Magic Sets for Disjunctive Datalog Programs

Mario Alviano, Wolfgang Faber, Gianluigi Greco, Nicola Leone
{alviano,faber,greco,leone}@mat.unical.it
Department of Mathematics, University of Calabria
87036 Rende, Italy
November 2009

Abstract

In this paper, a new technique for the optimization of (partially) bound queries over disjunctive Datalog programs with stratified negation is presented. The technique exploits the propagation of query bindings and extends the Magic Set optimization technique (originally defined for non-disjunctive programs).

An important feature of disjunctive Datalog programs is nonmonotonicity, which calls for nondeterministic implementations, such as backtracking search. A distinguishing feature of the new method is that the optimization can be exploited during the nondeterministic phase. In particular, after some assumptions have been made, parts of the program may become irrelevant to a query under these assumptions. This allows for dynamic pruning of the search space. In contrast, the effect of the only previously defined Magic Set method for disjunctive Datalog (APM) is limited to the deterministic portion of the process. In this way the proposed method allows for exponential performance gains with respect to APM, which could also be confirmed empirically.

The correctness of the method is established and proved in a formal way, in which the strong relationship between magic sets and unfounded sets is exploited, which has not been studied in the literature before. This knowledge allows for extending the method and the correctness proof also to programs with stratified negation in a clean and founded way.

The proposed method has been implemented in the DLV system and various experiments on synthetic as well as on real-world data have been conducted. The experimental results on synthetic data confirm the utility of Magic Sets for disjunctive Datalog, and they highlight the computational gain that may be obtained by the new method with respect to the previously proposed Magic Set method APM for disjunctive Datalog programs. Further experiments on data taken from a real-life application show the benefits of the Magic Set method within an application scenario that has received considerable attention in recent years, the problem of answering user queries over possibly inconsistent databases originating from integration of autonomous sources.
1 Introduction

Disjunctive Datalog is a language that has been proposed for modelling incomplete data [36]. Together with a light version of negation, in this paper stratified negation, this language can in fact express any query of the complexity class $\Sigma^P_2$ [19], under the stable model semantics. For this reason, it is not surprising that disjunctive Datalog has found several practical applications, also encouraged by the availability of some efficient inference engines, such as DLV [33], GuT [29], Cmodels [35], or ClaspD [18]. As a matter of fact, these systems are continuously enhanced to support novel optimization strategies, enabling them to be effective over increasingly larger application domains. In this paper, we contribute to this development by providing a novel optimization technique, inspired by deductive database optimization techniques, in particular the Magic Set method [41, 4, 7].

The goal of the original Magic Set method (defined for non-disjunctive Datalog programs) is to exploit the presence of constants in a query for restricting the possible search space by considering only a subset of a hypothetic program instantiation, which is sufficient to answer the query in question. The magic sets are extensions of predicates that make this restriction explicit. Extending these ideas to disjunctive Datalog faces a major challenge: While non-disjunctive Datalog programs are deterministic, which in terms of the stable model semantics means that any non-disjunctive Datalog program has exactly one stable model, disjunctive Datalog programs are nondeterministic in the sense that they may have multiple stable models. Of course the main goal is still isolating a subset of a hypothetic program instantiation, upon which the considered query will be evaluated in an equivalent way.

There are two basic possibilities how this nondeterminism can be dealt with in the context of Magic Sets: The first is to consider static magic sets, in the sense that the definition of the magic sets is still deterministic, and therefore the extension of the magic set predicates is equal in each stable model. This static behaviour is automatic for magic sets of non-disjunctive Datalog programs. The second possibility is to allow dynamic magic sets, which also allow for nondeterministic definitions of magic sets. This means that the extension of the magic set predicates may differ in various stable models, and thus can be viewed as being specialized for different stable models.

While the nature of dynamic magic sets intuitively seems to be more fitting for disjunctive Datalog than static magic sets, considering the architecture of modern reasoning systems for disjunctive Datalog substantiates this intuition: These systems work in two phases, which may be considered as a deterministic (grounding) and a non-deterministic (model search) part. The interface between these two is by means of a ground program, which is produced by the deterministic phase. Static magic sets will almost exclusively have an impact on the grounding phase, while dynamic magic sets also have the possibility to influence the model search phase. In particular, some assumptions made during the model search may render parts of the program irrelevant to the query, which may be captured by dynamic magic sets, but not (or only under very specific
circumstances) by static magic sets.

In the literature, there is only one previous attempt for defining a Magic Set method for disjunctive Datalog, reported in [25, 26], which will be referred to as Auxiliary Predicates Method (APM) in this work. The basic idea of APM is that bindings need to be propagated not only from rule heads to rule bodies (as in traditional Magic Sets), but also from a head predicate to other head predicates. In addition to producing definitions for the predicates defining magic sets, the method also introduces additional auxiliary predicates called collecting predicates. These collecting predicates however have a peculiar effect: Their use keeps the magic sets static. Indeed, both magic and collecting predicates are guaranteed to have deterministic definitions, which implies that disjunctive Datalog systems can exploit the magic sets only during the grounding phase. Most systems will actually produce a ground program which does contain neither magic nor collecting predicates.

In this article, we propose a dynamic Magic Set method for disjunctive Datalog with stratified negation under the stable model semantics, provide an implementation of it in the system DLV, and report on an extensive experimental evaluation. In more detail, the contributions are:

- We present a dynamic Magic Set method for disjunctive Datalog programs with stratified negation, referred to as Disjunctive Magic Sets (DMS). Different from the previously proposed static method APM, existing systems can exploit the information provided by the magic sets also during their nondeterministic model search phase. This feature allows for potentially exponential performance gains with respect to the previously proposed static method.

- We formally establish the correctness of DMS. In particular, we prove that the program obtained by the transformation DMS is query-equivalent to the original program. This result holds for both brave and cautious reasoning.

- We highlight a strong relationship between magic sets and unfounded sets, which characterize stable models. We can show that the atoms which are relevant for answering a query are either true or form an unfounded set, which eventually allows us to prove the query-equivalence results.

- Our results hold for a disjunctive Datalog language with stratified negation under the stable model semantics. In the literature, several works deal with non-disjunctive Datalog with stratified negation under the well-founded or the perfect model semantics, which are special cases of our language. For the static method APM an extension to disjunctive Datalog with stratified negation is briefly described in [26], which however works only under particular restrictions.

- We have implemented a DMS optimization module inside the DLV system [33]. In this way, we could exploit the internal data-structures of the DLV system and embed DMS in the core of DLV. As a result, the technique
is completely transparent to the end user. The system is available at http://www.dlvsystem.com/magic/.

- We have conducted extensive experiments on synthetic domains that highlight the potential of DMS. We have compared the performance of the DLV system without magic set optimization, with APM, and with DMS. The results show that in many cases the magic set methods yield a significant performance benefit. Moreover, we can show that the dynamic DMS can yield drastically better performance than the static APM. Importantly, in cases in which DMS cannot be beneficial (if all or most of the instantiated program is relevant for answering a query), the overhead incurred is very light.

- We also report on experiments which evaluate the impact of DMS on an industrial application scenario on real-world data. The application involves data integration and builds on several results in the literature (for example [3, 24, 5, 14, 11, 15]), which transform the problem of query answering over inconsistent databases (in this context stemming from integrating autonomous data sources) into query answering over disjunctive Datalog programs. By leveraging these results, DMS can be viewed as a query optimization method for inconsistent databases or for data integration systems. The results show that DMS can yield significant performance gains for queries of this application.

Organization.
The main body of this article is organized as follows. In Section 2, preliminaries on disjunctive Datalog and on the Magic Set method for positive Datalog queries are introduced. Subsequently, in Section 3 the extension DMS for the case of disjunctive Datalog programs, and we show its correctness. In Section 4 we discuss the implementation and integration of the Magic Set method within the DLV system. Experimental results on synthetic benchmarks are reported in Section 5, while the application to data integration and its experimental evaluation is discussed in Section 6. Finally, in Section 7 we draw our conclusions.

2 Preliminaries

In this section, (disjunctive) Datalog programs with (stratified) negation are briefly described, and the standard Magic Set method is presented together with the notion of sideways information passing strategy (SIPS) for Datalog rules.

2.1 Disjunctive Datalog Programs with stratified negation

In this paper, we adopt the standard Datalog name convention: Alphanumeric strings starting with a lowercase character are predicate or constant symbols, while alphanumeric strings starting with an uppercase character are variable
symbols; moreover, we allow the use of positive integer constant symbols. Each predicate symbol is associated with a non-negative integer, referred to as its \textit{arity}. An \textit{atom} $p(\bar{t})$ is composed of a predicate symbol $p$ and a list $\bar{t} = t_1, \ldots, t_k$ of terms, each of which is either a constant or a variable. A \textit{literal} is an atom $p(\bar{t})$ or a negated atom $\neg p(\bar{t})$; in the first case the literal is \textit{positive}, while in the second it is \textit{negative}.

A \textit{disjunctive Datalog rule with negation} (short: Datalog$^{\vee, \neg}$ rule) $r$ is of the form

$$p_1(\bar{t}_1) \lor \cdots \lor p_n(\bar{t}_n) := q_1(\bar{s}_1), \ldots, q_m(\bar{s}_m),$$

$$\neg q_{j+1}(\bar{s}_{j+1}), \ldots, \neg q_n(\bar{s}_n).$$

where $p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n), q_1(\bar{s}_1), \ldots, q_n(\bar{s}_n)$ are atoms and $n \geq 1, m \geq j \geq 0$. The disjunction $p_1(\bar{t}_1) \lor \cdots \lor p_n(\bar{t}_n)$ is the \textit{head} of $r$, while the conjunction $q_1(\bar{s}_1), \ldots, q_j(\bar{s}_j), \neg q_{j+1}(\bar{s}_{j+1}), \ldots, \neg q_n(\bar{s}_n)$ is the \textit{body} of $r$. Moreover, $H(r)$ denotes the set of head atoms, while $B(r)$ denotes the set of body literals. We also use $B^+(r)$ and $B^-(r)$ for denoting the set of atoms appearing in positive and negative body literals, respectively. If $r$ is disjunction-free, that is $n = 1$, and negation-free, that is $B^-(r)$ is empty, then we say that $r$ is a Datalog rule; in addition, if $B^+(r)$ is empty, then we say that $r$ is a \textit{fact}. A \textit{disjunctive Datalog program} $\mathcal{P}$ is a finite set of rules; if all the rules in it are disjunction-and negation-free, then $\mathcal{P}$ is a (standard) Datalog program.

Given a Datalog$^{\vee, \neg}$ program $\mathcal{P}$, a predicate belongs to the \textit{Intensional Database} (IDB) if it is either in the head of a rule with non-empty body, or in the head of a disjunctive rule; otherwise, it belongs to the \textit{Extensional Database} (EDB). The set of rules having some IDB predicate in head is denoted by $IDB(\mathcal{P})$, while $EDB(\mathcal{P})$ denotes the remaining rules, that is, $EDB(\mathcal{P}) = \mathcal{P} \setminus IDB(\mathcal{P})$.

The set of all constants appearing in a program $\mathcal{P}$ is the \textit{universe} of $\mathcal{P}$ and is denoted by $U_\mathcal{P}$, while the set of ground atoms constructible from predicates in $\mathcal{P}$ with constants in $U_\mathcal{P}$ is the \textit{base} of $\mathcal{P}$, denoted by $B_\mathcal{P}$. We call an atom (rule, or program) \textit{ground} if it does not contain any variable. A ground atom $p(\bar{t})$ (resp. ground rule $r_\mathcal{P}$) is an instance of an atom $p(\bar{t}')$ (resp. rule $r_\mathcal{P}$) if there is a substitution $\vartheta$ from the variables in $p(\bar{t}')$ (resp. in $r_\mathcal{P}$) to $U_\mathcal{P}$ such that $p(\bar{t}) = p(\bar{t}')\vartheta$ (resp. $r_\mathcal{P} = r_\mathcal{P}\vartheta$). Given a program $\mathcal{P}$, $Ground(\mathcal{P})$ denotes the set of all the instances of the rules in $\mathcal{P}$.

Given an atom $p(\bar{t})$ and a set of ground atoms $\mathcal{A}$, by $\mathcal{A}|_{p(\bar{t})}$ we denote the set of ground instances of $p(\bar{t})$ belonging to $\mathcal{A}$. For example, $B_\mathcal{P}|_{p(\bar{t})}$ is the set of all the ground atoms obtained by applying to $p(\bar{t})$ all the possible substitutions from the variables in $p(\bar{t})$ to $U_\mathcal{P}$, that is, the set of all the instances of $p(\bar{t})$. Abusing of notation, if $B$ is a set of atoms, by $\mathcal{A}|_{B}$ we denote the union of $\mathcal{A}|_{p(\bar{t})}$, for each $p(\bar{t}) \in B$.

A desirable property of Datalog$^{\vee, \neg}$ programs is \textit{ safet y}. A Datalog$^{\vee, \neg}$ rule $r$ is safe if each variable appearing in $r$ appears in at least one atom of $B^+(r)$. A Datalog$^{\vee, \neg}$ program is safe if all its rules are safe. Moreover, programs without recursion over negated literals constitute an interesting class of Datalog$^{\vee, \neg}$ programs. Without going into detail, an atom $p(\bar{t})$ in the head of a rule $r$ depends
on all the atoms q(\bar{s}) in the body of r; if q(\bar{s}) belongs to B^+(r), p(\bar{\xi}) depends on q(\bar{s}) positively, otherwise negatively. A program has recursion over negation if a cycle of dependencies with at least one negative dependency exists. If a program has no recursion over negation, then the program is stratified (short: Datalog^{\vee,-\wedge}). In this work only safe programs without recursion over negation are considered.

An interpretation for a program \mathcal{P} is a subset I of B_\mathcal{P}. A positive ground literal p(\bar{\xi}) is true w.r.t. an interpretation I if p(\bar{\xi}) \in I; otherwise, it is false. A negative ground literal \neg p(\bar{\xi}) is true w.r.t. I if and only if p(\bar{\xi}) is false w.r.t. I. The body of a ground rule r is true w.r.t. I if and only if all the body literals of r are true w.r.t. I, that is, if and only if B^+(r) \subseteq I and B^-(r) \cap I = \emptyset. An interpretation I satisfies a ground rule r \in Ground(\mathcal{P}) if at least one atom in H(r) is true w.r.t. I whenever the body of r is true w.r.t. I. An interpretation I is a model of a Datalog^{\vee,-\wedge} program \mathcal{P} if I satisfies all the rules in Ground(\mathcal{P}). Since an interpretation is a set of atoms, if I is an interpretation for a program \mathcal{P}, and \mathcal{P}' is another program, then by I\lceil_{\alpha_{\mathcal{P}}}, we denote the restriction of I to the symbols in \mathcal{P}'.

Given an interpretation I for a program \mathcal{P}, the reduct of \mathcal{P} w.r.t. I, denoted Ground(\mathcal{P})^I, is obtained by deleting from Ground(\mathcal{P}) all the rules \tau_q with B^-(\tau_q) \cap I = \emptyset, and then by removing all the negative literals from the remaining rules.

The semantics of a Datalog^{\vee,-\wedge} program \mathcal{P} is given by the set SM(\mathcal{P}) of the stable models of \mathcal{P}, where an interpretation M is a stable model for \mathcal{P} if and only if M is a subset-minimal model of Ground(\mathcal{P})^M. For a Datalog program, it is well-known that there is exactly one stable model, also in presence of stratified negation, while in general, for a Datalog^{\vee,-\wedge} program \mathcal{P}, |SM(\mathcal{P})| \geq 1 holds (Datalog^{\vee,-\wedge} programs, instead, can have no stable model).

Given a ground atom p(\bar{\xi}) and a Datalog^{\vee,-\wedge} program \mathcal{P}, p(\bar{\xi}) is a cautious (or certain) consequence of \mathcal{P}, denoted by \mathcal{P} \models_c p(\bar{\xi}), if p(\bar{\xi}) \in M for each M \in SM(\mathcal{P}); p(\bar{\xi}) is a brave (or possible) consequence of \mathcal{P}, denoted by \mathcal{P} \models_b p(\bar{\xi}), if p(\bar{\xi}) \in M for some M \in SM(\mathcal{P}). Note that brave and cautious consequences coincide for Datalog programs, as these programs have a unique stable model.

Given a query Q = g(\bar{\xi})? (an atom), Ans_c(Q, \mathcal{P}) denotes the set of all the substitutions \theta for the variables of g(\bar{\xi}) such that \mathcal{P} \models_c g(\bar{\xi})\theta, while Ans_b(Q, \mathcal{P}) denotes the set of substitutions \theta for the variables of g(\bar{\xi}) such that \mathcal{P} \models_b g(\bar{\xi})\theta.

Let \mathcal{P} and \mathcal{P}' be two Datalog^{\vee,-\wedge} programs and \mathcal{Q} a query. Then \mathcal{P} and \mathcal{P}' are brave-equivalent w.r.t. \mathcal{Q}, denoted by \mathcal{P} \equiv_b \mathcal{P}', if Ans_b(Q, \mathcal{P} \cup \mathcal{F}) = Ans_b(Q, \mathcal{P}' \cup \mathcal{F}) is guaranteed for each set of facts \mathcal{F} defined over the EDB predicates of \mathcal{P} and \mathcal{P}'; similarly, \mathcal{P} and \mathcal{P}' are cautious-equivalent w.r.t. \mathcal{Q}, denoted by \mathcal{P} \equiv_c \mathcal{P}', if Ans_c(Q, \mathcal{P} \cup \mathcal{F}) = Ans_c(Q, \mathcal{P}' \cup \mathcal{F}) is guaranteed for each set of facts \mathcal{F} defined over the EDB predicates of \mathcal{P} and \mathcal{P}'

2.2 Sideways Information Passing for Datalog rules

The Magic Set method aims to simulate a top-down evaluation of a query \mathcal{Q}, like for instance the one adopted by Prolog. According to this kind of evaluation,
all the rules \( r \) such that \( p(\overline{t}) \in H(r) \) and \( H(r)\vartheta = \mathcal{Q} \) (for some substitution \( \vartheta \) for all the variables of \( r \)) are considered in a first step. Then the atoms in \( B^+(r)\vartheta \) are taken as subqueries (we recall that standard Datalog rules have empty negative body), and the procedure is iterated. Note that, according to this process, if a (sub)query has some argument that is bound to a constant value, then this information is “passed” to the atoms in the body. Moreover, the body is considered to be processed in a certain sequence, and processing a body atom may bind some of its arguments for subsequently considered body atoms, thus “generating” and “passing” bindings within the body. Whenever a body atom is processed, each of its argument is therefore considered to be either bound or free. We illustrate this mechanism by means of an example.

**Example 2.1** Let \( \text{path}(1,5) \) be a query for a program having the following inference rules:

\[
\begin{align*}
  r_1 : & \quad \text{path}(X,Y) : \sim \text{edge}(X,Y). \\
  r_2 : & \quad \text{path}(X,Y) : \sim \text{edge}(X,Z), \text{path}(Z,Y). \\
\end{align*}
\]

Since this is a Datalog program, brave and cautious consequences coincide. Moreover, let \( \mathcal{F}_1 = \{ \text{edge}(1,3), \text{edge}(2,4), \text{edge}(3,5) \} \) be the EDB of the program. A top-down evaluation scheme considers \( r_1 \) and \( r_2 \) with \( X \) and \( Y \) bound to 1 and 5, respectively. In particular, when considering \( r_1 \), the information about the binding of the two variables is passed to \( \text{edge}(X,Y) \), which is indeed the only query atom occurring in \( r_1 \). Then the evaluation fails since \( \text{edge}(1,5) \) does not occur in \( \mathcal{F}_1 \).

Instead, when considering \( r_2 \), the binding information can be passed either to \( \text{path}(Z,Y) \) or to \( \text{edge}(X,Z) \). Suppose that atoms are evaluated according to their ordering in the rule (from left to right); then \( \text{edge}(X,Z) \) is considered before \( \text{path}(Z,Y) \). In particular, \( \mathcal{F}_1 \) contains the atom \( \text{edge}(1,3) \), which leads us to map \( Z \) to 3. Eventually, this inferred binding information might be propagated to the remaining body atom \( \text{path}(Z,Y) \), which hence becomes \( \text{path}(3,5) \).

The process has now to be repeated by looking for an answer to \( \text{path}(3,5) \). Again, rule \( r_1 \) can be considered, from which we conclude that this query is true since \( \text{edge}(3,5) \) occurs in \( \mathcal{F}_1 \). Thus, \( \text{path}(1,5) \) holds as well due to \( r_2 \).

Note that in the example above we have two degrees of freedom in the specification of the top-down evaluation scheme. The first one concerns which ordering is used for processing the body atoms. While Prolog systems are usually required to follow the ordering in which the program is written, Datalog has a purely declarative semantics which is independent of the body ordering, allowing for an arbitrary ordering to be adopted. The second degree of freedom is slightly more subtle, and concerns the selection of the terms to be considered bound to constants from previous evaluations. Indeed, while we have considered the propagation of all the binding information that originates from previously processed body atoms, it is in general possible to restrict the top-down evaluation to partially propagate this information. For instance, one may desire to propagate only information generated from the evaluation of EDB predicates, or even just the information that is passed on via the head atom.
The specific propagation strategy adopted in the top-down evaluation scheme is called *sideways information passing strategy* (SIPS), which is just a way of formalizing a partial ordering over the atoms of each rule together with the specification of how the bindings originated and propagate [7, 26]. To formalize this concept, in what follows, for each IDB atom \( p(\bar{x}) \), we shall denote its associated binding information (originated in a certain step of the top-down evaluation) by means of a string \( \alpha \) built over the letters \( b \) and \( f \), denoting “bound” and “free”, respectively, for each argument of \( p(\bar{x}) \).

**Definition 2.2 (SIPS for Datalog rules)** A SIPS for a Datalog rule \( r \) w.r.t. a binding \( \alpha \) for the atom \( p(\bar{x}) \in H(r) \) is a pair \((\prec^b_r, f^b_r)\), where:

1. \( \prec^b_r \) is a strict partial order over the atoms in \( H(r) \cup B^+(r) \), such that \( p(\bar{x}) \prec^b_r q(\bar{z}) \), for all atoms \( q(\bar{z}) \in B^+(r) \); and,

2. \( f^b_r \) is a function assigning to each atom \( q(\bar{z}) \in H(r) \cup B^+(r) \) a subset of the variables in \( \bar{z} \)—intuitively, those made bound when processing \( q(\bar{z}) \).

**Example 2.3** The SIPS we have adopted in Example 2.1 for \( r_1 \) w.r.t. the binding \( b \) (originating from the query \texttt{path(1,5)}) can be formalized as the pair \((\prec^b_{r_1}, f^b_{r_1})\), where:

\[
\text{path}(X, Y) \prec^b_{r_1} \text{edge}(X, Y), \text{ and } f^b_{r_1}(\text{path}(X, Y)) = f^b_{r_1}(\text{edge}(X, Y)) = \{X, Y\}.
\]

Instead, the SIPS we have adopted for \( r_2 \) w.r.t. the binding \( b \) can be formalized as the pair \((\prec^b_{r_2}, f^b_{r_2})\) where: \( \text{path}(X, Y) \prec^b_{r_2} \text{edge}(X, Z) \prec^b_{r_2} \text{path}(Z, Y) \),

\[
f^b_{r_2}(\text{path}(X, Y)) = \{X, Y\}, f^b_{r_2}(\text{edge}(X, Z)) = \{X, Z\}, f^b_{r_2}(\text{path}(Z, Y)) = \{Z, Y\}.
\]

Note that a SIPS \((\prec^b_r, f^b_r)\) for \( r_2 \) where binding information is only generated from EDB predicates is such that: \( f^b_{r_2}(\text{path}(X, Y)) = \{X, Y\}, \) \( f^b_{r_2}(\text{edge}(X, Z)) = \{X, Z\} \). Also, note that both \( \prec^b_r \) and \( f^b_r \) are total orders.

All the algorithms and techniques we shall develop in this paper are orthogonal w.r.t. the underlying SIPSEs to be used in the top-down evaluation. Thus, in Section 2.3, we shall assume that Datalog programs are provided in input together with some arbitrarily defined SIPS \((\prec^b_r, f^b_r)\), for each rule \( r \) and for each possible adornment \( \alpha \) for the head atom in \( H(r) \).

### 2.3 Magic Sets for Datalog Programs

The Magic Set method is a strategy for simulating the top-down evaluation of a query by modifying the original program by means of additional rules, which narrow the computation to what is relevant for answering the query. We next provide a brief and informal description of the Magic Set rewriting technique. The reader is referred to [41] for a detailed presentation.

The method is structured in four main phases, which are informally illustrated below by means of Example 2.1.
(1) Adornment. The key idea is to materialize the binding information for IDB predicates that would be propagated during a top-down computation. In particular, the fact that an IDB predicate \( p(\bar{t}) \) is associated with a binding information \( \alpha \) (i.e., a string over the letters \( b \) and \( f \), one for each term in \( \bar{t} \)) is denoted by the atom obtained adorning the predicate symbol with the binding at hand, that is, by \( p^\alpha(\bar{t}) \). In what follows, the predicate \( p^\alpha \) is said an adorned predicate.

First, adornments are created for query predicates so that an argument occurring in the query is adorned with the letter \( b \) if it is a constant, or with the letter \( f \) if it is a variable. For instance, the adorned version of the query atom \( \text{path}(1,5) \) is \( \text{path}^{bb}(1,5) \), which gives rise to the adorned predicate \( \text{path}^{bb} \).

Each adorned predicate is eventually used to propagate its information into the body of the rules defining it according to a SIPS, thereby simulating a top-down evaluation. In particular, assume that the binding \( \alpha \) has to be propagated into the rule \( r \) whose head is \( p(\bar{t}) \). Then the associated SIPS \( (\prec^\alpha_r, f^\alpha_r) \) determines which variables will be bound in the evaluation of the various body atoms. Indeed, a variable \( X \) of an atom \( q(\bar{s}) \) in \( r \) is bound if and only if either

1. \( X \in f^\alpha_r(q(\bar{s})) \) with \( q(\bar{s}) = p(\bar{t}) \); or,
2. \( X \in f^\alpha_r(b(\bar{z})) \) for an atom \( b(\bar{z}) \in B^+(r) \) such that \( b(\bar{z}) \prec^\alpha_r q(\bar{s}) \) holds.

Adorning a rule \( r \) w.r.t. an adorned predicate \( p^\alpha \) means propagating the binding information \( \alpha \), starting from the head predicate \( p(\bar{t}) \in H(r) \), thereby creating a novel adorned rule where all the IDB predicates in \( r \) are substituted by the adorned predicates originating from the binding according to (1) and (2).

Example 2.4 Adorning the query \( \text{path}(1,5) \)? generates \( \text{path}^{bb}(1,5) \). Then, propagating the binding information \( bb \) into the rule \( r_1 \), i.e., when adorning \( r_1 \) with \( p^\alpha \), produces the following adorned rule (recall here that adornments apply only to IDB predicates, whereas \( \text{edge} \) is an EDB predicate):

\[
r_1^a : \quad \text{path}^{bb}(X, Y) :- \text{edge}(X, Y).
\]

Instead, when propagating \( bb \) into the rule \( r_2 \) according to the SIPS \( (\prec^{bb}_{r_2}, f^{bb}_{r_2}) \) defined in Example 2.3, we obtain the following adorned rule:

\[
r_2^a : \quad \text{path}^{bb}(X, Y) :- \text{edge}(X, Z), \text{path}^{bb}(Z, Y).
\]

While adorning rules, novel binding information in the form of yet unseen adorned predicates may be generated, which should be used for adorning other rules. In fact, the adornment step is repeated until all bindings have been processed, yielding the adorned program, which is the set of all adorned rules created during the computation. For instance, in the above example, the adorned program just consists of \( r_1^a \) and \( r_2^a \) for no adorned predicate different from \( p^{bb} \) is generated.
Example 2.6 In our running example, the following modified rules are generated:

Example 2.5 In our running example, only one magic rule is generated, \( r^*_2 : \) magic\_path\_bb\((Z, Y) : magic\_path\_bb\((X, Y), edge(X, Z)) \).

In fact, the adorned rule \( r^*_2 \) does not produce any magic rule, since there is no adorned predicate in \( B^+(r^*_2) \).

Example 2.7 The complete rewriting of our running example is as follows:\footnote{The Magic Set rewriting of a program \( P \) affects only \( IDB(P) \), so we usually omit \( EDB(P) \) in examples.}
magic_path^{bb}(1, 5),
path(X, Y) :- path^{bb}(X, Y).

r_2^* : magic_path^{bb}(Z, Y) :- magic_path^{bb}(X, Y), edge(X, Z).

r_1^* : path^{bb}(X, Y) :- magic_path^{bb}(X, Y), edge(X, Y).

r_2^* : path^{bb}(X, Y) :- magic_path^{bb}(X, Y), edge(X, Z), path^{bb}(Z, Y).

In this rewriting, magic_path^{bb}(X, Y) represents a potential sub-path of the paths from 1 to 5. Therefore, when answering the query, only these sub-paths will be actually considered in the bottom-up computation. One can check that this rewriting is in fact equivalent to the original program w.r.t. the query path(1, 5)?.

\[ \]

3 Magic Set Method for Datalog^∨,¬s Programs

In this section we present the Disjunctive Magic Set algorithm (DMS) for the optimization of disjunctive programs with stratified negation. Before discussing the details of the algorithm, we informally present the main ideas that have been exploited for enabling the Magic Set method to work on disjunctive programs (without negation).

3.1 Overview of Binding Propagation in Datalog^∨ Programs

As first observed in [26], while in non-disjunctive programs bindings are propagated only head-to-body, a Magic Set transformation for disjunctive programs has to propagate bindings also head-to-head in order to preserve soundness. Roughly, suppose that a predicate \( p \) is relevant for the query, and a disjunctive rule \( r \) contains \( p(X) \) in the head. Then, besides propagating the binding from \( p(X) \) to the body of \( r \) (as in the non-disjunctive case), the binding must also be propagated from \( p(X) \) to the other head atoms of \( r \). The reason is that \( r \) can yield support to the truth of \( p(X) \) only if all other head atoms are false, which is due to the implicit minimality criterion in the semantics.

Consider, for instance, a Datalog^∨ program \( P \) consisting of the rule \( p(X) \lor q(Y) :- a(X, Y), b(X). \), and the query \( p(1) \)?. Even though the query propagates the binding for the predicate \( p \), in order to correctly answer the query, we also need to evaluate the truth value of \( q(Y) \), which indirectly receives the binding through the body predicate \( a(X, Y) \). For instance, suppose that the program contains the facts \( a(1, 2) \) and \( b(1) \); then the atom \( q(2) \) is relevant for the query \( p(1) \) (i.e., it should belong to the magic set of the query), since the truth of \( q(2) \) would invalidate the derivation of \( p(1) \) from the above rule, due to the minimality of the semantics. It follows that, while propagating the binding, the head atoms of disjunctive rules must be all adorned as well.

However, the adornment of the head of one disjunctive rule \( r \) may give rise to multiple rules, having different adornments for the head predicates. This
process can be somehow seen as “splitting” r in multiple rules. While this is not a problem in the non-disjunctive case, the semantics of a disjunctive program may be affected. Consider, for instance, the program consisting of the rule \( p(X, Y) \lor q(Y, X) : a(X, Y) \), in which \( p \) and \( q \) are mutually exclusive (due to minimality) since they do not appear in any other rule head. Assuming the adornments \( p^{bf} \) and \( q^{bf} \) to be propagated, we might obtain rules whose heads have the form \( p^{bf}(X, Y) \lor q^{bf}(Y, X) \) (derived while propagating \( p^{bf} \)) and \( p^{fb}(X, Y) \lor q^{fb}(Y, X) \) (derived while propagating \( q^{bf} \)). These rules could support two atoms \( p^{bf}(m, n) \) and \( q^{bf}(n, m) \), while in the original program \( p(m, n) \) and \( p(n, m) \) could not hold simultaneously (due to semantic minimality), thus changing the original semantics.

The method proposed in [26] circumvents this problem by using some auxiliary predicates that collect all facts coming from the different adornments. For instance, in the above example, two rules of the form \( \text{collect}_p(X, Y) : p^{bf}(X, Y) \) and \( \text{collect}_p(X, Y) : p^{fb}(X, Y) \) are added for the predicate \( p \). The main lack of this approach is that collecting predicates will store a sizeable superset of all the atoms relevant to answer the given query. An important observation is that this set is defined in a deterministic way, which means that assumptions during the computation cannot be exploited for restricting the relevant part of the program. In terms of bottom-up systems, this implies that the optimization affects only the grounding portion of the solver. Intuitively, it would be beneficial to also have this form of conditional relevance, exploiting also relevance for assumptions. Indeed in Section 5, we provide experimental evidence for this intuition.

In the following, we propose a novel Magic Set method that guarantees semantic equivalence and also allows for the exploitation of conditional or dynamic relevance, overcoming a major drawback of APM.

### 3.2 DMS Algorithm

Our proposal to enhance the Magic Set method for disjunctive Datalog programs relies on two crucial features:

1. First, the semantics of the program is preserved by stripping off the adornments from non-magic predicates in modified rules, avoiding collecting predicates that can introduce overhead in the grounding process.

2. Second, the proposed Magic Set technique is not just a way to cut irrelevant rules from the ground program; in fact, it allows for dynamic determination of relevance, thus optimizing also the nondeterministic computation by disabling parts of the programs which are not relevant in any extension of the current computation state.

The algorithm DMS implementing these strategies is reported in Figure 1. Its input is a Datalog\(^2\)\(-\lor\) program \(^2\) \( \mathcal{P} \) and a query \( \mathcal{Q} \). The algorithm exploits a set

\(^2\)Note that the algorithm can be used for non-disjunctive and/or positive programs as a special case.
\textbf{Input:} A Datalog\textsuperscript{\dagger} program $P$, and a query $Q = g(\mathbf{t})$?

\textbf{Output:} The optimized program $\text{DMS}(Q, P)$.

\begin{algorithm}
\begin{algorithmic}[1]
\Var S: set of adorned predicates; modifiedRules$\_Q$,$\_P$, magicRules$\_Q$,$\_P$: set of rules;
\Begin
1. $S := \emptyset$; modifiedRules$\_Q$,$\_P := \emptyset$; magicRules$\_Q$,$\_P := \text{BuildQuerySeeds}(Q, S)$;
2. while $S \neq \emptyset$ do
3. $p^* :=$ an element of $S$; $S := S \setminus \{p^*\}$;
4. for each rule $r \in P$ and for each atom $p(\mathbf{t}) \in H(r)$ do
5. $r^* := \text{Adorn}(r, p^*, S)$;
6. magicRules$\_Q$,$\_P :=$ magicRules$\_Q$,$\_P \cup \text{Generate}(r^*)$;
7. modifiedRules$\_Q$,$\_P :=$ modifiedRules$\_Q$,$\_P \cup \{\text{Modify}(r^*)\}$;
8. end for
9. end while
10. $\text{DMS}(Q, P) :=$ modifiedRules$\_Q$,$\_P \cup$ modifiedRules$\_Q$,$\_P \cup EDB(P)$;
11. return $\text{DMS}(Q, P)$;
\End
\end{algorithmic}
\end{algorithm}

Figure 1: Disjunctive Magic Set algorithm (DMS) for Datalog\textsuperscript{\dagger} programs.

$S$ for storing all the adorned predicates to be used for propagating the binding of the query and, after all the adorned predicates are processed, outputs a rewritten program $\text{DMS}(Q, P)$ consisting of a set of \textit{modified} and \textit{magic} rules, stored by means of the sets \textit{modifiedRules$\_Q$,$\_P$} and \textit{magicRules$\_Q$,$\_P$}, respectively (together with the original EDB). The main steps of the algorithm are illustrated by means of the following running example.

\textbf{Example 3.1 (Strategic Companies [13])} Let $C$ be a collection of companies producing some goods in a set $G$, such that each company $c_i \in C$ is controlled by a set of other companies $O_i \subseteq C$. A subset of the companies $C' \subseteq C$ is a \textit{strategic set} if it is a minimal set of companies producing all the goods in $G$, such that if $O_i \subseteq C'$ for some $i = 1, \ldots, m$ then $c_i \in C'$ must hold.

We assume that each product is produced by at most two companies and that each company is controlled by at most three companies. It is known that the problem retains its hardness under this restriction. We assume that production of goods is represented by an EDB containing a fact \texttt{produced}$\_by$(p, c_1, c_2) for each product $p$ produced by companies $c_1$ and $c_2$, and that the control is represented by facts \texttt{controlled}$\_by$(c, c_1, c_2, c_3) for each company $c$ controlled by companies $c_1$, $c_2$, and $c_3$.\footnote{If a product is controlled by only one company, $c_2 = c_1$, and similar for companies controlled by fewer than three companies.} This problem can be modelled via the following disjunctive program $P_{sc}$:

\begin{align*}
 r_3 & : \text{sc}(C_1) \lor \text{sc}(C_2) : = \text{produced}$\_by$(P, C_1, C_2), \\
r_4 & : \text{sc}(C) : = \text{controlled}$\_by$(C, C_1, C_2, C_3), \text{sc}(C_1), \text{sc}(C_2), \text{sc}(C_3).
\end{align*}

Moreover, given a company $c \in C$, we consider a query $Q_{sc} = \text{sc}(c)$? asking whether $c$ belongs to some strategic set of $C$.\hfill \Box
The computation starts in step 1 by initializing $S$ and $\text{modifiedRules}_\mathcal{Q}_\mathcal{P}$ to the empty set. Then the function $\text{BuildQuerySeed}(\mathcal{Q}, \mathcal{P})$ is used for storing in $\text{modifiedRules}_\mathcal{Q}_\mathcal{P}$ the magic seed, and inserting in the set $S$ the adorned predicate of $\mathcal{Q}$. Note that we do not generate any query rules because standard atoms in the transformed program will not contain adornments.

Example 3.2 Given the query $\mathcal{Q}_{\text{sc}} = \text{sc}(c)$ and the program $\mathcal{P}_{\text{sc}}$, $\text{BuildQuerySeed}(\mathcal{Q}_{\text{sc}}, \mathcal{P}_{\text{sc}})$ creates the fact $\text{magic}_{\text{sc}}b(c)$ and inserts $\text{sc}b$ in $S$. □

The core of the algorithm (steps 2–9) is repeated until the set $S$ is empty, i.e., until there is no further adorned predicate to be propagated. In particular, an adorned predicate $p^α$ is removed from $S$ in step 3, and its binding is propagated in each (disjunctive) rule $r \in \mathcal{P}$ of the form

$$r : p(\bar{t}) \lor p_1(\bar{t}_1) \lor \cdots \lor p_n(\bar{t}_n) := q_1(\bar{s}_1), \ldots, q_j(\bar{s}_j),$$

$$\text{not } q_{j+1}(\bar{s}_{j+1}), \ldots, \text{not } q_m(\bar{s}_m).$$

(with $n \geq 0$) having an atom $p(\bar{t})$ in the head (note that the rule $r$ is processed a number of times equal to the number of head atoms with predicate $p$; steps 4–8).

1. Adornment. Step 5 implements the adornment of the rule. Different from the case of non-disjunctive positive programs, the binding of the predicate $p^α$ needs to be also propagated to the atoms $p_1(\bar{t}_1), \ldots, p_n(\bar{t}_n)$ in the head. Therefore, binding propagation has to be extended to the head atoms different from $p(\bar{t})$, which are therefore adorned according to a SIPS specifically conceived for disjunctive programs.

Definition 3.3 (SIPS for Datalog$^{\lor, \neg}$ rules) A SIPS for a Datalog$^{\lor, \neg}$ rule $r$ w.r.t. a binding $α$ for an atom $p(\bar{t}) \in H(r)$ is a pair $(\prec^p_r(\bar{t}), f^p_r(\bar{t}))$, where:

1. $\prec^p_r(\bar{t})$ is a strict partial order over the atoms in $H(r) \cup B^+(r) \cup B^-(r)$, such that:
   
   (a) $p(\bar{t}) \prec^p_r(\bar{t}) q(\bar{s})$, for all atoms $q(\bar{s}) \in H(r) \cup B^+(r) \cup B^-(r)$ different from $p(\bar{t})$;
   
   (b) for each pair of atoms $q(\bar{s}) \in (H(r) \setminus \{p(\bar{t})\}) \cup B^-(r)$ and $b(\bar{z}) \in H(r) \cup B^+(r) \cup B^-(r)$, $q(\bar{s}) \prec^p_r(\bar{t}) b(\bar{z})$ does not hold; and,

2. $f^p_r(\bar{t})$ is a function assigning to each atom $q(\bar{s}) \in H(r) \cup B^+(r) \cup B^-(r)$ a subset of the variables in $\bar{s}$—intuitively, those made bound when processing $q(\bar{s})$.

In particular, the difference w.r.t. SIPSes for Datalog rules is precisely in the dependency from $p(\bar{t})$ in addition to $α$, and in condition (1.b) stating that head atoms different from $p(\bar{t})$ and negative body literals cannot provide bindings to other atoms. In the following, we shall assume that each Datalog$^{\lor, \neg}$ program is provided in input together with some arbitrarily defined SIPS for Datalog$^{\lor, \neg}$.
rules \( \setminus^p_r (E) \) and \( \setminus^f_r (E) \). In fact, armed with \( \setminus^p_r (E) \) and \( \setminus^f_r (E) \), the adornment can be carried out precisely as we discussed for Datalog programs; in particular, we recall here that a variable \( X \) of an atom \( q(\bar{z}) \) in \( r \) is bound if and only if either:

1. \( X \in f_r^p (E) (q(\bar{z})) \) with \( q(\bar{z}) = p(\bar{t}) \); or,
2. \( X \in f_r^p (E) (b(\bar{z})) \) for an atom \( b(\bar{z}) \in B^+ (r) \) such that \( b(\bar{z}) \setminus^p_r (E) q(\bar{z}) \) holds.

The function \( \text{Adorn}(r, p^a, S) \) produces an adorned disjunctive rule \( r^a \) from an adorned predicate \( p^a \) and a suitable unadorned rule \( r \) (according to the bindings defined in the points (1) and (2) above), by inserting all newly adorned predicates in \( S \). Hence, in step 5 the rule \( r^a \) is of the form

\[
r^a : p^a (\bar{e}) v f_1^\beta_1 (\bar{t}_1) v \cdots v f_n^\beta_n (\bar{t}_n) := q_1^\beta_1 (\bar{s}_1), \ldots, q_j^\beta_j (\bar{s}_j),
\]

not \( q_{j+1}^\beta_{j+1} (\bar{s}_{j+1}), \ldots, \) not \( q_n^\beta_n (\bar{s}_n) \).

**Example 3.4** Let us resume from Example 3.2. We are supposing the adopted SIPS is passing the bindings via \textit{produced by} and \textit{controlled by} to the variables of \textit{sc} atoms, in particular

\[
\begin{align*}
\text{sc}(C_1) & \quad \setminus^p_{r_3} (C_1) & \text{produced by}(P, C_1, C_2) \\
\text{sc}(C_1) & \quad \setminus^p_{r_3} (C_1) & \text{sc}(C_2) \\
& \text{produced by}(P, C_1, C_2) & \setminus^p_{r_3} (C_1) & \text{sc}(C_2) \\
\text{sc}(C_2) & \quad \setminus^p_{r_3} (C_2) & \text{produced by}(P, C_1, C_2) \\
\text{sc}(C_2) & \quad \setminus^p_{r_3} (C_2) & \text{sc}(C_1) \\
& \text{produced by}(P, C_1, C_2) & \setminus^p_{r_3} (C_2) & \text{sc}(C_1) \\
\text{sc}(C) & \quad \setminus^p_{r_4} (C) & \text{controlled by}(C, C_1, C_2, C_3) \\
\text{sc}(C) & \quad \setminus^p_{r_4} (C) & \text{sc}(C_1) \\
\text{sc}(C) & \quad \setminus^p_{r_4} (C) & \text{sc}(C_2) \\
\text{sc}(C) & \quad \setminus^p_{r_4} (C) & \text{sc}(C_3) \\
\text{controlled by}(C, C_1, C_2, C_3) & \setminus^p_{r_4} (C) & \text{sc}(C_1) \\
\text{controlled by}(C, C_1, C_2, C_3) & \setminus^p_{r_4} (C) & \text{sc}(C_2) \\
\text{controlled by}(C, C_1, C_2, C_3) & \setminus^p_{r_4} (C) & \text{sc}(C_3)
\end{align*}
\]
When $sc^b$ is removed from the set $S$, we first select rule $r_3$ and the head predicate $sc(C_1)$. Then the adorned version is

$$r^a_3 : sc^b(C_1) \lor sc^b(C_2) :- produced_by(P, C_1, C_2).$$

Next, $r_3$ is processed again, this time with head predicate $sc(C_2)$, producing

$$r^a_{3,2} : sc^b(C_2) \lor sc^b(C_1) :- produced_by(P, C_1, C_2).$$

Finally, processing $r_4$ we obtain

$$r^a_4 : sc^b(C) :- controlled_by(C, C_1, C_2, C_3), sc^b(C_1), sc^b(C_2), sc^b(C_3).$$

(2) Generation. The algorithm uses the adorned rule $r^a$ for generating and collecting the magic rules in step 6. Since $r^a$ is a disjunctive rule with negation, $Generate(r^a)$ first produces a non-disjunctive and positive intermediate rule by moving the head atoms different from $p(\bar{x})$ and the atoms in $B^- (r)$ into $B^+ (r)$. Thus, the standard technique for Datalog rules, as described in Generation Step in Section 2, is applied.

Example 3.5 In the program of Example 3.4, from the rule $r^a_{3,1}$, we first produce its non-disjunctive intermediate rule

$$sc^b(C_1) :- sc^b(C_2), produced_by(P, C_1, C_2).$$

Thus, from this rule we generate the magic rule

$$r^*_3 : magic_sc^b(C_2) :- magic_sc^b(C_1), produced_by(P, C_1, C_2).$$

Similarly, from the rule $r^a_{3,2}$ we obtain

$$r^*_3 : magic_sc^b(C_1) :- magic_sc^b(C_2), produced_by(P, C_1, C_2).$$
Finally, \( r^a_4 \) gives rise to the following rules:

\[
\begin{align*}
  r^a_{4,1} : & \text{magic}_{sc}(C_1) : := \text{magic}_{sc}(C), \text{controlled}_by(C, C_1, C_2, C_3). \\
  r^a_{4,2} : & \text{magic}_{sc}(C_2) : := \text{magic}_{sc}(C), \text{controlled}_by(C, C_1, C_2, C_3). \\
  r^a_{4,3} : & \text{magic}_{sc}(C_3) : := \text{magic}_{sc}(C), \text{controlled}_by(C, C_1, C_2, C_3). \\
\end{align*}
\]

(3) **Modification.** In step 7 the modified rules are generated and collected. The only difference with the Datalog case is that the adornments are stripped off the original atoms. Hence, the function \( \text{Modify}(r^a) \) constructs a rule \( r' \) of the form:

\[
r' : p(\bar{s}) \lor p_1(\bar{s}_1) \lor \cdots \lor p_n(\bar{s}_n) : = \text{magic}(p^n(\bar{s})), \text{magic}(p_1^n(\bar{s}_1)), \ldots, \\
  \text{magic}(p_n^n(\bar{s}_n)), q_1(\bar{s}_1), \ldots, q_j(\bar{s}_j), \text{not } q_{j+1}(\bar{s}_{j+1}), \ldots, \text{not } q_k(\bar{s}_k).
\]

Finally, after all the adorned predicates have been processed, the algorithm outputs the program \( \text{DMS}(Q, P) \).

**Example 3.6** In our running example, we derive the following set of modified rules:

\[
\begin{align*}
  r^a_{3,1} : & \text{sc}(C_1) \lor \text{sc}(C_2) : := \text{magic}_{sc}(C_1), \text{magic}_{sc}(C_2), \\
  & \text{controlled}_by(P, C_1, C_2). \\
  r^a_{3,2} : & \text{sc}(C_2) \lor \text{sc}(C_1) : := \text{magic}_{sc}(C_2), \text{magic}_{sc}(C_1), \\
  & \text{controlled}_by(P, C_1, C_2). \\
  r^a_4 : & \text{sc}(C) : := \text{magic}_{sc}(C), \text{controlled}_by(C, C_1, C_2, C_3), \\
  & \text{sc}(C_1), \text{sc}(C_2), \text{sc}(C_3).
\end{align*}
\]

Here, \( r^a_{3,1} \) (resp. \( r^a_{3,2}, r^a_4 \)) is derived by adding magic predicates and stripping off adornments for the rule \( r^a_{3,1} \) (resp. \( r^a_{3,2}, r^a_4 \)). Thus, the optimized program \( \text{DMS}(Q_{sc}, P_{sc}) \) comprises the above modified rules as well as the magic rules in Example 3.5, and the magic seed \( \text{magic}_{sc}(c) \). (together with the original EDB).

**3.3 Query Equivalence Results**

We conclude the presentation of the DMS algorithm by formally proving its correctness. To this aim we use the well established notion of unfounded set for disjunctive Datalog programs (possibly with negation) defined in [34]. Since we deal with total interpretations, represented as the set of atoms interpreted as true, the definition of unfounded set can be restated as follows.

**Definition 3.7 (Unfounded sets)** Let \( I \) be an interpretation for a Datalog program \( P \), and \( X \subseteq B_P \) be a set of atoms. Then \( X \) is an unfounded set for \( P \) w.r.t. \( I \) if and only if for each ground rule \( r_g \in \text{Ground}(P) \) with \( X \cap H(r_g) \neq \emptyset \), either (1.a) \( B^+(r_g) \not\subseteq I \), or (1.b) \( B^-(r_g) \cap I \neq \emptyset \), or (2) \( B^+(r_g) \cap X \neq \emptyset \), or (3) \( H(r_g) \cap (I \setminus X) \neq \emptyset \).

Intuitively, conditions (1.a), (1.b) and (3) check if the rule is satisfied by \( I \) regardless of the atoms in \( X \), while condition (2) assures that the rule can be satisfied by taking the atoms in \( X \) as false.
Example 3.8 Consider again the program $\mathcal{P}_{sc}$ of Example 3.1 and assume $EDB(\mathcal{P}_{sc}) = \{\text{produced by}(p, c, c_1)\}$. Then $\text{Ground}(\mathcal{P}_{sc})$ consists of the rule

$$r_{sc} : \text{sc}(c) \lor \text{sc}(c_1) :- \text{produced by}(p, c, c_1).$$

(together with facts, and rules having some ground instance of EDB predicate not occurring in $EDB(\mathcal{P}_{sc})$, omitted for simplicity). Consider now the interpretation $M_{sc} = \{\text{produced by}(p, c, c_1), \text{sc}(c)\}$. Thus, $\{\text{sc}(c_1)\}$ is an unfounded set for $\mathcal{P}$ w.r.t. $M_{sc}$ ($r_{sc}$ satisfies condition (3) of Definition 3.7), while $\{\text{sc}(c), \text{sc}(c_1)\}$ is not ($r_{sc}$ violets all conditions).

The following result immediately follows from the characterization of unfounded sets in [34].

Theorem 3.9 ([34]) Let $I$ be an interpretation for a Datalog' program $\mathcal{P}$. Then, for any stable model $M \supseteq I$ of $\mathcal{P}$, and for each unfounded set $X$ of $\mathcal{P}$ w.r.t. $I$, $M \cap X = \emptyset$ holds.

Example 3.10 The interpretation $M_{sc}$ of Example 3.8 is a stable model of $\mathcal{P}_{sc}$. The careful reader can check that each unfounded set for $\mathcal{P}_{sc}$ w.r.t. $M_{sc}$ is disjoint from $M_{sc}$.

Armed with these notions and Theorem 3.9, we now proceed to prove the correctness of the DMS strategy. In particular, we shall first show that the method is sound in that for each stable model $M$ of $\text{DMS}(\mathcal{Q}, \mathcal{P})$, there is a stable model $M'$ of $\mathcal{P}$ such that $M'|_\mathcal{Q} = M|_\mathcal{Q}$ (i.e., the two models coincide when restricted to the query). Then we prove that the method is also complete, i.e., for each stable model $M'$ of $\mathcal{P}$, there is a stable model $M$ of $\text{DMS}(\mathcal{Q}, \mathcal{P})$ such that $M'|_\mathcal{Q} = M|_\mathcal{Q}$.

In both parts of the proof, we shall exploit the following (syntactic) relationship between the original program and the transformed one.

Lemma 3.11 Let $\mathcal{P}$ be a Datalog' program, $\mathcal{Q}$ a query, and let $\text{magic}(p^\alpha(\bar{e}))$ be a ground atom\footnote{Note that in this way the Lemma refers only to rules that contain a head atom for which a magic predicate has been generated during the transformation.} in $B_{\text{DMS}}(\mathcal{Q}, \mathcal{P})$ (the base of the transformed program). Then the ground rule

$$r_g : p(\bar{e}) \lor p_1(\bar{e}_1) \lor \cdots \lor p_n(\bar{e}_n) :- q_1(\bar{s}_1), \ldots, q_j(\bar{s}_j),$$

$$\text{not } q_{j+1}(\bar{s}_{j+1}), \ldots, \text{not } q_n(\bar{s}_n).$$

belongs to $\text{Ground}(\mathcal{P})$ if and only if the ground rule

$$r'_g : p(\bar{e}) \lor p_1(\bar{e}_1) \lor \cdots \lor p_n(\bar{e}_n) :- \text{magic}(p^\alpha(\bar{e})), \text{magic}(p_1^\alpha(\bar{e}_1)), \ldots, \text{magic}(p_n^\alpha(\bar{e}_n)), q_1(\bar{s}_1), \ldots, q_j(\bar{s}_j), \text{not } q_{j+1}(\bar{s}_{j+1}), \ldots, \text{not } q_n(\bar{s}_n).$$

belongs to $\text{Ground}(\text{DMS}(\mathcal{Q}, \mathcal{P}))$.\footnote{Note that in this way the Lemma refers only to rules that contain a head atom for which a magic predicate has been generated during the transformation.}
3.3.1 Soundness of the Magic Set Method

Let us now start with the first part of the proof, in particular, by stating some further definition and notation. Given a model \( M' \) of \( \text{DMS}(Q, P) \), and a model \( N' \subseteq M' \) of \( \text{Ground}(\text{DMS}(Q, P))^{M'} \), we next define the set of atoms which are relevant for \( Q \) but are false w.r.t. \( N' \).

**Definition 3.12 (Killed atoms)** Given a model \( M' \) for \( \text{DMS}(Q, P) \), and a model \( N' \subseteq M' \) of \( \text{Ground}(\text{DMS}(Q, P))^{M'} \), the set killed\(_{Q,P}(N')\) of the killed atoms w.r.t. \( M' \) and \( N' \) is defined as:

\[
\{ k(\bar{x}) \in B_P \setminus N' \mid \text{either } k \text{ is an EDB predicate, or there is a binding } \alpha \text{ such that } \text{magic}(k^\alpha(\bar{x})) \in N' \} \]
Example 3.13 We consider the program \( \text{DMS}(\text{Q}_{sc}, \text{P}_{sc}) \) presented in Section 3.2 (we recall that \( \text{Q}_{sc} = \text{sc}(c) \)), the EDB introduced in Example 3.8, and the (stable) model \( M'_{sc} = M_{sc} \cup \{ \text{magic}_{sc}(c), \text{magic}_{sc}(c_1) \} \) for \( \text{DMS}(\text{Q}_{sc}, \text{P}_{sc}) \) (where \( M_{sc} \) is as reported in Example 3.8). Then \( \text{Ground}(\text{DMS}(\text{Q}_{sc}, \text{P}_{sc}))^{M_{sc}} \) consists of the rules

\[
\begin{align*}
\text{magic}_{sc}(c) & \quad: \quad \text{magic}_{sc}(c).
\text{sc}(c) \vee \text{sc}(c_1) & \quad: \quad \text{magic}_{sc}(c), \text{magic}_{sc}(c_1), \text{produced_by}(p, c, c_1).
\end{align*}
\]

Since \( M'_{sc} \) is also a model of the program above, we can compute \( \text{killed}_{\text{Q}_{sc}, \text{P}_{sc}}(M'_{sc}) \) and check that \( \text{sc}(c_1) \) belongs to it because of \( \text{magic}_{sc}(c_1) \) in \( M'_{sc} \). Note that, by definition, also false ground instances of EDB predicates like \( \text{produced_by}(p, c, c) \) or \( \text{controlled_by}(c, c_1, c_1) \) belong to \( \text{killed}_{\text{Q}_{sc}, \text{P}_{sc}}(M'_{sc}) \). Moreover, the careful reader can check that no other atom belongs to this set.

The intuition underlying the definition above is that killed atoms are either false ground instances of some EDB predicate, or false atoms which are relevant with respect to \( Q \) (for there exists an associated magic atom in the model \( N' \)); since \( N' \) is a model of \( \text{Ground}(\text{DMS}(Q, P))^{M'} \) contained in \( M' \), we expect that these atoms are also false in any stable model for \( P \) “extending” \( M'_{B_P} \) (which, we recall here, is the model \( M' \) restricted on the atoms originally occurring in \( P \)).

Example 3.14 For instance, \( M'_{sc} \) of Example 3.13 restricted to the symbols of \( \text{P}_{sc} \) coincides with \( M_{sc} \) of Example 3.8. From Example 3.8, we know that \( \{ \text{sc}(c_1) \} \) is an unfounded set for \( \text{P}_{sc} \) w.r.t. \( M_{sc} \). Each other atom \( k(\bar{x}) \) in \( \text{killed}_{\text{Q}_{sc}, \text{P}_{sc}}(M'_{sc}) \) is such that \( k \) is an EDB predicate. Hence, we have that \( \text{killed}_{\text{Q}_{sc}, \text{P}_{sc}}(M'_{sc}) \) is an unfounded set for \( \text{P}_{sc} \) w.r.t. \( M_{sc} \). Therefore, as a consequence of Theorem 3.9, each stable model of \( \text{P}_{sc} \) “extending” \( M_{sc} \) (in this case only \( M_{sc} \) itself) is disjoint from \( \text{killed}_{\text{Q}_{sc}, \text{P}_{sc}}(M'_{sc}) \).

This intuition is formalized below.

Proposition 3.15 Let \( M' \) be a model for \( \text{DMS}(Q, P) \), and \( N' \subseteq M' \) a model of \( \text{Ground}(\text{DMS}(Q, P))^{M'} \). Then \( \text{killed}_{Q, P}(N') \) is an unfounded set for \( P \) w.r.t. \( M'_{B_P} \).

Proof. Let \( r_g \) be a rule in \( \text{Ground}(P) \) such that \( k(\bar{x}) \in H(r_g) \cap \text{killed}_{Q, P}(N') \):

\[
\begin{align*}
\text{r}_g : \quad k(\bar{x}) & \quad: \quad p_1(\bar{x}_1) \lor \cdots \lor p_n(\bar{x}_n) := q_1(\bar{s}_1), \ldots, q_j(\bar{s}_j), \\
& \quad: \quad \text{not } q_{j+1}(\bar{s}_{j+1}), \ldots, \text{not } q_k(\bar{s}_k).
\end{align*}
\]

According to Definition 3.7, we have to show that either (1.a) \( B^+(r_g) \not\subseteq M'_{B_P} \), or (1.b) \( B^+(r_g) \cap M'_{B_P} \neq \emptyset \), or (2) \( B^+(r_g) \cap \text{killed}_{Q, P}(N') \neq \emptyset \), or (3) \( H(r_g) \cap (M'_{B_P} \setminus \text{killed}_{Q, P}(N')) \neq \emptyset \).

20
Thus, we can apply Lemma 3.11 and conclude the existence of a ground rule $r_g' \in \text{Ground}(\text{DMS}(Q, P))$ such that:

$$
r_g': \kappa(\bar{e}) \lor p_1(\bar{e}_1) \lor \ldots \lor p_n(\bar{e}_n) := \text{magic}(\kappa(\bar{e})), \text{magic}(p_1^0(\bar{e}_1)), \ldots, \text{magic}(p_n^0(\bar{e}_n)), q_1(\bar{s}_1), \ldots, q_j(\bar{s}_j), \text{not } q_{j+1}(\bar{s}_{j+1}), \ldots, \text{not } q_n(\bar{s}_n).
$$

Since $M'$ is a model of $\text{DMS}(Q, P)$, when considering the rule $r_g'$, three cases may occur:

- $B^-(r_g') \cap M' \neq \emptyset$, i.e., the negative body of $r_g'$ is false w.r.t. $M'$.
  
  In this case, since $B^-(r_g) = B^-(r_g')$ and $B^-(r_g) \subseteq B_P$, from $B^-(r_g') \cap M' \neq \emptyset$ we immediately conclude $B^-(r_g) \cap M'|_{B_P} \neq \emptyset$, i.e., (1.b) holds.

- $B^+(r_g') \not\subseteq M'$, i.e., the positive body of $r_g'$ is false w.r.t. $M'$.
  
  In this case, we shall show that (1.a) holds, i.e., that $B^+(r_g) \not\subseteq M'|_{B_P}$. Consider first a modified rule $r' \in \text{DMS}(Q, P)$ such that $r_g' = r' \vartheta$ for some substitution $\vartheta$:

$$
r': \kappa(\bar{e}) \lor p_1(\bar{e}_1') \lor \ldots \lor p_n(\bar{e}_n') := \text{magic}(\kappa(\bar{e})), \text{magic}(p_1^0(\bar{e}_1')), \ldots, \text{magic}(p_n^0(\bar{e}_n')), q_1(\bar{s}_1'), \ldots, q_j(\bar{s}_j'), \text{not } q_{j+1}(\bar{s}_{j+1}'), \ldots, \text{not } q_n(\bar{s}_n').
$$

During the Generation step preceding the production of $r'$, a magic rule $r_i^*$ such that $H(r_i^*) = \{ \text{magic}(p_i^0(\bar{e}_i')) \}$ and $B^+(r_i^*) \subseteq \{ \text{magic}(\kappa(\bar{e})), q_1(\bar{s}_1'), \ldots, q_j(\bar{s}_j'), \text{not } q_{j+1}(\bar{s}_{j+1}'), \ldots, \text{not } q_n(\bar{s}_n') \}$ has been produced for each $1 \leq i \leq n$ (we recall that magic rules have empty negative body). Hence, since the variables of $r_i^*$ are a subset of the variables of $r'$, by applying the substitution $\vartheta$ to $r_i^*$ we obtain a ground rule $r_{i,g}^*$ such that $H(r_{i,g}^*) = \{ \text{magic}(p_i^0(\bar{e}_i')), q_1(\bar{s}_1'), \ldots, q_j(\bar{s}_j') \} = \{ \text{magic}(\kappa(\bar{e})), q_1(\bar{s}_1), \ldots, q_j(\bar{s}_j), \text{not } q_{j+1}(\bar{s}_{j+1}), \ldots, \text{not } q_n(\bar{s}_n) \}$.

Assume now by contradiction that $B^+(r_g) \subseteq M'|_{B_P}$ holds. Then $B^+(r_{i,g}^*)$ is in turn contained in $M'$, since $M'|_{B_P} \subseteq M'$ and $\text{magic}(\kappa(\bar{e})) \in N' \subseteq M'$. Thus, since $M'$ is a model for $\text{DMS}(Q, P)$, the head of $r_{i,g}^*$ must be true w.r.t. $M'$, that is, $\text{magic}(p_i^0(\bar{e}_i')) \in M'$ holds for each $1 \leq i \leq n$. Hence, $B^+(r_g') \not\subseteq M'$, which contradicts the original assumption. Then $B^+(r_g) \not\subseteq M'|_{B_P}$ must hold, i.e., condition (1.a).

- $H(r_g') \cap M' \neq \emptyset$, i.e., the head of $r_g'$ is true w.r.t. $M'$.
  
  In this case, we shall show that if both previous cases do not hold, then (2) or (3) hold. Assume that $B^+(r_g') \subseteq M'$ and $B^-(r_g') \cap M' = \emptyset$ hold. From $B^-(r_g') \cap M' = \emptyset$ we can conclude that there is a rule in $\text{Ground}(\text{DMS}(Q, P))^{M'}$ obtained from $r_g'$ by removing its negative body literals. Consider now the rules $r_{i,g}^*$ produced during the Generation step, for each $1 \leq i \leq n$. We distinguish two cases.
(I) If \( \{ q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j) \} \subseteq N' \), since \( \text{magic}(k^\alpha(\bar{e})) \in N' \), we can conclude that \( B^+(r'_{q_i}) \subseteq N' \), for each \( 1 \leq i \leq n \). Moreover, since \( N' \) is a model of \( \text{Ground}(\text{DMS}(Q, P))^M \), the latter implies that \( \text{magic}(p_1^\alpha(\bar{e}_i)) \in N' \), for each \( 1 \leq i \leq n \). Then \( B^+(r'_{q_i}) \subseteq N' \) holds, and so \( H(r'_{q_i}) \cap N' \neq \emptyset \) (because \( N' \) is a model of \( \text{Ground}(\text{DMS}(Q, P))^M \)). We now observe that \( H(r'_{q_i}) \cap (M'|_{B_P} \setminus \text{killed}_Q^M(N')) \neq \emptyset \) is equivalent to \( (H(r'_{q_i}) \cap M'|_{B_P}) \setminus \text{killed}_Q^M(N') \neq \emptyset \). Moreover, the latter is equivalent to \( (H(r_{q_i}) \cap M') \setminus \text{killed}_Q^M(N') \neq \emptyset \) because \( H(r'_{q_i}) \) contains only standard atoms and \( H(r_{q_i}) = H(r_{q_i}) \). In addition, from \( N' \subseteq M' \) we conclude \( H(r_{q_i}) \cap N' \subseteq H(r_{q_i}) \cap M' \), and by Definition 3.12, \( N' \cap \text{killed}_Q^M(N') = \emptyset \). Hence, \( (H(r_{q_i}) \cap M') \setminus \text{killed}_Q^M(N') \supseteq H(r_{q_i}) \cap N' \), which is not empty, and so condition (3) holds.

(II) Otherwise, \( \{ q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j) \} \not\subseteq N' \). In this case, we consider again the modified rule \( r' \) such that \( r' = r' \). Such a rule \( r' \) has been generated from a rule \( r \) of \( P \) such that \( r_g = r'^v \):

\[
r : k(\bar{e}') \lor p_1(\bar{e}_1') \lor \cdots \lor p_{i+1}(\bar{e}_{i+1}') : q_1(\bar{s}_1'), \ldots, q_1(\bar{s}_j'), \not q_{i+1}(\bar{s}_{i+1}'), \ldots, \not q_n(\bar{s}_n'),
\]

Thus, there is \( q_u(\bar{s}_u) \in \{ q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j) \} \setminus N' \) such that \( q_u(\bar{s}_u) \not\prec^k_r \), \( q_u(\bar{s}_u) \not\in N' \) (for all \( 1 \leq v \leq j \)). Our aim is to show that such a \( q_u(\bar{s}_u) \) belongs to \( \text{killed}_Q^M(N') \), as in this case condition (2) will hold. If \( q_u \) is an IDB predicate, then a magic rule \( r^* \) such that \( H(r^*) = \{ \text{magic}(q_u^k(\bar{s}_u)) \} \) and \( B^+(r^*) = \{ \text{magic}(k^\alpha(\bar{e}')) \} \cup \{ q_u(\bar{s}_u) | q_u(\bar{s}_u) \prec^k_r q_u(\bar{s}_u') \} \) has been produced during the Generation step preceding the production of \( r' \) (we recall that magic rules have empty negative body). Hence, we can conclude that \( B^+(r'^v) \subseteq N' \), and so \( \text{magic}(q_u^k(\bar{s}_u)) = \text{magic}(q_u^k(\bar{s}_u')) \neq \emptyset \) because \( N' \) is a model of \( \text{Ground}(\text{DMS}(Q, P))^M \). Otherwise \( q_u \) is an EDB predicate. In both cases, \( q_u(\bar{s}_u) \in \text{killed}_Q^M(N') \) by Definition 3.12.

\[ \square \]

We can now complete the first part of the proof.

**Lemma 3.16** For each stable model \( M' \) of \( \text{DMS}(Q, P) \), there is a stable model \( M \) of \( P \) such that \( M \supseteq M'|_{B_P} \).

**Proof.** Let \( M \) be a stable model of \( P \cup M'|_{B_P} \), the program obtained by adding to \( P \) a fact for each atom in \( M'|_{B_P} \). Clearly enough, \( M \) is in turn a model of \( P \) such that \( M \supseteq M'|_{B_P} \). We shall show that \( M \) is in fact a stable model of \( P \).

Assume, for the sake of contradiction, that \( M \) is not a stable model of \( P \) and let \( N \subset M \) be a model of \( \text{Ground}(P)^M \).

Consider the interpretation \( N' = (N \cap M'|_{B_P}) \cup (M' \setminus B_P) \). By construction, note that \( N' \subseteq M' \), since \( M' \) coincides with \( M'|_{B_P} \cup (M' \setminus B_P) \). In fact, in the
case where $N' = M'$, we would have that $N \supseteq M'|_{B_p}$, since $(N \cap M'|_{B_p})$ and $(M' \setminus B_p)$ are disjoint. Hence, $N$ would not only be a model for $\text{Ground}(\mathcal{P})^M$ but also a model for $\text{Ground}(\mathcal{P} \cup M'|_{B_p})^M$, while on the other hand $N \subset M$ holds. However, this is impossible, since $M$ is a stable model of $\mathcal{P} \cup M'|_{B_p}$. So, $N' \subset M'$ must hold.

Then we have to consider the following two kinds of rules in $\text{Ground}(\text{DMS}(\mathcal{Q}, \mathcal{P}))^{M'}$:

1. Consider first a rule obtained by removing the negative literals from a ground modified rule $r'_g \in \text{Ground}(\text{DMS}(\mathcal{Q}, \mathcal{P}))$ where

$$r'_g : p(\xi) \lor p_1(\xi_1) \lor \ldots \lor p_n(\xi_n) := \text{magic}(p^n(\xi)), \text{magic}(p^n_1(\xi_1)), \ldots, \text{magic}(p^n_n(\xi_n)), q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j), \text{not } q_{j+1}(\bar{s}_{j+1}), \ldots, \text{not } q_k(\bar{s}_k).$$

and where $B^+(r'_g) \subseteq N'$. So, $B^-(r'_g) \cap M' = \emptyset$ holds by definition of reduct.

Our aim is to show that $H(r'_g) \cap N' \neq \emptyset$, so that $N'$ is indeed a model for the rule obtained from $r'_g$. To this end, since $N' \subset M'$, we note that $B^+(r'_g) \subseteq N'$ implies $B^+(r'_g) \subset M'$. So, by Definition 3.12 and since $B^+(r'_g) \subset M'$, we have that $H(r'_g) \setminus M' \subseteq \text{killed}_{\mathcal{Q}, \mathcal{P}}^M(M')$. In addition, we can show that the IDB atoms in $B^-(r'_g)$ belong to $\text{killed}_{\mathcal{Q}, \mathcal{P}}^{M'}(M')$. To this aim, consider a modified rule $r' \in \text{DMS}(\mathcal{Q}, \mathcal{P})$ such that $r'_g = r'\vartheta$ for some substitution $\vartheta$:

$$r' : p(\bar{\xi}') \lor p_1(\bar{\xi}_1') \lor \ldots \lor p_n(\bar{\xi}_n') := \text{magic}(p^n(\bar{\xi}')), \text{magic}(p^n_1(\bar{\xi}_1)'), \ldots, \text{magic}(p^n_n(\bar{\xi}_n)'), q_1(\bar{s}_1'), \ldots, q_1(\bar{s}_j'), \text{not } q_{j+1}(\bar{s}_{j+1}'), \ldots, \text{not } q_k(\bar{s}_k').$$

During the Generation step preceding the production of $r'$, a magic rule $r^*_i$ such that $H(r^*_i) = \{\text{magic}(q^i_j(\bar{s}_j))\}$ and $B^+(r^*_i) \subseteq \{\text{magic}(p^i(\bar{\xi}))\}$, $q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j)$ has been produced for each $j + 1 \leq i \leq m$ such that $q_i$ is an IDB predicate (we recall that magic rules have empty negative body). Hence, since the variables of $r^*_i$ are a subset of the variables of $r'$, the substitution $\vartheta$ can be used to map $r^*_i$ to a ground rule $r^*_i \vartheta = r^*_i \vartheta$ such that $H(r^*_i \vartheta) = \{\text{magic}(q^i_j(\vartheta(\bar{s}_j)))\}$ and $B^+(r^*_i \vartheta) \subseteq \{\text{magic}(p^i(\vartheta(\bar{\xi})))\}$, $q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j) \subseteq B^+(r'_g)$. Thus, the head of $r^*_i \vartheta$ must be true w.r.t. $M'$, that is, $\text{magic}(q^i_j(\vartheta(\bar{s}_j))) \in M'$ holds for each $j + 1 \leq i \leq m$ such that $q_i$ is an IDB predicate. So, by Definition 3.12, from $B^-(r'_g) \cap M' = \emptyset$ and $B^+(r'_g) \subset M'$, we can conclude that $B^-(r'_g) \subseteq \text{killed}_{\mathcal{Q}, \mathcal{P}}^{M'}(M')$.

We recall that, by Proposition 3.15, we already know that $\text{killed}_{\mathcal{Q}, \mathcal{P}}^{M'}(M')$ is an unfounded set for $\mathcal{P}$ w.r.t. $M'|_{B_p}$. In fact, one may notice that $\text{killed}_{\mathcal{Q}, \mathcal{P}}^{M'}(M')$ is an unfounded set for $\mathcal{P} \cup M'|_{B_p}$ w.r.t. $M'|_{B_p}$ too, since
the rules added to \( \mathcal{P} \) are facts corresponding to the atoms in \( M'|_{B_P} \) and \( M'|_{B_P} \cap \text{kill}_M(M') = \emptyset \) by Definition 3.12. Thus, since \( M \supseteq M'|_{B_P} \) and \( M \) is a stable model of \( \mathcal{P} \cup M'|_{B_P} \), we can apply Theorem 3.9 in order to conclude that \( M \cap \text{kill}_M(M') = \emptyset \). Then the following relationships are established:

\[
(H(r'_g) \setminus M') \cap M = \emptyset, \tag{1}
\]

\[
B^-(r'_g) \cap M = \emptyset, \tag{2}
\]

that is, the atoms in \( H(r'_g) \cup B^-(r'_g) \) which are false w.r.t. \( M' \) are also false w.r.t. \( M \). Moreover, since \( H(r'_g) \subseteq B_P \) and \( M \supseteq M'|_{B_P} \), Equation 1 implies that:

\[
H(r'_g) \cap M' = H(r'_g) \cap M'|_{B_P} = H(r'_g) \cap M. \tag{3}
\]

Let us now apply Lemma 3.11 in order to conclude the existence of a rule

\[
\text{such that } r_g : p(\bar{x}) \lor p_1(\bar{x}_1) \lor \cdots \lor p_n(\bar{x}_n) := q_1(\bar{z}_1), \ldots, q_j(\bar{z}_j), \not q_{j+1}(\bar{z}_{j+1}), \ldots, \not q_m(\bar{z}_m).
\]

such that \( r_g \in \text{Ground}(\mathcal{P}) \). In particular, note that \( B^-(r_g) = B^-(r'_g) \), and so Equation 2 implies that there is a rule in \( \text{Ground}(\mathcal{P})^M \) obtained from \( r_g \) by removing the atoms in \( B^-(r_g) \). Note also that \( B^+(r_g) = B^+(r'_g) \cap B_P \subseteq N' \cap B_P \) (since \( B^+(r'_g) \subseteq N' \)). Thus, by definition of \( N' \), \( B^+(r_g) \subseteq N \) (more specific, \( B^+(r_g) \subseteq N \cap M'|_{B_P} \)). Moreover, since \( N \) is a model of \( \text{Ground}(\mathcal{P})^M \), the latter entails that:

\[
H(r_g) \cap N \neq \emptyset. \tag{4}
\]

In addition, since \( H(r_g) \subseteq B_P \), by definition of \( N' \) we can conclude that (a) \( H(r_g) \cap N' = (H(r_g) \cap (N \cap M')_{B_P}) = (H(r_g) \cap N) \cap (H(r_g) \cap M'|_{B_P}) \). Now, recall that \( N \subseteq M \). Thus, (b) \( H(r_g) \cap N \subseteq H(r_g) \cap M = H(r_g) \cap M'|_{B_P} \), where the latter equality is obtained by combining Equation 3 and \( H(r_g) = H(r'_g) \). From (a) and (b) we can finally conclude that \( H(r_g) \cap N' \cap M'|_{B_P} = H(r_g) \cap N \cap M'|_{B_P} \), which is not empty by Equation 4.

(2) Consider a ground magic rule \( r^*_g \in \text{Ground}(\text{DMS}(\mathcal{Q}, \mathcal{P}))^{M'} \) such that \( B^+(r^*_g) \subseteq N' \), and let \( \text{magic}(p^n(\bar{x})) \) be the (only) atom in \( H(r^*_g) \). Since \( N' \subseteq M ', B^+(r^*_g) \subseteq N' \) implies that \( B^+(r^*_g) \subseteq M' \). In fact, since \( M' \) is a model of \( \text{DMS}(\mathcal{Q}, \mathcal{P}) \) and \( |H(r^*_g)| = 1, \text{magic}(p^n(\bar{x})) \in M' \) must hold (we recall that \( B^-(r^*_g) = \emptyset \) because \( r^*_g \) is a magic rule). Moreover, since \( B_P \) does not contain any magic atom, \( \text{magic}(p^n(\bar{x})) \) is also contained in \( M' \setminus B_P \). Thus, by construction of \( N' \), we can conclude that \( H(r^*_g) \cap N' \neq \emptyset \).

Therefore, any rule in \( \text{Ground}(\text{DMS}(\mathcal{Q}, \mathcal{P}))^{M'} \) is satisfied by \( N' \), which is impossible since \( N' \subseteq M' \) and \( M' \) is a stable model of \( \text{DMS}(\mathcal{Q}, \mathcal{P}) \).
Theorem 3.17 Let $\mathcal{Q}$ be a query for a Datalog$^w$~ program $\mathcal{P}$. Then, for each stable model $M'$ of $\text{DMS}(\mathcal{Q}, \mathcal{P})$, there is a stable model $M$ of $\mathcal{P}$ such that $M'|\mathcal{Q} = M|\mathcal{Q}$.

Proof. Because of Lemma 3.16, for each stable model $M'$ of $\text{DMS}(\mathcal{Q}, \mathcal{P})$, there is a stable model $M$ of $\mathcal{P}$ such that $M \supseteq M'|_{B_P}$. Thus, we trivially have that $M|\mathcal{Q} \supseteq M'|\mathcal{Q}$ holds. We now show that the inclusion cannot be proper.

In fact, by definition of $\text{DMS}(\mathcal{Q}, \mathcal{P})$, the magic seed is associated to any ground instance of $\mathcal{Q}$. Then $B_P|\mathcal{Q} \setminus M' \subseteq \text{killed}^{M'}_{\mathcal{Q}, \mathcal{P}}(M')$ by Definition 3.12 (we recall that $B_P|\mathcal{Q}$ denotes the ground instances of $\mathcal{Q}$). By Proposition 3.15, $\text{killed}^{M'}_{\mathcal{Q}, \mathcal{P}}(M')$ is an unfounded set for $\mathcal{P}$ w.r.t. $M'|_{B_P}$. Hence, by Theorem 3.9, we have that $M \cap \text{killed}^{M'}_{\mathcal{Q}, \mathcal{P}}(M') = \emptyset$. It follows that $M \cap (B_P|\mathcal{Q} \setminus M') = \emptyset$. Thus, $M|\mathcal{Q} \setminus M'|\mathcal{Q} = \emptyset$, which combined with $M \supseteq M'|_{B_P}$ implies $M|\mathcal{Q} = M'|\mathcal{Q}$.

3.3.2 Completeness of the Magic Set Method

For the second part of the proof, we construct an interpretation for $\text{DMS}(\mathcal{Q}, \mathcal{P})$ based on one for $\mathcal{P}$.

Definition 3.18 (Magic variant) Let $I$ be an interpretation for $\mathcal{P}$. We define an interpretation $\text{variant}_I^{\mathcal{Q}, \mathcal{P}}(I)$ for $\text{DMS}(\mathcal{Q}, \mathcal{P})$, called the magic variant of $I$ w.r.t. $\mathcal{Q}$ and $\mathcal{P}$, as the fixpoint of the following sequence:

\begin{align*}
\text{variant}_0^{\mathcal{Q}, \mathcal{P}}(I) &= EDB(\mathcal{P}) \cup \{\text{magic}(g^a(\overline{t})) : \text{magic}(g^a(\overline{t})) \text{ is the query seed} \} \\
\text{variant}_{i+1}^{\mathcal{Q}, \mathcal{P}}(I) &= \text{variant}_i^{\mathcal{Q}, \mathcal{P}}(I) \cup \{p(\overline{t}) \in I : \text{there is a binding } g \text{ such that } \text{magic}(p^g(\overline{t})) \in I' \} \cup \\
&\quad \{\text{magic}(p^g(\overline{t})) : \exists r^+_g \in \text{Ground}(\text{DMS}(\mathcal{Q}, \mathcal{P})) \text{ such that } \\
&\quad \quad \text{magic}(p^g(\overline{t})) \in H(r^+_g) \text{ and } B^+(r^+_g) \subseteq I', \quad \forall i \geq 0 \}
\end{align*}

Example 3.19 Consider the interpretation $M_{sc}$ for $\mathcal{P}_{sc}$ reported in Example 3.8. We next compute the magic variant $\text{variant}_{\mathcal{Q}_{sc}, \mathcal{P}_{sc}}^{\mathcal{Q}}(M_{sc})$ of $M_{sc}$ w.r.t. $\mathcal{Q}_{sc}$ and $\mathcal{P}_{sc}$. We start the sequence with the query seed $\text{variant}_0^{\mathcal{Q}_{sc}, \mathcal{P}_{sc}}(M_{sc}) = \{\text{magic}_{sc}(c)\}$. For $\text{variant}_1^{\mathcal{Q}_{sc}, \mathcal{P}_{sc}}(M_{sc})$, we add $\text{sc}(c)$ (because $\text{magic}_{sc}(c)$ belongs to $\text{variant}_0^{\mathcal{Q}_{sc}, \mathcal{P}_{sc}}(M_{sc})$), and $\text{magic}_{sc}(c_1)$ (because $\text{magic}_{sc}(c_1) = \text{magic}_{sc}(c)$). Any other element of the sequence coincides with $\text{variant}_1^{\mathcal{Q}_{sc}, \mathcal{P}_{sc}}(M_{sc})$, and so also $\text{variant}_{\mathcal{Q}_{sc}, \mathcal{P}_{sc}}^{\mathcal{Q}}(M_{sc})$.

By definition, for a magic variant $\text{variant}_{\mathcal{Q}, \mathcal{P}}^{\mathcal{Q}}(I)$ of an interpretation $I$ for $\mathcal{P}$, $\text{variant}_{\mathcal{Q}, \mathcal{P}}^{\mathcal{Q}}(I)|_{B_P} \subseteq I$ holds. More interesting, the magic variant of a stable model for $\mathcal{P}$ is in turn a stable model for $\text{DMS}(\mathcal{Q}, \mathcal{P})$.

Example 3.20 The magic variant of $M_{sc}$ w.r.t. $\mathcal{Q}_{sc}$ and $\mathcal{P}_{sc}$ (see Example 3.19) coincides with the interpretation $M'_{sc}$ introduced in Example 3.13. From previous examples, we know that $M_{sc}$ is a stable model of $\mathcal{P}_{sc}$, and $M'_{sc}$ is a stable model of $\text{DMS}(\mathcal{Q}_{sc}, \mathcal{P}_{sc})$. 

25
The following formalizes the intuition above and is, indeed, the counterpart of Lemma 3.16.

Lemma 3.21 For each stable model $M$ of $P$, there is a stable model $M'$ of $\text{DMS}(Q, P)$ (which is the magic variant of $M$) such that $M \supseteq M'|_{B_P}$.

Proof. Let $M' = \text{variant}^\mathcal{S}_{P}(M)$ be the magic variant of the stable model $M$, then $M \supseteq M'|_{B_P}$ holds. We first show that $M'$ is a model for $\text{Ground}(\text{DMS}(Q, P))^{M'}$. To this end, consider a rule in $\text{Ground}(\text{DMS}(Q, P))^{M'}$ having the body true, that is, a rule obtained by removing the negative body literals from a rule $r_g' \in \text{Ground}(\text{DMS}(Q, P))$ such that $B^-(r_g') \cap M = \emptyset$ and $B^+(r_g') \subseteq M'$ hold. We distinguish two cases:

1. In the case where $r_g'$ is a modified rule

$$r_g': p(\xi) \lor p_1(\xi_1) \lor \cdots \lor p_n(\xi_n) \leftarrow \text{magic}(p^o(\xi)), \text{magic}(p_1^o(\xi_1)), \ldots, \text{magic}(p_n^o(\xi_n)), q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j), \ldots, \text{not } q_1(\bar{s}_{j+1}), \ldots, \text{not } q_n(\bar{s}_n).$$

then we can apply Lemma 3.11 in order to conclude the existence of a rule $r_g \in \text{Ground}(P)$ of the form

$$r_g : p(\xi) \lor p_1(\xi_1) \lor \cdots \lor p_n(\xi_n) \leftarrow q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j),$$

$$\text{not } q_1(\bar{s}_{j+1}), \ldots, \text{not } q_n(\bar{s}_n).$$

Moreover, consider a modified rule $r' \in \text{DMS}(Q, P)$ such that $r_g' = r'\vartheta$ for a substitution $\vartheta$:

$$r' : p(\bar{\xi}) \lor p_1(\bar{\xi}_1) \lor \cdots \lor p_n(\bar{\xi}_n) \leftarrow \text{magic}(p^o(\bar{\xi})), \text{magic}(p_1^o(\bar{\xi}_1)), \ldots, \text{magic}(p_n^o(\bar{\xi}_n)), q_1(\bar{s}_1'), \ldots, q_1(\bar{s}_j'), \ldots, \text{not } q_1(\bar{s}_{j+1}'), \ldots, \text{not } q_n(\bar{s}_n').$$

and the rule $r \in P$ from which $r'$ is produced (such that $r_g = r\vartheta$):

$$r : p(\bar{\xi}) \lor p_1(\bar{\xi}_1) \lor \cdots \lor p_n(\bar{\xi}_n) \leftarrow q_1(\bar{s}_1'), \ldots, q_1(\bar{s}_j'),$$

$$\text{not } q_1(\bar{s}_{j+1}'), \ldots, \text{not } q_n(\bar{s}_n').$$

During the Generation step preceding the production of $r'$, a magic rule $r^*_i$ such that $H(r^*_i) = \{\text{magic}(q_i^o(\bar{s}_i'))\}$ and $B^+(r^*_i) \subseteq \{\text{magic}(p^o(\bar{\xi})), q_1(\bar{s}_1'), \ldots, q_1(\bar{s}_j')\}$ has been produced for each $j + 1 \leq i \leq m$ such that $q_1$ is an IDB predicate (we recall that magic rules have empty negative body). Hence, since the variables of $r^*_i$ are a subset of the variables of $r'$, the substitution $\vartheta$ can be used to map $r^*_i$ to a ground rule $r^*_{i,\vartheta} = r^*_i \vartheta$ such that $H(r^*_{i,\vartheta}) = \{\text{magic}(q_i^o(\bar{s}_i))\}$ and $B^+(r^*_{i,\vartheta}) \subseteq \{\text{magic}(p^o(\bar{\xi})), q_1(\bar{s}_1), \ldots, q_1(\bar{s}_j)\} \subseteq B^+(r_g')$. Now, since $B^+(r_g') \subseteq M'$, we can conclude
that \( B^+(r^*_{q}) \) is in turn contained in \( M' \). Thus, by construction of \( M' \), the head of \( r^*_{q} \) must be true w.r.t. \( M' \), that is, \( \text{magic}(q^d_1(\xi_1)) \in M' \)
holds for each \( j + 1 \leq i \leq m \) such that \( q_i \) is an IDB predicate. So, if some (IDB) atom \( q_1(\xi_1) \in B^-(r_g) \) belongs to \( M \), by Definition 3.18 we can conclude that \( q_1(\xi_1) \in M' \), which contradicts the assumption that \( B^-(r^*_{q}) \in M' = \emptyset \) (we recall that \( B^-(r_g) = B^-(r^*_{q}) \)). We can then safely assume that there is a rule in \( \text{Ground(\( P \))}^M \) obtained from \( r_g \) by removing the atoms in \( B^-(r_g) \).

Now, note that \( B^+(r_g) = B^+(r^*_{q}) \cap B_P \) and then, since \( M' \subseteq M'_{A_B} \) by construction of \( M' \) and \( B^+(r^*_{q}) \subseteq M' \) (by hypothesis), we can conclude that \( B^+(r_g) \subseteq M' \). Moreover, since \( M \) is a model of \( \text{Ground(\( P \))}^M \), \( B^+(r_g) \subseteq M \) implies \( H(r_g) \cap M \neq \emptyset \). Now, recall that \( H(r_g) = H(r^*_{q}) \), and hence let \( p_i(\xi_1) \) be an atom in \( H(r_g) \cap M = H(r^*_{q}) \cap M \) and \( \text{magic}(p^d_i(\xi_1)) \) be its corresponding magic atom in \( B^+(r^*_{q}) \) (we recall that \( \xi \) is the empty string). Since \( B^+(r^*_{q}) \subseteq M' \) (by hypothesis) and since \( p_i(\xi_1) \in M \), we can then conclude that \( p_i(\xi_1) \) is in \( M' \) as well by Definition 3.18.

(2) If \( r^*_{q} \) is a magic rule, then \( B^+(r^*_{q}) \subseteq M' \) implies that the (only) atom in \( H(r^*_{q}) \) belongs to \( M' \) (by Definition 3.18).

We now show that \( M' = M'_{A_B} \) is also minimal. Let \( N' \subseteq M' \) be a minimal model of \( \text{Ground(DMS(\( Q, P \)))}^{M'_{A_B}} \). We prove by induction on the definition of the magic variant that \( M' \) is in turn contained in \( N' \). The base case (i.e., \( \text{variant}(\text{DMS}(\( Q, P \)))^M \subseteq N' \)) is clearly true, since \( \text{variant}(\text{DMS}(\( Q, P \)))^M \) contains only the query seed, which is a fact in \( \text{Ground(DMS(\( Q, P \)))}^{M'_{A_B}} \). Suppose \( \text{variant}(\text{DMS}(\( Q, P \)))^M \subseteq N' \) in order to prove that \( \text{variant}(\text{DMS}(\( Q, P \)))^M \subseteq N' \) holds as well.

While considering an atom in \( \text{variant}(\text{DMS}(\( Q, P \)))^M \setminus \text{variant}(\text{DMS}(\( Q, P \)))^M \), we distinguish two cases:

(a) For a magic atom \( \text{magic}(p^d(\xi)) \) in \( \text{variant}(\text{DMS}(\( Q, P \)))^M \setminus \text{variant}(\text{DMS}(\( Q, P \)))^M \), by Definition 3.18 there must be a rule \( r^*_{q} \in \text{Ground(DMS(\( Q, P \)))}^M \) having \( r^*_{q} \subseteq \text{variant}(\text{DMS}(\( Q, P \)))^M \) (we recall that magic rules have empty negative body and so \( r^*_{q} \in \text{Ground(DMS(\( Q, P \)))}^M \) holds). We can then conclude that \( B^+(r^*_{q}) \subseteq N' \) holds by the induction hypothesis and so \( \text{magic}(p^d(\xi)) \subseteq N' \) (because \( N' \) is a model of \( \text{Ground(DMS(\( Q, P \)))}^{M'_{A_B}} \)).

(b) For a standard atom \( p(\xi) \) in \( \text{variant}(\text{DMS}(\( Q, P \)))^M \setminus \text{variant}(\text{DMS}(\( Q, P \)))^M \), by Definition 3.18 there is a binding \( \alpha \) such that \( \text{magic}(p^d(\xi)) \subseteq \text{variant}(\text{DMS}(\( Q, P \)))^M \) and the atom \( p(\xi) \) belongs to \( M \). Assume for the sake of contradiction that \( p(\xi) \notin N' \). Since \( M' \) is a model of \( \text{DMS}(\( Q, P \)) \) and \( N' \) is a model of \( \text{Ground(DMS(\( Q, P \)))}^{M'_{A_B}} \), we can compute the set \( \text{killed}(\text{DMS}(\( Q, P \)))^{M'_{A_B}} \) as introduced in Section 3.3.1 and note, in particular, that \( p(\xi) \in \text{killed}(\text{DMS}(\( Q, P \)))^{M'_{A_B}} \) holds (by definition). Moreover, by Proposition 3.15, \( \text{killed}(\text{DMS}(\( Q, P \)))^{M'_{A_B}} \)
is an unfounded set for \( \mathcal{P} \) w.r.t. \( M'|_{B_{\mathcal{P}}} \). In addition, \( M \supseteq M'|_{B_{\mathcal{P}}} \) holds by Definition 3.18. Thus, \( M \) is a stable model for \( \mathcal{P} \) such that \( M \supseteq M'|_{B_{\mathcal{P}}} \), and we can hence apply Theorem 3.9 in order to conclude that \( M \cap \text{killed}_{Q,\mathcal{P}}^{M'}(N') = \emptyset \). The latter is in contradiction with \( p(\bar{e}) \in \text{killed}_{Q,\mathcal{P}}^{M'}(N') \) and \( p(\bar{e}) \in M \). Hence, \( p(\bar{e}) \in N' \).

\[ \square \]

We can then prove the correspondence of stable models with respect to queries.

**Theorem 3.22** Let \( Q \) be a query for a Datalog\(^{\text{\neg\neg}}\) program \( \mathcal{P} \). Then, for each stable model \( M \) of \( \mathcal{P} \), there is a stable model \( M' \) of \( \text{DMS}(Q, \mathcal{P}) \) (which is the magic variant of \( M \)) such that \( M'|_{Q} = M|_{Q} \).

**Proof.** Let \( M \) be a stable model of \( \mathcal{P} \) and \( M' = \text{variant}_{Q,\mathcal{P}}(M) \) its magic variant. Because of Lemma 3.21, \( M' \) is a stable model of \( \text{DMS}(Q, \mathcal{P}) \) such that \( M \supseteq M'|_{B_{\mathcal{P}}} \). Thus, we trivially have that \( M|_{Q} \supseteq M'|_{Q} \) holds. We now show the reverse inclusion.

Since \( M' \) is a stable model of \( \text{DMS}(Q, \mathcal{P}) \), we can determine the set \( \text{killed}_{Q,\mathcal{P}}^{M'}(M') \) as defined in Section 3.3.1. Hence, by Definition 3.12 we can conclude that (a) \( B_{\mathcal{P}}|_{Q} \setminus M' \subseteq \text{killed}_{Q,\mathcal{P}}^{M'}(M') \) because \( M' \) contains the magic seed by construction (we recall that \( B_{\mathcal{P}}|_{Q} \) denotes the ground instances of \( Q \)). Moreover, since \( M \) is a stable model of \( \mathcal{P} \) with \( M \supseteq M'|_{B_{\mathcal{P}}} \) and \( \text{killed}_{Q,\mathcal{P}}^{M'}(M') \) is an unfounded set for \( \mathcal{P} \) w.r.t. \( M'|_{B_{\mathcal{P}}} \) (by Proposition 3.15), by Theorem 3.9 we can conclude that (b) \( M \cap \text{killed}_{Q,\mathcal{P}}^{M'}(M') = \emptyset \). Thus, by combining (a) and (b) we obtain that \( (B_{\mathcal{P}}|_{Q} \setminus M') \cap M = \emptyset \), which is equivalent to \( M|_{Q} \subseteq M'|_{Q} \).

\[ \square \]

From the above theorem, together with Theorem 3.17, the correctness of the Magic Set method with respect to query answering directly follows.

**Corollary 3.23** Let \( \mathcal{P} \) be a Datalog\(^{\text{\neg\neg}}\) program, and let \( Q \) be a query. Then \( \text{DMS}(Q, \mathcal{P}) \equiv_{Q}^{b} \mathcal{P} \) and \( \text{DMS}(Q, \mathcal{P}) \equiv_{Q}^{b} \mathcal{P} \) hold.

**Proof.** We want to show that, for any set of facts \( \mathcal{F} \) defined over the EDB predicates of \( \mathcal{P} \) (and \( \text{DMS}(Q, \mathcal{P}) \)), \( \text{Ans}_{b}(Q, \text{DMS}(Q, \mathcal{P}) \cup \mathcal{F}) = \text{Ans}_{b}(Q, \mathcal{P} \cup \mathcal{F}) \) and \( \text{Ans}_{c}(Q, \text{DMS}(Q, \mathcal{P}) \cup \mathcal{F}) = \text{Ans}_{c}(Q, \mathcal{P} \cup \mathcal{F}) \) hold. We first observe that the Magic Set rewriting does not depend on EDB facts; thus, \( \text{DMS}(Q, \mathcal{P}) \cup \mathcal{F} = \text{DMS}(Q, \mathcal{P} \cup \mathcal{F}) \) holds. Therefore, as a direct consequence of Theorem 3.17 and Theorem 3.22, we can conclude \( \text{Ans}_{b}(Q, \text{DMS}(Q, \mathcal{P}) \cup \mathcal{F}) = \text{Ans}_{b}(Q, \mathcal{P} \cup \mathcal{F}) \) and \( \text{Ans}_{c}(Q, \text{DMS}(Q, \mathcal{P} \cup \mathcal{F})) = \text{Ans}_{c}(Q, \mathcal{P} \cup \mathcal{F}) \).

\[ \square \]

### 3.4 Magic Sets for Stratified Datalog Programs without Disjunction

Stratified Datalog programs without disjunction have exactly one stable model [23]. However, the Magic Set transformation can introduce new dependencies
between atoms, possibly resulting in unstratified programs (we refer to the analysis in [30]). Clearly, original and rewritten programs agree on the query, as proved in the previous section, but the question whether the rewritten program admits a unique stable model is also important. In fact, for programs having the unique stable model property, brave and cautious reasoning coincide and a solver can immediately answer the query after the first (and unique) stable model is found. The following theorem states that the rewritten program of a stratified program indeed has a unique stable model.

**Theorem 3.24** Let $\mathcal{P}$ be a disjunction-free Datalog program with stratified negation and $Q$ a query. Then $\text{DMS}(Q, \mathcal{P})$ has a unique stable model.

**Proof.** Let $M$ be the unique stable model of $\mathcal{P}$, and $M' = \text{variant}_Q^\mathcal{P}(M)$ its magic variant as presented in Definition 3.18. By Lemma 3.21 we already know that $M'$ is a stable model of $\text{DMS}(Q, \mathcal{P})$. We now show that any stable model $N'$ of $\text{DMS}(Q, \mathcal{P})$ contains $M'$ by induction on the structure of $M'$. The base case ($\text{variant}_Q^0(M) \subseteq N'$) is clearly true, since $\text{variant}_Q^0(M)$ contains only the query seed, which is a fact in $\text{DMS}(Q, \mathcal{P})$ and so must belong also to $N'$. Suppose $\text{variant}_Q^i(M) \subseteq N'$ in order to prove that $\text{variant}_Q^{i+1}(M) \subseteq N'$ holds as well. Thus, while considering an atom in $\text{variant}_Q^{i+1}(M) \setminus \text{variant}_Q^i(M)$, two cases are possible:

1. For a magic atom $\text{magic}(p^\alpha(\bar{t}))$ in $\text{variant}_Q^{i+1}(M) \setminus \text{variant}_Q^i(M)$, by Definition 3.18 there must be a rule $r_g^\alpha \in \text{Ground}(\text{DMS}(Q, \mathcal{P}))$ having $H(r_g^\alpha) = \{\text{magic}(p^\alpha(\bar{t}))\}$ and $B^+(r_g^\alpha) \subseteq \text{variant}_Q^i(M)$ (we recall that magic rules have empty negative bodies and so $r_g^\alpha \in \text{Ground}(\text{DMS}(Q, \mathcal{P}))^{N'}$ holds). We can then conclude that $B^+(r_g^\alpha) \subseteq N'$ holds by the induction hypothesis and so $\text{magic}(p^\alpha(\bar{t})) \in N'$ (because $N'$ is a model of $\text{Ground}(\text{DMS}(Q, \mathcal{P}))^{N'}$).

2. For a standard atom $p(\bar{t})$ in $\text{variant}_Q^{i+1}(M) \setminus \text{variant}_Q^i(M)$, by Definition 3.18 there is a binding $\alpha$ such that $\text{magic}(p^\alpha(\bar{t})) \in \text{variant}_Q^i(M)$ and the atom $p(\bar{t})$ belongs to $M$. Assume for the sake of contradiction that $p(\bar{t}) \notin N'$. Since $N'$ is a stable model of $\text{DMS}(Q, \mathcal{P})$, we can compute the set $\text{killed}_{Q, \mathcal{P}}^\mathcal{P}(N')$ as introduced in Section 3.3.1 and note, in particular, that $p(\bar{t}) \in \text{killed}_{Q, \mathcal{P}}^\mathcal{P}(N')$ holds, by definition. Moreover, by Proposition 3.15, $\text{killed}_{Q, \mathcal{P}}^\mathcal{P}(N')$ is an unfounded set for $\mathcal{P}$ w.r.t. $N'[B_p]$. In addition, by Lemma 3.22 there is a stable model $N$ of $\mathcal{P}$ such that $N \supseteq N'[B_p]$, which would mean that $p(\bar{t}) \notin N$ holds. Hence, we can conclude that $N$ and $M$ are two different stable models of $\mathcal{P}$, obtaining a contradiction, as $\mathcal{P}$ has a unique answer set.

Since stable models are incomparable w.r.t. containment, $M' \subseteq N'$ implies $M' = N'$. Hence, $M'$ is the unique stable model of $\text{DMS}(Q, \mathcal{P})$. \hfill $\square$
4 Implementation

The Disjunctive Magic Set method (DMS) has been implemented and integrated into the core of the DLV [33] system. In this section, we shall first provide a few details on the implementation issues and on the system usage. Then we briefly present our strategy for eliminating redundant rules that might be introduced during the Magic Set rewriting.

4.1 System Architecture and Usage

We have created a prototype system by implementing our Magic Set technique in the core of DLV, as shown in the architecture reported in Figure 2. The original system DLV supports both brave and cautious semantics, and for a completely ground query it can be also used for computing all the stable models in which the query is true. DLV performs brave reasoning if invoked with the command-line option -FB, while -FC indicates cautious reasoning.

In our prototype, the DMS algorithm is applied automatically by default when the user invokes DLV with -FB or -FC with a (partially) bound query. Magic Sets are not applied by default if the query is free, that is, if it does not contain any constant. The user can modify this default behaviour by specifying the command-line options -ODMS (for applying Magic Sets) or -ODMS- (for not applying magic sets).

If a completely bound query is specified, DLV can print out the magic variant of the stable model witnessing its truth (for brave reasoning) or its falsity (for cautious reasoning), by specifying the command-line option --print-model.

Within DLV, DMS is applied immediately after parsing the program and the query by the Magic Set Rewriter module. The rewritten (and optimized) program is then processed by the Intelligent Grounding module. Finally, the Model Generator module performs a backtracking algorithm and the answer is printed out.

The interested reader can retrieve a downloadable executable of the DLV system supporting the Magic Set optimization from http://www.dlvsystem.com/magic/.

4.2 Dealing with Redundant Rules

Even though our rewriting algorithm keeps the amount of generated rules low, it might happen that some redundant rules are generated when adorning disjunctive rules, thereby somewhat deteriorating the optimization effort. For instance, in Example 3.6 the first two modified rules coincide, and this might happen even if the two head predicates differ. In general not only duplicated rules might be created, but also rules which are logically subsumed by other rules in the program. Let us first state the definition of subsumption for Datalog rules.

Definition 4.1 Let $\mathcal{P}$ be a disjunctive Datalog program, and let $r$ and $r'$ be two rules of $\mathcal{P}$. Then, $r$ is subsumed by $r'$ (denoted by $r \subseteq r'$) if there exists a
Figure 2: Prototype system architecture.
substitution $\vartheta$, over the variables of $r'$, such that $H(r')\vartheta \subseteq H(r)$ and $B(r')\vartheta \subseteq B(r)$. Moreover, a rule $r$ is redundant if there exists a rule $r'$ such that $r \subseteq r'$.

Ideally, a Magic Set rewriting algorithm should be capable of identifying all the possible redundant rules and removing them from the output. Unfortunately, this approach is unlikely to be feasible in polynomial time, given that checking for redundancy precisely coincides with the problem of finding a homomorphism between two relational structures, which is NP-complete.

Thus, in order to identify whether a rule $r$ produced during the Magic Set transformation is redundant, we pragmatically apply a greedy subsumption algorithm in our implementation, for checking whether $r \sqsubseteq r'$ holds for some rule $r'$. In particular, the employed heuristics aims at building the substitution $\vartheta$ (as in Definition 4.1) by iteratively choosing an atom $p(\bar{t})$ (which is not yet processed) from $r'$ and by matching it (if possible) with some atom of $r$. The greedy approach prefers those atoms of $r'$ with the maximum number of variables not yet matched.

To turn on subsumption checking (applied once after the Magic Set rewriting), DLV should be invoked with the command-line option `-O DMS+`.

5 Experimental Results on Literature Benchmarks

We performed experimental activity aimed at assessing the efficacy of the proposed technique. In this section we present the results obtained on various benchmarks, most of which are taken from the literature, while further experiments on a real application scenario will be discussed in detail in Section 6.

5.1 Compared Methods, Benchmark Problems and Data

In order to evaluate the impact of the proposed method, we have compared DMS (and its variant with subsumption checking, DMS+) both with the traditional DLV evaluation without Magic Sets and with the APM method proposed in [26]. For the comparison, we consider the following benchmark problems that have been already used to assess APM in [26] (see therein for more details):

- **Simple Path:** Given a directed graph $G$ and two nodes $a$ and $b$, does there exist a unique path connecting $a$ to $b$ in $G$? The problem can be encoded by the program

  \[
  \begin{align*}
  \text{sp}(X, X) \lor \text{not}_\text{sp}(X, X) & : = \text{edge}(X, Y). \\
  \text{sp}(X, Y) \lor \text{not}_\text{sp}(X, Y) & : = \text{sp}(X, Z), \text{edge}(Z, Y). \\
  \text{path}(X, Y) & : = \text{sp}(X, Y). \\
  \text{path}(X, Y) & : = \text{not}_\text{sp}(X, Y). \\
  \text{not}_\text{sp}(X, Z) & : = \text{path}(X, Y_1), \text{path}(X, Y_2), Y_1 <> Y_2, \text{edge}(Y_1, Z), \text{edge}(Y_2, Z).
  \end{align*}
  \]
with the query $\text{sp}(a,b)$?. The structure of the graph, which is the same as the one reported in [26], consists of a square matrix of nodes connected as shown in Figure 3, and the instances are generated at the varying of the number of nodes.

• **Related:** Given a genealogy graph storing information of relationship (father/brother) among people and given two persons $p_1$ and $p_2$, is $p_1$ an ancestor of $p_2$? The problem can be encoded by the program

$$\text{father}(X,Y) \lor \text{brother}(X,Y) : \neg \text{related}(X,Y).$$
$$\text{ancestor}(X,Y) : \neg \text{father}(X,Y).$$
$$\text{ancestor}(X,Y) : \neg \text{father}(X,Z), \text{ancestor}(Z,Y).$$

and the query is $\text{ancestor}(p_1,p_2)$?. The structure of the “genealogy” graph is the same as the one presented in [26] and coincides with the one used for testing Simple Path. The instances are generated at the varying of the number of nodes, i.e., the number of persons, of such graph.

• **Strategic Companies:** We are given a collection $C$ of companies producing some goods in a set $G$, such that each company $c_i \in C$ is controlled by a set of other companies $O_i \subseteq C$. A subset of the companies $C' \subseteq C$ is a **strategic set** if it is a minimal set of companies producing all the goods in $G$, such that if $O_i \subseteq C'$ for some $i = 1, \ldots, m$ then $c_i \in C'$ must hold. Given two distinct companies $c_i, c_j \in C$, do $c_i$ and $c_j$ belong to some strategic set of $C$? The problem can be encoded by the program

$$\text{st}(C_i) \lor \text{st}(C_2) \lor \text{st}(C_3) \lor \text{st}(C_4) : \neg \text{produced}_\text{by}(P, C_1, C_2, C_3, C_4).$$
$$\text{st}(C) : \neg \text{controlled}_\text{by}(C, C_1, C_2, C_3, C_4), \text{st}(C_1), \text{st}(C_2), \text{st}(C_3), \text{st}(C_4).$$

with the query $\text{st}(c_i)$, $\text{st}(c_j)$? (the definition of query and query answering can be extended from a single atom to a conjunction of atoms in
an obvious way). For this benchmark we used the instances submitted for the Second Answer Set Competition\(^5\).

In addition, we have performed further experiments on a real application scenario pertaining the problem of answering user queries in data integration systems. These latter experiments will be discussed in more detail in Section 6.

5.2 Results and Discussion

The experiments have been performed on a 3GHz Intel\(^\text{R}\) Xeon\(^\text{R}\) processor system with 4GB RAM under the Debian 4.0 operating system with GNU/Linux 2.6.23 kernel. The DLV prototype used has been compiled with GCC 4.3.3. For every instance, we have allowed a maximum running time of 600 seconds (10 minutes) and a maximum memory usage of 3GB.

On all problems, DMS outperforms APM, even without considering the time for the rewriting needed in [26], which is also not reported in the figures.

The results for Simple Path are reported in Figure 4. On this problem, DMS scales much better than APM, as shown in the left diagram, and the additional speed-up provided by DMS\(^+\) can be appreciated in the right diagram (in which y-axis has logarithmic scale).

In Figure 5 we report the average execution times for Related, showing the advantages of DMS over APM. In this case the performance of our two Magic Set methods are almost the same (see the right diagram). We have not included DLV without optimization since it solves only the first instances in the allowed time.

In Figure 6 we report the results for Strategic Companies. The advantages of the Magic Set methods are strongly evident: DLV without optimization solves only two instances. For this benchmark DMS\(^+\) and APM produce exactly the same ground program, and for this reason they are equivalently efficient, while the execution of DMS is slowed down by the presence of redundant rules.

We now conclude the discussion about results of experimental activity conducted on the Magic Set method, by discussing an important application of this optimization technique in the context of data integration systems.

### 6 Application to Data Integration

#### 6.1 Data Integration Systems in a Nutshell

In recent years, the database community has renewed the interest in (disjunctive) logic programs, in the light of the various efforts demonstrating the usefulness of these formalisms with respect to computing consistent answers in data integration systems.

We briefly recall here that data integration systems offer transparent access to heterogeneous sources by providing users with the so-called global schema, which users can query in order to extract data relevant for their aims. Then, the systems are in charge of accessing each source separately, and combining local results into the global answer, by using a set of mapping assertions, specifying
the relationship between the sources and the global schema. Following [31], we formalize a data integration system \( \mathcal{I} \) as a triple \( \langle G, S, M \rangle \), where:

1. \( G \) is the global (relational) schema, i.e., a pair \( \langle \Psi, \Sigma \rangle \), where \( \Psi \) is a finite set of relation symbols, each with an associated positive arity, and \( \Sigma \) is a finite set of integrity constraints (ICs) expressed on the symbols in \( \Psi \), i.e., first-order assertions that are intended to be satisfied by database instances.

2. \( S \) is the source schema, constituted by the schemas of the various sources that are part of the data integration system. We assume that \( S \) is a relational schema of the form \( S = \langle \Psi', \emptyset \rangle \), i.e., there are no integrity constraints on the sources. This assumption implies that data stored at the sources are locally consistent; this is a common assumption in data integration, because sources are in general external to the integration system, which is not in charge of analyzing or restoring their consistency.

3. \( M \) is the mapping which establishes the relationship between \( G \) and \( S \). In our framework, the mapping follows the GAV approach, i.e., each global relation is associated with a view—a Datalog \( \lor, \neg \) query over the sources.

The main semantical issue in data integration systems is that, since integrated sources are originally autonomous, their data, filtered through the mapping, are likely not to satisfy the constraints on the global schema. An approach to remedy to this problem that has lately received a lot of interest in the literature (see, e.g., [14, 11, 10, 9, 17, 16, 21, 22, 1, 15]) is based on the notion of repair for an inconsistent database as introduced in [2]. Roughly speaking, a repair of a database is a new database that satisfies the constraints in the schema, and minimally differs from the original one. Since multiple repairs might be singled out for an inconsistent database, the standard approach in answering user queries is to compute those answers that are true in every possible repair, called consistent answers in the literature.

Clearly enough, there is a precise correspondence between the concept of consistent answers and the cautious consequences of logic programs. In fact, various authors [3, 24, 5, 14, 11, 15] considered the idea of encoding the constraints of the global schema \( G \) into (various kinds of) Datalog programs, such that the stable models of this program yield the repairs of the database retrieved from the sources. For instance, approaches mapping an integration system to Datalog programs under unstratified negations are in [14], whereas disjunctive Datalog programs (also enriched with unstratified negation) have been considered in [12].

Magic Sets play then a crucial role in this context, since by boosting performances of cautious reasoning, they immediately serve as a mean to optimize query evaluation over inconsistent databases. In particular, the benefits of Magic Sets in the context of optimizing logic programs with unstratified negation have been discussed in [20], whereas the benefits of Magic Sets over encodings based on disjunctive programs enriched with unstratified negation have been assessed.
in [12], with various experiments specifically designed to deal with data integration systems.

Below, we consider instead a “pure” disjunctive encoding, and experiment the benefit of Magic Sets over it. This rewriting has been considered within the EU project on data integration INFOMIX [32].

6.2 A Disjunctive Encoding for Consistent Query Answering

Let $I = (G, S, M)$ be a data integration system where $G = (\Psi, \Sigma)$, and let $D$ be a database for $G$, which is simply viewed in the following as a set of facts over the relational predicates in $G$. We assume that constraints over the global schema are keys and exclusion dependencies. In particular, we recall here that a set of attributes $\bar{x}$ is a key for the relation $r$ if:

\[ r(\bar{x}, \bar{y}) \land r(\bar{x}, \bar{z}) \rightarrow \bar{y} = \bar{z}, \quad \forall \{r(\bar{x}, \bar{y}), r(\bar{x}, \bar{z})\} \subseteq D \]

and that an exclusion dependency holds between relations $r$ and $s$ over the set of attributes $\bar{x}$ if:

\[ r(\bar{x}, \bar{y}) \cap s(\bar{x}, \bar{z}) = \emptyset, \quad \forall \{r(\bar{x}, \bar{y}), s(\bar{x}, \bar{z})\} \subseteq D \]

Then, the disjunctive rewriting of $q$ w.r.t. $I$ is the Datalog\$^{\lor,\neg}$ program $\Pi(I) = \Pi_{KD} \cup \Pi_{ED} \cup \Pi_{M} \cup \Pi_{coll}$ where:

- For each relation $r$ in $G$ and for each key defined over its set of attributes $\bar{x}$, $\Pi_{KD}$ contains the rules:

  \[ r_{out}(\bar{x}, \bar{y}) \lor r_{out}(\bar{x}, \bar{z}) := r_{D}(\bar{x}, \bar{y}), r_{D}(\bar{x}, \bar{z}), Y_1 \neq Z_1. \]

  \[ \cdots \]

  \[ r_{out}(\bar{x}, \bar{y}) \lor r_{out}(\bar{x}, \bar{z}) := r_{D}(\bar{x}, \bar{y}), r_{D}(\bar{x}, \bar{z}), Y_m \neq Z_m. \]

  where $\bar{y} = Y_1, \ldots, Y_m$, and $\bar{z} = Z_1, \ldots, Z_m$.

- For each exclusion dependency between relations $r$ and $s$ over the set of attributes $\bar{x}$, $\Pi_{ED}$ contains the rule:

  \[ r_{out}(\bar{x}, \bar{y}) \lor s_{out}(\bar{x}, \bar{z}) := r_{D}(\bar{x}, \bar{y}), s_{D}(\bar{x}, \bar{z}). \]

- For each relation $r$ in $G$, $\Pi_{coll}$ contains the rule:

  \[ r(\bar{w}) := r_{D}(\bar{w}), \text{ not } r_{out}(\bar{w}). \]

- For each Datalog rule $r$ in $M$ such that:

  \[ r : k(\bar{t}) := q_1(\bar{s}_1), \ldots, q_m(\bar{s}_m). \]
where $k$ is a relation in $\mathcal{G}$ and $q_i$ (for $1 \leq i \leq m$) is a relation in $\mathcal{S}$, $\Pi_M$ contains the rule:

$$r : \ k \left( \bar{t} \right) \leftarrow q_1(\bar{s}_1), \ldots, q_m(\bar{s}_m).$$

Within the INFOMIX project, the above encoding has been used after noticing that for each user query $Q$ (over $\mathcal{G}$), and for each source database $F$ (over $\mathcal{S}$), consistent query answers to $Q$ precisely coincide with the set $\text{Ans}_c(Q, \Pi(I) \cup F)$. In particular, within INFOMIX, arbitrary conjunctive queries have been considered, since their definition can be easily mapped in $\Pi(I)$. Also, inclusion dependencies have been considered according to the rewriting discussed in [14], whose details are omitted here.

To our ends, it is however important to notice that the program $\Pi(I)$ is a stratified disjunctive program, and hence the Magic Set method defined in this paper can be applied. In particular, the effectiveness of the Magic Set method in this crucial application context has then been assessed via a number of experiments carried out the demonstration scenario of the INFOMIX project, which refers to the information system of the University “La Sapienza” in Rome.

The global schema consists of 14 global relations with 29 constraints, while the source schema includes 29 relations (in 3 legacy databases) and 12 web wrappers (generating relational data) for more than 24MB of data regarding students, professors and exams in any faculty of the University.

On this schema, some (5) typical queries with peculiar characteristics are formulated, which model different use cases. In particular, we measured the average execution time of DLV computing $\text{Ans}_c(Q, \Pi(I) \cup F)$ and $\text{Ans}_c(Q, \text{DMS}^+(Q, \Pi(I)) \cup F)$ on dataset of increasing size. All results have been collected running the INFOMIX prototype system on a 3GHz Intel® Xeon® processor system with 4GB RAM under the Debian 4.0 operating system with GNU/Linux 2.6.23 kernel. The DLV prototype used has been compiled with GCC 4.3.3. For every instance, we have allowed a maximum running time of 600 seconds (10 minutes) and a maximum memory usage of 3GB.

The results, reported in Figure 7, confirm that on various practical queries the performance is greatly improved by Magic Sets, while only minor overhead is introduced if no propagation occurs (e.g., Query 5). We point out that the evaluation time under the Magic Set technique nicely scales and dramatically improves the evaluation time without optimization (see Query 1 to 4 in Figure 7).

7 Conclusion

The Magic Set method is one of the most well-known techniques for the optimization of positive recursive Datalog programs due to its efficiency and its generality, even though other focused methods such as the supplementary magic set and other special techniques for linear and chain queries have been proposed as well (see, e.g., [27, 41, 38]). After seminal papers [4, 7], the viability of the
Figure 7: Average execution time of query evaluation in the INFOMIX Demo Scenario.

approach was demonstrated e.g. in [28, 37]. Later, extensions and refinements have been proposed, addressing e.g. query constraints in [40], the well-founded semantics in [30], or integration into cost-based query optimization in [39]. The research on variations of the Magic Set method is still going on. For instance, in [20] an extensions of the Magic Set method is discussed for the class of unstratified logic programs (without disjunction). In [8] a technique for the class of soft-stratifiable programs is given. And, in [26] an elaborated technique for disjunctive programs based on [25] is described.

In this paper, we have elaborated on the issues addressed in [25, 26]. Our approach is similar in spirit to APM, but differs in several respects:

- **DMS** is a dynamic optimization of the query answering, in the sense that in addition to the optimization of the grounding process, which is the only optimization performed by APM, DMS can drive the model generation phase by dynamically disabling parts of the program that became irrelevant in the processed model.

- **DMS** has a strong relationship with unfounded sets, allowing for a clean application to disjunctive Datalog programs also in presence of stratified
negation.

- **DMS** can be further improved by performing a downstream subsumption checking.

- **DMS** has been integrated into the DLV system [33], profitably exploiting the DLV internal data-structures and the ability of controlling the grounding module.

Experimental activity has been conducted on several benchmarks, many of which taken from the literature. The results of our experimentation evidenced that our implementation outperforms **APM** in general, often by an exponential factor. This is mainly due to the optimization of the model generation phase, which is specific of our Magic Set technique. In addition, further experimental activity has been conducted on a real application scenario, which evidenced that Magic Sets may play a crucial role in optimizing consistent query answering over inconsistent databases. Importantly, other authors have already recognized the benefits of our optimization strategies w.r.t. this very important application domain [12], thereby confirming the validity and the robustness of the work discussed in this paper.

We conclude by observing that it has been noted in the literature (e.g. in [30]) that in the non-disjunctive case, *memoing* techniques lead to similar computations as evaluations after Magic Set transformations. Also, in the disjunctive case such techniques have been proposed (e.g. Hyper Tableaux [6]), for which similar relations might hold. While [30] has already evidenced that an advantage of Magic Sets over such methods is that the latter may be more easily combined with other optimization techniques, we believe that achieving a deeper comprehension of the relationships among these techniques constitutes an interesting avenue for further research.

References


