selectivity estimates and cardinality estimates. Consequently, two-phase implementations of the magic-sets transformation cannot do meaningful cost-based join optimization; and indeed they don't. A second drawback of the two-phase transformation is that when non-equijoin predicates, such as $X > Y$, are being pushed using condition (c) adornments, complex adornments must be used to represent different types of non-equijoin predicates.

In this paper we propose a one-phase transformation wherein the adornment and magic-sets phases are interleaved. The interleaving enables us to see the predicates on a rule before adorning the rule. Consequenly, (1) information about actual predicates is available when determining join-orders, enabling a cost-based join optimizer to consider the selectivities of the predicates, and (2) simple condition adornments can be used to represent all types of non-equijoin predicates.

1 Introduction

Magic-sets [BMSU86, BR91, Ram88, Mum91] is a query-rewrite optimization algorithm for recursive and nonrecursive queries written in Datalog or SQL. The magic-sets transformation has been implemented in database systems for optimizing recursive and nonrecursive queries: Coral [RSS92] implements magic templates [Ram88], Aditi [VRK+90], EDS [FF93], $\mathcal{LDL}$ [NT88], NAIL [MUVG86, MNS+87], and Glue-Nail [PDR91, DMP93] implement the supplementary magic-sets transformation [BR91], and Starburst [MP94] implements magic conditions [MFPR90a].

Performance experiments have shown the magic-sets transformation to be a good optimization technique, both for recursive and nonrecursive queries [BR88, MFPR90b]. However, magic-sets transformation has not found its way into commercial databases. A prime reason is the problem with determining join orders needed for the adornment phase of the magic-sets transformation. A second problem is the proliferation of adorned variants of each predicate when trying to push non-equijoin predicates (using the condition (c) adornment, or actual constraints as adornments [MPR94]). We address both these problems in this paper, and show that a simple rethinking of the manner in which adornments are determined can help in determining join orders and reducing the complexity of adornments. The rethinking leads to interleaving the adornment and magic-sets phases to obtain a one-phase magic-sets transformation.

Paper Outline: Section 2 illustrates that actual predicates are not available to the join optimizer when the magic-sets transformation is done in two-phases. We then show how a one-phase transformation can make actual predicates available. Estimates of cardinalities and selectivities can then be made while determining join-orders for purposes of adorning a rule. The one-phase algorithm is outlined in Section 3. Section 4 shows, through an example, that the condition (c) adornment can be used for all types of conditions if we use the one-phase magic-sets transformation, but not if we use the two-phase algorithm. Section 5 summarizes our contributions. The one-phase ground magic-sets transformation for pushing conditions is discussed in the Appendix Section A.

2 Join Orders

Magic-sets transformation is typically done in two phases. The given query is adorned in the first phase to indicate which arguments of a predicate are bound to a set of values, and which arguments are free. Full sips [BR91] are used during adornment, and a join order for the subgoals in a rule is needed to do the adornment. However, not much information is available for doing cost-based join ordering, and indeed the two-phase implementations either assume a join order identical to the given syntactic ordering of subgoals, or use a bound-is-easier heuristic. For example, consider the query Q on program P, where p, q, and s are idb predicates, and r, u, v, w, and t are edb predicates.

(Q): ?- $r(2, X)$ & $p(X, Y)$.

$(P1)$: $p(X, Y)$:– $q(X, Z)$ & $s(Z, Y)$.
$(P2)$: $q(X, Y)$:– $v(X, W, V, Y)$ & $w(Y, W, U, V)$.
$(P3)$: $s(X, Y)$:– $t(X, Y)$ & $u(X, Y)$.

In the first phase, the query is adorned as follows (edb predicates are not adorned):

(AQ): ?- $r(2, X)$ & $p^{bf}(X, Y)$.

$(A1)$: $p^{bf}(X, Y)$:– $q^{bf}(X, Z)$ & $s^{bf}(Z, Y)$.
$(A2)$: $q^{bf}(X, Y)$:– $v(X, W, V, Y)$ & $w(Y, W, U, V)$.
$(A3)$: $s^{bf}(X, Y)$:– $t(X, Y)$ & $u(X, Y)$.

The subgoal $p^{bf}(X, Y)$ in rule AQ is adorned bf since its first argument X is restricted by an equijoin predicate with the second argument of $r(2, X)$. Now, the rule $P1$ is adorned for the head adornment p^{bf}. A decision must be made about the join order between the subgoals in the rule $P1$. The only information available for determining the join orders is that the variable X is bound to some set of constants. Given this information, and a bound is easier heuristic, one may decide on the join order $q \bowtie s$. Consequently, rules $A1$, $A2$, and $A3$ will be adorned as shown.

Suppose the edb relations v, w, t, and u are such that (1) it is more optimal to use the join order $s \bowtie q$ in rule $P1$ whenever the binding predicate on X is not selective (the number of bound values is large). (2) it is more optimal to use the join order $q \bowtie s$ in rule $P1$ whenever the binding predicate on X is very selective (the number of bound values is small). However, if a cost optimizer is used to determine the join order in rule $P1$, no information about the binding predicate is available. Consequently, the optimizer has no way of choosing between the two possible join orders.

The second phase of a two phase magic-sets transformation leads to the program M:

(AQ): ?- $r(2, X)$ & $p^{bf}(X, Y)$.

$(M1)$: $p^{bf}(X, Y)$:– $m\_p^{bf}(X)$ & $q^{bf}(X, Z)$ & $s^{bf}(Z, Y)$.
$(M2)$: $q^{bf}(X, Y)$:– $m\_q^{bf}(X)$ & $v(X, W, V, Y)$ & $w(Y, W, U, V)$.
$(M3)$: $s^{bf}(X, Y)$:– $m\_s^{bf}(X)$ & $t(X, Y)$ & $u(X, Y)$.

$(M4)$: $m\_p^{bf}(X)$:– $r(2, X)$.
$(M5)$: $m\_q^{bf}(X)$:– $m\_p^{bf}(X)$.
$(M6)$: $m\_s^{bf}(Z)$:– $m\_p^{bf}(X)$ & $q^{bf}(X, Z)$.

Looking at the rule $M1$ in program M, we have access to the binding predicate on attribute X, and its selectivity estimate is given by the cardinality estimate of the magic set $m\_p^{bf}(X)$. The cardinality of $m\_p^{bf}(X)$ can be estimated by a cost optimizer from the statistics for edb relation r (rule $M4$).

Thus we find that the join order optimizer can be helped if it knows about the predicate that provides the binding represented by the b adornment. However, this predicate was not available until after the first phase.

Can we provide the actual binding predicate in the first phase itself, when we were trying to choose a join order for rule $P1$? We show that by interleaving the adornment and magic-sets phases into a one-phase transformation, we can have access to the binding predicate before adorning the rule $P1$.

The idea is to apply magic-sets to the query rule Q before adorning rule $P1$. Applying magic-sets to the query rule Q means generating the magic rule $M4$ for the magic predicate m_p^{bf}, and adding the magic predicate as a subgoal into the rule $P1$. We would thus get:

$(P1')$: $p^{bf}(X,Y)$:- $m\_p^{bf}(X)$ & $q(X,Z)$ & $s(Z,Y)$.
$(P2)$: $q(X,Y)$:- $v(X,W,V,Y)$ & $w(Y,W,U,V)$.
$(P3)$: $s(X,Y)$:- $t(X,Y)$ & $u(X,Y)$.

$(M4)$: $m\_p^{bf}(X)$:- $r(2,X)$.

Note that the subgoals in rule $P1'$ have not been adorned, because we have not determined the join order yet. Now, we will invoke a cost based optimizer to determine the join order in rule $P1$. As a hypothetical scenario, the optimizer estimates that the magic set m_p^{bf} has a cardinality of 100, and that $s(Z,Y)$ has a cardinality of 200 if evaluated as s^{ff}, and that $q(X,Z)$ has a cardinality of $10,000$ if evaluated as q^{bf} with a binding on X of size 100, and a cardinality of 250 if evaluated as q^{bb} with a binding on X of size 100 and a binding on Z of size 200. The optimizer then chooses the join order $s \bowtie q$.

Given this join order, the rule $P1'$ is adorned as shown in rule $M1$ below, and magic-sets is applied immediately to it. Applying magic-sets to rule $P1'$ means that the magic rule $M5$ is generated, and the rule $M2$ and $M3$ are written for adorned head predicates.

$(M1)$: $p^{bf}(X,Y)$:- $m\_p^{bf}(X)$ & $s^{ff}(Z,Y)$ & $q^{bb}(X,Z)$.
$(M2)$: $q^{bb}(X,Y)$:- $m\_q^{bb}(X,Y)$ & $v(X,W,V,Y)$ & $w(Y,W,U,V)$.
$(M3)$: $s^{ff}(X,Y)$:- & $t(X,Y)$ & $u(X,Y)$.

$(M4)$: $m\_p^{bf}(X)$:- $r(2,X)$.
$(M5)$: $m\_q^{bb}(X,Z)$:- $m\_p^{bf}(X)$ & $s^{ff}(Z,Y)$.

The optimizer would presumably prevent computation of the cross product in rule $M5$ by constructing the projections of the magic set on the two attributes.

2.1 Selection Predicates

By interleaving the adornment and magic-sets phases above, we were able to use the cardinality and consequent selectivity estimates for the join predicate between $r(2,X)$ and $p(X,Y)$. A similar affect is achieved when the binding is provided by a selection predicate such as $X = 1$. As an example the query Q':

(Q'): ?- $p^{bf}(1,Y)$ & $p^{bf}(2,Z)$.

on the program P given above generates the binding set $\{1,2\}$. If we apply magic sets to the query rule Q' before adorning the rule $P1'$ for p, we get:
$(P1')$: $p^{bf}(X,Y)$:- $m\_p^{bf}(X)$ & $q(X,Z)$ & $s(Z,Y)$.

$(M4a)$: $m\_p^{bf}(1)$.
$(M4b)$: $m\_p^{bf}(2)$.

A cost optimizer invoked at this stage can see that m_p^{bf} contains exactly the values 1 and 2. Knowing the values of the constants in the binding set can help determine good join orders and subsequent adornments, because the selectivities can be different for different constants. For instance, we may know that values 1 and 2 occur very commonly in the first attribute of edb relation v, so that we would not gain much from evaluating the rule in the order $m\_p^{bf} \bowtie q^{bf} \bowtie s^{bf}$. The optimizer may then choose the join order

$$m\_p^{bf} \bowtie s^{ff} \bowtie q^{bb}.$$

In relational databases, the actual predicates, and not just the fact that predicates exists, are often used to influence the join ordering. Further, in systems such as IBM DB2 [IBM92] the actual binding value may be used to select indices if there are spikes in the probability density function of attribute values. We can make the actual predicates available for join order determination while adorning a rule by interleaving the adornment and magic-sets phases. Thereby, existing join order estimation techniques can be used to the fullest to aid the magic-sets transformation.

3 One-Phase Magic-sets transformation

We will give only an outline of the one-phase magic-sets transformation algorithm. The algorithm works stratum by stratum, where stratum numbers are defined as follows:

Definition 3.1 Stratum Numbers: The *dependency graph* of a program has predicate names as nodes; a directed edge (p, q) is in the edge set if there is a rule headed by a q atom and having in its body a p literal. Construct a *reduced dependency graph* of the given program by collapsing every strongly connected component of the dependency graph to a single node. A reduced graph is guaranteed to be acyclic. A topological sort [AHU83] of the reduced dependency graph assigns a stratum number to each node. If a node represents a strongly connected component, all predicates in the strongly connected component are assigned the stratum number of the node. By convention, the base predicates (edb) are assigned stratum number = 0. □

The 1MST Algorithm: Given a program P and query Q, the 1MST algorithm transforms (P, Q) stratum by stratum, starting from the top stratum. Within each stratum, we apply one-phase magic-sets transformation to each rule of each predicate in the stratum. Applying the one-phase magic-sets transformation to rule r for an adorned predicate p^{α} involves:

- Determine a join order for subgoals of rule r. A cost optimizer may be used. If adornment α is not $ff\ldots f$, then a magic subgoal will be present in rule r and the exact binding predicate is available by looking at the definition of the magic predicate.

- If a new adorned subgoal in a lower stratum is generated, create adorned versions of all rules for the predicate. Add a magic subgoal into each rule. These rules will be processed later during processing of the lower stratum.

- If a new adorned subgoal in the same stratum is generated, create adorned versions of all rules for the predicate. Add a magic subgoal into each rule. These rules will be processed during processing of the current stratum.

- Generate a rule defining the magic predicate for each idb subgoal of rule r.

$\diamond$

Lemma 3.1 *Given a program P and query Q, let M be the program obtained by the 1MST algorithm. Then,*

1. *Program M is equivalent to program P on query Q.*

2. *Program M is identical to the program obtained by the two-phase magic-sets transformation for same choice of join orders for each rule.*

$\square$

When determining the join order for rule r in the first step of the 1MST algorithm, even though some rules defining the magic predicate are available, all rules for the magic predicate may not be available. However, for nonrecursive programs, 1MST guarantees that all magic rules would be available.

Theorem 3.1 *If program P is nonrecursive, all rules for the magic predicate m_p rule are available when determining the join order for a rule r of predicate p.* $\square$

Theorem 3.1 means that complete information is available to join optimizers in a nonrecursive system implementing magic-sets. The theorem is very important for commercial optimizers trying to implement magic-sets for nonrecursive queries.

For a recursive program P, even though all rules are not available, having some of the rules is advantageous. The recursive rules are evaluated during several iterations, and the magic rule available in the first step describes the magic set that would be computed before the first iteration.

4 Non-equijoin predicates and condition adornments

The magic-sets transformation has been applied to push non-equijoin predicates, such as $X > 5$ and $X > Y$. Such predicates are called conditions. In this paper we will consider the ground magic-sets transformation (GMST) [MFPR90a] for pushing conditions. GMST operates in two phases: In the first phase, condition (c) adornments are used to mark arguments that will be conditioned. In the second phase, the condition predicates are pushed down by constructing non-ground magic rules that are immediately ground into supplementary magic rules.

For example, consider the query Q:

(Q): ?- $w(U,V)$ & $X > U$ & $Y > V$ & $p(X,Y)$.

$(E1)$: $p(X,Y)$:- $q(X,Z)$ & $s(Z,Y)$ & $u(Z)$.

$(E2)$: $q(X,Y)$:- $v(X,Y,Z)$.

$(E3)$: $s(X,Y)$:- $v(Z,X,Y)$.

Using condition adornments to represent a condition on an argument of a predicate, GMST derives the adorned program:

(AQ): ?- $w(U,V)$ & $X > U$ & $Y > V$ & $p^{cc}(X,Y)$.

$(AE1)$: $p^{cc}(X,Y)$:- $q^{cf}(X,Z)$ & $s^{bc}(Z,Y)$ & $u(Z)$.

$(AE2)$: $q^{cf}(X,Y)$:- $v(X,Y,Z)$.

$(AE3)$: $s^{bc}(X,Y)$:- $v(Z,X,Y)$.

The adornment p^{cc} represents the fact that each argument of p is restricted by some condition. The subgoals in the adorned rule for p^{cc} are ordered with the knowledge that some conditions exist, but without knowing what the conditions are. The join order $q \bowtie s \bowtie u$ is used, so it is determined that (1) the first argument of q will have a condition, leading to the adornment q^{cf}, and (2) the first argument of s will be bound, and the second argument of s will have a condition, leading to the adornment s^{bc}.

Now consider an alternative query Q':

(Q'): ?- $X > Y$ & $p(X,Y)$.

The predicate $X > Y$ also conditions each argument of predicate p. However, if we generate the adornment p^{cc} for the p subgoal, then we cannot adorn the rule $E1$ for p as:

$(A'E1)$: $p^{cc}(X, Y)$:- $q^{cf}(X, Z)$ & $s^{bc}(Z, Y)$ & $u(Z)$.

since the condition $X > Y$ on p does not imply that the first argument of q will have a condition on it. With the join order $q \bowtie s \bowtie u$, the adornment q^{cf} is simply incorrect. This problem is discussed in detail in [MPR94], and is characterized by the *faithful adornment property*. Complex adornment schemes, where different adornment symbols are used for the two types of conditions, or the conditions themselves are used as the adornment, are proposed to resolve the difficulty.

The One-Phase Solution: However, if we simply interleave the adornment and magic-sets phases, we can determine the exact conditions on each rule by looking up the magic set. The exact conditions can then be used to do the join order and obtain the adornment, avoiding the problems faced when adornments are done without seeing the magic set.

Let's explain the benefit of interleaving through an example. Suppose we use the p^{cc} adornment to represent the condition $X > Y$ in query Q' as:

(AQ'): ?- $X > Y$ & $p^{cc}(X, Y)$.

(Step 1:) We would then apply GMST to rule AQ' to obtain the magic rule:

$(I1)$: $m\_p^{cc}(X, Y)$:- $X > Y$.

Rule $I1$ is unsafe, and is grounded by looking at the rule $E1$ where it would be used, deriving the supplementary rule $I2$:

$(I2)$: $sm\_e1(X, Y, Z)$:- $X > Y$ & $q(X, Z)$ & $s(Z, Y)$.

The following rule $N1$ then computes the p^{cc} predicate from the supplementary predicate.

$(N1)$: $p^{cc}(X, Y)$:- $sm\_e1(X, Y, Z)$ & $u(Z)$.

(Step 2:) Next we determine the join order (say $q \bowtie s$) and adorn the supplementary rule $I2$ to derive the adorned supplementary rule $N4$:

$(N4)$: $sm\_e1(X, Y, Z)$:- $q^{ff}(X, Z)$ & $X > Y$ & $s^{bc}(Z, Y)$.

Note that q is given the adornment *ff* since we (1) see the condition $X > Y$ and, (2) know that this condition is not strong enough to place a condition on the first argument of q. However, once q is evaluated, we get a binding for the first argument of s, and a condition on the second argument of s, obtaining the adornment bc.

(Step 3:) We now apply GMST to rule $N4$, obtaining the magic rule $I3$:

$(I3)$: $m\_s^{bc}(Z, Y) :- q^{ff}(X, Z)\ \&\ X > Y.$

Rule $I3$ is unsafe, and is grounded by looking at the rule $E3$ where it would be used, deriving the supplementary rule $N5$:

$(N5)$: $sm\_e3(X, Y) :- q^{ff}(W, X)\ \&\ W > Y\ \&\ \ \&\ v(Z, X, Y).$

The following rule $N3$ then computes the s^{bc} predicate from the supplementary predicate.

$(N3)$: $s^{bc}(X, Y) :- sm\_e3(X, Y).$

Rule $E2$ is simply transformed into the rule $N2$:

$(N2)$: $q^{ff}(X, Y) :- v(X, Y, Z).$

The final program consists of rules $N1 \ldots N5$. The one-phase ground magic-sets transformation algorithm is briefly discussed in the Appendix Section A.

5 Conclusions

We argue, through examples, for adorning and doing the magic-sets transformation in one phase. This takes away the need to do subgoal ordering based upon the information in the adornment symbols alone. Instead detailed information about the exact binding or condition predicates can be used, thereby

1. Enabling better join order determination.

2. Enabling simpler adornment patterns to push arbitrary conditions.

We discuss the one-phase magic-sets transformation algorithms for pushing just equality predicates (b adornments), and for pushing equality and non-equality predicates.

The one-phase algorithm makes it easier to implement the magic-sets transformation in commercial databases where join-order determination and proliferation of adornments are major considerations. For example, the Starburst system uses the one-phase algorithm [MP94].

References

[AHU83] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1983.

[BMSU86] Francois Bancilhon, David Maier, Yehoshua Sagiv, and Jeffrey D. Ullman. Magic sets and other strange ways to implement logic programs. In *Proceedings of the Fifth Symposium on Principles of Database Systems (PODS)*, pages 1–15, Cambridge, MA, March 1986.

[BR88] Francois Bancilhon and Raghu Ramakrishnan. Performance evaluation of data intensive logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 439–518, Washington D.C., 1988. Morgan Kaufmann.

[BR91] Catriel Beeri and Raghu Ramakrishnan. On the power of magic. *Journal of Logic Programming*, 10:255–300, 1991. Preliminary version appeared in the 6th ACM Symposium on Principles of Database Systems, 1987.

[DMP93] Marcia Derr, Shinichi Morishita, and Geoffrey Phipps. Design and implementation of the Glue-Nail database system. In *Proceedings of ACM SIGMOD 1993 International Conference on Management of Data*, Washington, DC, May 26-28 1993.

[FF93] Jérôme Fessy and Béatrice Finance. Early experience with recursion optimization in an extensible rewriter. In Maria E. Orlowska and Michael Papazoglou, editors, *Advances in Database Research: Proceedings of the Fourth Australian Database Conference (ADC)*, pages 262–273, Brisbane, Australia, February 1-2 1993. World Scientific Publishing Co.,PO Box 128, Farrer Road, Singapore 9128.

[IBM92] IBM. *IBM Database 2 Administration Guide*, March 1992.

[MFPR90a] Inderpal Singh Mumick, Sheldon J. Finkelstein, Hamid Pirahesh, and Raghu Ramakrishnan. Magic conditions. In *Proceedings of the Ninth Symposium on Principles of Database Systems (PODS)*, pages 314–330, Nashville, TN, April 2-4 1990.

[MFPR90b] Inderpal Singh Mumick, Sheldon J. Finkelstein, Hamid Pirahesh, and Raghu Ramakrishnan. Magic is relevant. In *Proceedings of ACM SIGMOD 1990 International Conference on Management of Data*, pages 247–258, Atlantic City, NJ, May 23-25 1990.

[MNS+87] Katherine A. Morris, Jeffrey F. Naughton, Yatin Saraiya, Jeffrey D. Ullman, and Allen Van Gelder. Yawn! (yet another window on nail!). In *Data Engineering 10:4*, pages 28–43, 1987.

[MP94] Inderpal Singh Mumick and Hamid Pirahesh. Implementation of magic in starburst. In *Proceedings of ACM SIGMOD 1994 International Conference on Management of Data*, Minneapolis, MN, May 25-27 1994.

[MPR94] Inderpal Singh Mumick, Hamid Pirahesh, and Raghu Ramakrishnan. Adornments in database programs. *To Appear in Computers and AI*, 1994. Preliminary version appeared in Proceedings of the International Workshop on Deductive Databases, International Conference on Logic Programming, 1993.

[Mum91] Inderpal Singh Mumick. *Query Optimization in Deductive and Relational Databases.* PhD thesis, Stanford University, Stanford, CA 94305, USA, December 1991. Technical Report No. STAN-CS-91-1400. Also available from University Microfilms International, 300 N. Zeeb Road, Ann Arbor, MI 48106. (313)761-4700.

[MUVG86] Katherine A. Morris, Jeffrey D. Ullman, and Allen Van Gelder. Design overview of the nail! system. In *Logic Programming: Proceedings of the Third International Conference, London*, pages 554–568, 1986.

[NT88] Shamim Naqvi and Shalom Tsur. *A Logic Language for Data and Knowledge Bases.* Computer Science Press, 1988.

[PDR91] Geoff Phipps, Marcia Derr, and Kenneth A. Ross. Glue-Nail: A deductive database system. In *Proceedings of ACM SIGMOD 1991 International Conference on Management of Data*, pages 308–317, Denver, CO, May 29-31 1991.

[Ram88] Raghu Ramakrishnan. Magic templates: A spellbinding approach to logic programs. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 140–159, Seattle, WA, August 1988. MIT Press.

[RSS92] Raghu Ramakrishnan, Divesh Srivastava, and S. Sudarshan. CORAL: Control, relations and logic. In *Proceedings of the Eighteenth International Conference on Very Large Databases (VLDB)*, pages 238–250, Vancouver, Canada, August 23-27 1992.

[VRK+90] Jayen Vaghani, Kotagiri Ramamohanarao, David Kemp, Zoltan Somogyi, and Peter Stuckey. The Aditi deductive database system. In *Workshop on Deductive Databases, North American Conference on Logic Programming (NACLP)*, 1990.

A The One-Phase Ground Magic-sets transformation

The core of the one-phase GMST algorithm is the following idea: Adorn a predicate p in a rule r, and then immediately, before adorning the rules for predicate p, create a rule for the magic set m_p from the usage of p in rule r. The actual bindings and conditions that caused the b and c adornment on predicate p are thus placed in this rule for the magic predicate m_p. Now, adorn the rules for predicate p, using the actual bindings and conditions in the magic predicate, rather than the abstract conditions represented by the c adornment.

The 1GMST Algorithm: Given a program P and query Q, the 1MST algorithm transforms (P, Q) stratum by stratum, starting from the top stratum. Within each stratum, we apply one-phase ground magic-sets transformation to each rule of each predicate in the stratum. Applying the one-phase magic-sets transformation to a stratum k involves:

- (A:) For each rule r for each adorned predicate p^α in stratum k:
  - Determine the join order.
  - Construct a supplementary predicate, and replace subgoals that will be used in the supplementary rule (this set of subgoals is called the *grounding set*) by the supplementary subgoal.
- (B:) Adorn the remaining subgoals in rule r, and create magic rules for the subgoals in this rule.
- (C:) Ground each magic rule for stratum k by the grounding set to create supplementary rules.

- (D:) Select join orders for supplementary rules of stratum k, adorn the supplementary rules, and create magic rules for the subgoals in the supplementary rules.

$\diamond$

The 1GMST restricts the subgoals that can move into the supplementary predicate (the *grounding set*) so that any magic rules created in the last step belong to a stratum lower than k. Further, we can guarantee that the magic rules for a predicate m_p can all be grounded by a unique grounding set for a rule for p. This is the crucial result that ehlps us prove the correctness of 1GMST.

Theorem A.1 *Given a rule r for predicate p, and an adornment string α with c adornments for predicate p, let $V(r, \alpha)$ be the conditioned variables in the head of rule r. Let $G(r, \alpha)$ be the grounding set. Let m be an unsafe magic rule for the magic predicate m_p^{α}, derived from a reference to the adorned predicate p^{α}. Then, regardless of the actual conditions in magic rule m, Step C of the 1GMST Algorithm grounds the rule m.* $\Box$