A simple correctness proof for magic transformation

Wlodzimierz Drabent

Original Publication:

Wlodzimierz Drabent , A simple correctness proof for magic transformation, 2012, Theory and Practice of Logic Programming, (12), 6, 929-936.
http://dx.doi.org/10.1017/S1471068411000032

TLP **12** (6): 929–936, 2012.
doi:10.1017/S1471068411000032 First published online 4 March 2011

A simple correctness proof for magic transformation

WŁODZIMIERZ DRABENT

Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, Pl – 01-237 Warszawa, Poland
and
Department of Computer and Information Science, Linköpings Universitet, S – 581 83 Linköping, Sweden
(*e-mail:* drabent@ipipan.waw.pl)

submitted 21 December 2009; revised 14 August 2010; accepted 11 January 2011

Abstract

The paper presents a simple and concise proof of correctness of the magic transformation. We believe that it may provide a useful example of formal reasoning about logic programs. The correctness property concerns the declarative semantics. The proof, however, refers to the operational semantics (LD-resolution) of the source programs. Its conciseness is due to applying a suitable proof method.

KEYWORDS: program correctness, logic programming, magic transformation, declarative semantics, LD-resolution, operational semantics

1 Introduction

Magic transformation (see Nilsson and Małuszyński 1995, Ch. 15.3 for references) is a technique to facilitate efficient bottom-up evaluation of logic programs. Given a program and an initial goal, the transformation produces the so-called magic program; the answers of both programs for the initial goal should be the same. Looking for a correctness proof of magic transformation, the author found that such a proof was rather easy to construct. Moreover, the result turned out to be surprisingly concise. In this paper the author presents the proof with all the details, and believes that it provides a useful example of formal reasoning about logic programs.

Mascellani and Pedreschi (2002) stated that "all known proofs of correctness of the magic-set transformation(s) are rather complicated" (see Ramakrishnan 1991, for an example), and presented a simpler proof, which concerns the declarative semantics of the original and transformed programs. Our proof might be even more simpler; moreover, it formalizes the relation between the declarative semantics of the transformed program and the operational semantics of the original one. The simplification is due to applying a suitable proof method for program correctness, instead of constructing a proof from scratch.

2 Preliminaries

For standard notions and notation, see (Apt 1997). We consider definite clause programs (not restricted to Datalog). By a *query*, we mean a conjunction of atoms. Given a program P, by an *answer* (or *correct answer*), we mean any query Q that is a logical consequence of the program ($P \models Q$). If an answer is an instance of some initial query Q_0, then we say that it is an answer for P and Q_0. By a *computed answer* for a program P and initial query Q_0, we mean an instance $Q_0\theta$ of Q_0, produced by a successful SLD-derivation for P and Q_0.[1] A fundamental theorem relates answers and computed answers.

Theorem 1 (*Soundness and completeness of SLD-resolution*)
For any program P, any query Q, and any selection rule:

If Q is a computed answer for P, then $P \models Q$.

If $P \models Q\theta$, then there exists a computed answer $Q\sigma$ for P and Q such that $Q\theta$ is an instance of $Q\sigma$.[2]

A *proof tree* (sometimes called implication tree or derivation tree) for a program P and an atomic query A is a finite tree whose nodes are atoms, the root is A, and in which if $B_1, \ldots, B_n$ ($n \geqslant 0$) are the children of a node H, then $H \leftarrow B_1, \ldots, B_n$ is an instance of a clause of P. Proof trees provide a useful characterization of logic program answers.

Theorem 2
For any program P and query Q, $P \models Q$ iff for each atom A of Q there exists a proof tree for P and A.

The theorem follows immediately from its version for atomic queries (Clark 1979), see, e.g. (Apt 1997, Th. 4.24(v)), or (Deransart 1993, Prop. 2.6).

We focus on LD-resolution (SLD-resolution with the Prolog selection rule) and study the sets of procedure calls and procedure successes in LD-derivations. The procedure calls are the atoms selected in the derivation. A definition of procedure successes is given in Appendix A. For the proof of the main theorem of this paper it is sufficient to know that any computed answer for an initial atomic query is a procedure success.

Consider a pair $\langle pre, post \rangle$ of sets of atoms, each closed under substitution. We can treat such a pair as a specification of procedure calls and successes of a program (a *call-success specification*).

Definition 1
We say that a program P with a query Q is *correct* with respect to a call-success specification $\langle pre, post \rangle$ iff in any LD-derivation for P and Q all the procedure calls are in *pre* and all the successes are in *post*.

[1] In (Apt 1997) answers are also called correct instances of initial queries, and computed answers are called computed instances.

[2] For a proof see, e.g. (Apt 1997, Ths. 4.4, 4.13).

Note that such correctness is not a declarative property, as it depends on a particular operational semantics. We will use the following sufficient criterion for correctness (Drabent and Miłkowska 2005). (See Concluding Remarks for discussion and references, and Appendix A for a proof.)

Theorem 3
Assume that for a call-success specification $\langle pre, post \rangle$, a program P, and an atomic query $Q \in pre$ the following holds:

For each (possibly nonground) instance $H \leftarrow B_1, \ldots, B_n$ $(n \geqslant 0)$ of each clause of P

$$\begin{aligned} &\text{if } H \in pre,\ B_1, \ldots, B_n \in post, \text{ then } H \in post, \\ &\text{if } H \in pre,\ B_1, \ldots, B_{i-1} \in post, \text{ then } B_i \in pre \text{ (for } i = 1, \ldots, n). \end{aligned} \tag{1}$$

Then P with Q is correct with respect to $\langle pre, post \rangle$.

For a nonatomic initial query, the requirement $Q \in pre$ has to be generalized to the following: For each instance $B_1, \ldots, B_n$ $(n > 0)$ of the query, if $B_1, \ldots, B_{i-1} \in post$ then $B_i \in pre$ (for $i = 1, \ldots, n$).

It remains to define the magic transformation. It adds new predicate symbols to the alphabet $\mathscr{L}$ of programs and queries; for each predicate symbol p, the unique new symbol $^\bullet p$ is added. In a simple version, for instance (Nilsson and Małuszyński 1995), the arity of $^\bullet p$ is that of p. In the general case, some k_p argument positions of p are selected, and the arity of $^\bullet p$ is k_p. (We do not discuss the choice of k_p and that of the selected positions, as it is irrelevant for the correctness of magic transformation.) Let $^\bullet\mathscr{P}$ denote the set of new predicate symbols. If $A = p(t_1, \ldots, t_n)$ is an atom over $\mathscr{L}$, then $^\bullet A$ denotes $^\bullet p(t_{i_1}, \ldots, t_{i_{k_p}})$, where $i_1, \ldots, i_{k_p}$ are the selected positions of p. Such an $^\bullet A$ is called *magic template*. In what follows A, B, H, possibly with subscripts, denote atoms over $\mathscr{L}$ (hence, $^\bullet A, ^\bullet B, ^\bullet H$ stand for atoms with the new predicate symbols).

Definition 2 (*Magic transformation*)
Let P be a program and Q be an atomic query. The magic program $magic(P, Q)$ for P and Q is the program containing

(1) a clause $H \leftarrow {}^\bullet H, B_1, \ldots, B_n$ for each clause $H \leftarrow B_1, \ldots, B_n$ in P;
(2) a clause $^\bullet B_i \leftarrow {}^\bullet H, B_1, \ldots, B_{i-1}$ for each clause $H \leftarrow B_1, \ldots, B_n$ in P and each $i = 1, \ldots, n$;
(3) the clause $^\bullet Q \leftarrow$.

3 The proof

Now we are ready to prove correctness of the magic transformation. The required property is that both programs have the same answers for Q. Our proof consists of two lemmas (inclusion in two directions). Moreover, the second lemma formalizes the main intuition behind the transformation: program $magic(P, Q)$ describes the sets of procedure calls and successes in computations of program P and query Q, under Prolog selection rule. In the lemmas, P is a program and Q is an atomic query, both over $\mathscr{L}$.

Lemma 1
For any query R over $\mathscr{L}$, if $magic(P,Q) \models R$, then $P \models R$.

Proof
Consider a proof tree $\mathscr{T}$ for $magic(P,Q)$ and A, where A is an atom from R. Removing from $\mathscr{T}$ each atom $^\bullet B$ results in (a set of trees containing) a proof tree for P and A. Thus, by Theorem 2 if $magic(P,Q) \models R$, then $P \models R$. □

Lemma 2
P with Q is correct with respect to a call-success specification $\langle pre, post \rangle$ given by

$$pre = \{ A \mid magic(P,Q) \models {}^\bullet A \},$$
$$post = \{ A \mid magic(P,Q) \models A \}.$$

In particular, each computed answer $Q\theta$ for P and Q is in *post*.

Proof (outline). Notice that the magic program is an encoding of the correctness conditions from Theorem 3. □

Proof (detailed). The magic program contains $^\bullet Q \leftarrow$; hence, $Q \in pre$. Consider an instance $(H \leftarrow B_1, \ldots, B_n)\theta$ of a clause of P. Assume that $H\theta \in pre$ and $B_1\theta, \ldots, B_{i-1}\theta \in post$ $(0 < i \leqslant n+1)$. Then $magic(P,Q) \models {}^\bullet H\theta, B_1\theta, \ldots, B_{i-1}\theta$. If $i = n+1$, then $magic(P,Q) \models H\theta$ (by the clause from Case 1 of Definition 2). If $i \leqslant n$, then $magic(P,Q) \models {}^\bullet B_i\theta$ (by the clause from Case 2 of Definition 2). Thus, the sufficient condition for correctness (from Theorem 3) is satisfied. □

Corollary 1
If $P \models Q\sigma$, then $magic(P,Q) \models Q\sigma$.

Proof
By completeness of LD-resolution, $Q\sigma$ is an instance of a computed answer $Q\theta$ for P and Q. By Lemma 2, $Q\theta \in post$. Hence, $Q\sigma \in post$. □

From Lemma 1 and Corollary 1, the following immediately follows:

Theorem 4 (*Correctness of the transformation*)
Let P be a program, Q an atomic query, and θ a substitution. Then

$$P \models Q\theta \quad \text{iff} \quad magic(P,Q) \models Q\theta.$$

In other words, programs P and $magic(P,Q)$ have the same sets of answers for Q. Hence by Theorem 1, any computed answer for P, Q is an instance of a computed answer for $magic(P,Q)$, Q; and any computed answer for $magic(P,Q)$, Q is an instance of a computed answer for P, Q. The correctness is sometimes expressed in a less general way, as in the Corollary 2 given below (which follows immediately from Theorem 4).

Corollary 2
$\mathscr{M}_P \cap [Q] = \mathscr{M}_{magic(P,Q)} \cap [Q]$, where $\mathscr{M}_P$ denotes the least Herbrand model of P, and $[Q]$ is the set of ground instances of Q.

3.1 Variants of magic transformation

The reader is encouraged to check that the proof is also valid for a class of magic transformations, characterized as follows: (1) In a clause $H \leftarrow {}^{\bullet}H, \ldots$ from Case 1 of Definition 2, the body atom ${}^{\bullet}H$ may be removed; and (2) some body atom(s) from a clause ${}^{\bullet}B_i \leftarrow \ldots$ (Definition 2, Case 2) may be removed (Nilsson and Małuszyński 1995).

In some approaches (e.g., Beeri and Ramakrishnan 1991), an atom ${}^{\bullet}B_i$ may be added to the body of a magic program clause when the body contains B_i. Such program is logically equivalent to $magic(P, Q)$, thus our correctness theorem also holds for this case.[3]

An important class of magic transformations employs adornments (see, e.g., Ramakrishnan 1991; Beeri and Ramakrishnan 1991). The original program P is transformed into an *adorned program* P^{ad} by renaming predicate symbols into fresh ones. (We omit the details of the transformation.) A symbol p may be renamed into more than one symbols; thus several renamings of a clause $C \in P$ may appear in P^{ad}. Similarly, the query Q is transformed into $\overline{Q}$ (by applying a selected renaming of its predicate symbol). The two programs are equivalent in the sense that $P \models Q\theta$ iff $P^{ad} \models \overline{Q}\theta$. The new magic program is obtained by applying the magic transformation from Definition 2 to the adorned program: $magic'(P, Q) = magic(P^{ad}, \overline{Q})$. From Theorem 4, we obtain[4] correctness of this magic transformation: $P \models Q\theta$ iff $magic'(P, Q) \models \overline{Q}\theta$.

4 Concluding remarks

We first outline some other correctness proofs of magic transformation. Then we discuss the method of Theorem 3 used in our proof.

Mascellani and Pedreschi (2002) prove the equivalence $\mathscr{M}_P \cap [Q] = \mathscr{M}_{magic(P,Q)} \cap [Q]$ of Corollary 2. The proof employs Herbrand interpretations. In particular, it studies the intersection of the least Herbrand models (of $magic(P, Q)$ and P) with a Herbrand interpretation I, which is related to the set *pre* of Lemma 2.

The main part of the proof of Ramakrishnan (1991, Th. 5.1), corresponding to proving Corollary 1, is based on constructing a proof tree for $magic(P, Q)$ and Q, whenever a proof tree for P and Q exists. The proof is by induction on the tree for P. The inductive step considers an instance $Q \leftarrow {}^{\bullet}Q, B_1, \ldots, B_n$ of a clause of $magic(P, Q)$. By the inductive assumption, there exist trees for $magic(P, B_i)$ and B_i.

[3] In order to show the equivalence, let P' be the program $magic(P, Q)$ modified as described. Any clause of P' can be seen as $C' = A \leftarrow {}^{\bullet}H, B_1, \ldots, B_{i-1}, F$, where $C = A \leftarrow {}^{\bullet}H, B_1, \ldots, B_{i-1}$ is a clause of $magic(P, Q)$, and F is a possibly empty conjunction of some literals of the form ${}^{\bullet}B_j$ $(j < i)$. Formula $C \rightarrow C'$ is a tautology; hence, $magic(P, Q) \models P'$.
In order to show $P' \models magic(P, Q)$, we prove by induction on i that $P' \models C$, for each clause $C \in magic(P, Q)$ as above. For $i = 1$, $C = C' \in P'$, as F is empty. For the inductive step, assume without loss of generality that F is a single atom ${}^{\bullet}B_j$. There is a clause $C_{B_j} = {}^{\bullet}B_j \leftarrow {}^{\bullet}H, B_1, \ldots, B_{j-1}$ in $magic(P, Q)$, where $j < i$. By the inductive assumption, $P' \models C_{B_j}$. Also, $P' \models C'$. Formula $(C_{B_j} \wedge C') \rightarrow C$ is a tautology (e.g., apply the resolution principle w.r.t. F to C_{B_j} and C'). Thus, $P' \models C$.

[4] The proof is as follows: $P \models Q\theta$ iff $P^{ad} \models \overline{Q}\theta$ iff (by Th. 4) $magic(P^{ad}, \overline{Q})$.

In order to construct trees for $magic(P,Q)$ and each B_i, one needs to show that $magic(P,Q) \models {}^{\bullet}B_i$. This is done by induction on i. The correctness proof of Beeri and Ramakrishnan (1991) is similar.

An important intuition about the magic transformation, and a motivation for introducing it, seems to be the correspondence between the magic program and the calls and successes of the original one. This correspondence is neglected in the aforementioned proofs. In contrast, we formalize it as Lemma 2, and it is a core of our proof.

Nilsson (1995) presented a concise proof of a property related to Lemma 2 and Theorem 4. He showed correspondence between the declarative semantics[5] of $magic(P,Q)$ and the collecting top-down abstract interpretation of P with Q. The latter provides supersets of the set of calls and the set of successes in LD-derivations. So the main idea is similar to that of our proof; however, the notion of abstract interpretation is additionally employed.

The main reason for conciseness of the proof of Theorem 4 was employing the correctness proof method of Theorem 3 (Drabent and Miłkowska 2005, Sec. 3.2). The method deals with properties of LD-derivations. Such a property may be nondeclarative (i.e., inexpressible by means of the declarative semantics). The sufficient condition from Theorem 3 was initially proposed by Bossi and Cocco (1989) and is a central concept of (Apt 1997, Ch. 8). (Programs/queries satisfying the condition are called there *well-asserted*.) Formally, Theorem 3 is stronger than the corresponding results in (Bossi and Cocco 1989), or (Apt 1997), as they do not deal with calls and successes, or – respectively – with successes in the derivations.[6] So we give its proof in the Appendix.

The method of Theorem 3 is a special case of that of (Drabent and Małuszyński 1988).[7] The main difference is that call-success specifications in Drabent and Małuszyński (1988) are not required to be closed under substitution. Another correctness proof methods for nondeclarative properties, with specifications not necessarily closed under substitution, are presented in (Colussi and Marchiori 1991; Drabent 1997).

Often we are interested in declarative properties of programs. For such properties a simpler proof method exists, usually attributed to Clark (1979). We illustrate that method in Appendix B by another proof of Corollary 1. The reader is referred to (Drabent and Miłkowska 2005, Secs. 3.1, 3.2) for a presentation, further references, and for a comparison with methods dealing with nondeclarative properties.

A Appendix A

Here we present a formal definition of procedure calls and successes, and a soundness proof for the method of proving programs correct with respect to call-success

[5] More precisely, the s-semantics (Bossi *et al.* 1994).

[6] Thus the proof method of Apt (1997, Ch. 8) is insufficient to obtain Lemma 2. However, it can be used to obtain a weaker lemma, stating that the computed answers are in *post*. Such lemma is sufficient to derive Theorem 4.

[7] In (Apt and Marchiori 1994), it is shown that the sufficient condition of Theorem 3 is a special case of that of Drabent and Małuszyński (1988).

specifications (Theorem 3). The definition follows that of Drabent and Małuszyński (1988).

Definition 3 (*Calls and successes*)
Let $Q_0, Q_1, Q_2, \ldots$ be the sequence of queries and $\theta_1, \theta_2, \ldots$ the sequence of mgus of an LD-derivation $\mathscr{D}$. Let $\theta_{i,j} = \theta_{i+1} \cdots \theta_j$ for $i < j$.

An atom A is a *procedure call* in $\mathscr{D}$ iff A is the first atom of some Q_i ($Q_i = A, \mathbf{B}$).

An atom A' is a *procedure success* (of a call A) in $\mathscr{D}$ iff
- $Q_i = A, \mathbf{B}$ for some $i \geqslant 0$,
- $Q_j = \mathbf{B}\theta_{i,j}$ for some $j > i$,
- and $A' = A\theta_{i,j}$ for the least such j.

Note that if A' is a success of a procedure call A (in an LD-derivation for a program P), then A' is a computed answer for A (and P). The corresponding successful derivation for A can be constructed out of the queries $Q_i, \ldots, Q_j$ as above, by removing $\mathbf{B}\theta_{i,l}$ from each query $Q_l = Q'_l, \mathbf{B}\theta_{i,l}$, for $l = i \ldots, j$ (where $\theta_{i,i}$ stands for ϵ, and $Q'_i = A$).

Proof of Theorem 3. Assume that the conditions of the theorem are satisfied, and consider an LD-derivation for P and Q. By (Apt 1997, Cor. 8.8), each procedure call in the derivation is in *pre*.

As explained above, each procedure success A' of a call A is a computed answer for A. By (Apt 1997, Corollary 8.9), the computed answer is in *post*. □

B Appendix B. Declarative proof of Corollary 1

The proof method (Clark 1979) is based on a property that, given an interpretation I, if $I \models P$, then $I \models Q$ for each answer Q of a program P. Such I is treated as a specification; $I \models P$ is a sufficient condition for correctness of P with respect to I.

We will use term interpretations (Apt 1997, Sec. 4.4); their interpretation domain is the set of all the terms (of the given language). Ground terms are interpreted as themselves. A valuation for variables is a substitution. Under a valuation η, a term t is interpreted as $t\eta$. An interpretation is (represented as) a set of atoms. An atom A is true in an interpretation I under a valuation η iff $A\eta \in I$. Thus, $I \models A$ iff each instance of A is in I. For a clause $C = H \leftarrow B_1, \ldots, B_n$, we have $I \models H \leftarrow B_1, \ldots, B_n$ iff $B_1\eta, \ldots, B_n\eta \in I$ implies $H\eta \in I$ for each instance $C\eta$ of C.

Proof of Corollary 1. Let us abbreviate $MP = magic(P, Q)$. As a specification for P, we take the interpretation

$$I = \{\, A \mid A \text{ is an atom, } MP \not\models {}^{\bullet}A \text{ or } MP \models A \,\}.$$

Obviously,

$$\text{if } A \in I, \text{ then } MP \models {}^{\bullet}A \text{ implies } MP \models A. \tag{B 1}$$

We show $I \models P$ (hence, P is correct with respect to I). Let $H \leftarrow B_1, \ldots, B_n \in P$. Assume $B_1\eta, \ldots, B_n\eta \in I$. We have to show that $H\eta \in I$. First note that

$MP \models {}^{\bullet}H\eta$, $MP \models B_1\eta, \ldots$, $MP \models B_{i-1}\eta$ imply $MP \models {}^{\bullet}B_i\eta$ (by a clause of MP from Case 2 of Definition 2), and hence, $MP \models B_i\eta$, by (B 1). By simple induction we obtain that $MP \models {}^{\bullet}H\eta$ implies $MP \models B_1\eta, \ldots, MP \models B_n\eta$, and thus it implies $MP \models H\eta$ (by the clause from Case 1 of Definition 2). If $MP \not\models {}^{\bullet}H\eta$, then $H\eta \in I$ (by the definition of I). Otherwise, by the above implication $MP \models H\eta$; thus, $H\eta \in I$.

By the assumption of the Corollary, $Q\sigma$ is an answer for P. Thus, from $I \models P$, it follows that $I \models Q\sigma$; hence, $Q\sigma \in I$. As $MP \models {}^{\bullet}Q\sigma$, we have $MP \models Q\sigma$. □

References

Apt, K. R. 1997. *From Logic Programming to Prolog*. International series in computer science. Prentice-Hall, New Jersey.

Apt, K. R. and Marchiori, E. 1994. Reasoning about Prolog programs: From modes through types to assertions. *Formal Aspects of Computing 6,* 6A, 743–765.

Beeri, C. and Ramakrishnan, R. 1991. On the power of magic. *Journal of Logic Programming 10,* 1–4, 255–299.

Bossi, A. and Cocco, N. 1989. Verifying correctness of logic programs. In *TAPSOFT,* Vol. 2, J. Díaz and F. Orejas, Eds. Lecture notes in computer science, vol. 352. Springer, New York, 96–110.

Bossi, A., Gabbrielli, M., Levi, G. and Martelli, M. 1994. The s-semantics approach: Theory and applications. *Journal of Logic Programming 19/20*, 149–197.

Clark, K. L. 1979. *Predicate Logic as Computational Formalism.* Technical Report 79/59, Imperial College, London.

Colussi, L. and Marchiori, E. 1991. Proving correctness of logic programs using axiomatic semantics. In *Logic Programming, Proceedings of the Eigth International Conference*, K. Furukawa, Ed. MIT Press, Cambridge, MA, 629–642.

Deransart, P. 1993. Proof methods of declarative properties of definite programs. *Theoretical Computer Science 118,* 2, 99–166.

Drabent, W. 1997. A Floyd-Hoare method for Prolog. Linköping electronic articles in computer and information science 2. URL: http://www.ep.liu.se/ea/cis/1997/013/

Drabent, W. and Małuszyński, J. 1988. Inductive assertion method for logic programs. *Theoretical Computer Science 59*, 133–155.

Drabent, W. and Miłkowska, M. 2005. Proving correctness and completeness of normal programs – a declarative approach. *Theory and Practice of Logic Programming 5,* 6, 669–711.

Mascellani, P. and Pedreschi, D. 2002. The declarative side of magic. In *Computational Logic: Logic Programming and Beyond*, A. C. Kakas and F. Sadri, Eds. Lecture notes in computer science, vol. 2408. Springer, New York, 83–108.

Nilsson, U. 1995. Abstract interpretation: A kind of magic. *Theoretical Computer Science 142,* 1, 125–139.

Nilsson, U. and Małuszyński, J. 1995. *Logic, Programming and Prolog*, 2nd ed. (Previously published by John Wiley).URL: http://www.ida.liu.se/~ulfni/lpp/

Ramakrishnan, R. 1991. Magic templates: A spellbinding approach to logic programs. *Journal of Logic Programming 11,* 3 & 4, 189–216.