

# *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*

*Linköpings universitet, Department of Computer and Information Science  
S – 581 83 Linköping, Sweden*

*(e-mail: drabent at ipipan dot waw dot pl)*

*submitted 21 December 2009; revised 7 August 2010; accepted ?*

---

## **Abstract**

The paper presents a simple and concise proof of correctness of the magic transformation. We believe 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, magic transformation, declarative semantics, LD-resolution, operational semantics

---

## **1 Introduction**

Magic transformation (see (Nilsson and Maluszynski 1995, Chapter 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 a 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 I found that such a proof was rather easy to construct. Moreover the result turned out to be surprisingly concise. In this note I present the proof with all the details. I believe 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-sets 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 is maybe even 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  $Q$  which 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  $Q_0$ .<sup>1</sup> 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  $Q\sigma$ .<sup>2</sup>

A *proof tree* (called sometimes 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  $B_1, \dots, B_n$  ( $n \geq 0$ ) are the children of a node  $H$  then  $H \leftarrow B_1, \dots, B_n$  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 (Apt 1997, Th.4.24(v)). The latter is attributed to (Clark 1979) in (Deransart 1993, Proposition 2.6).

We focus on LD-resolution (SLD-resolution with the Prolog selection rule) and will 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* w.r.t. 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 in *post*.

<sup>1</sup> In (Apt 1997) answers are also called correct instances of initial queries, and computed answers are called computed instances.

<sup>2</sup> For a proof see e.g. (Apt 1997, Th. 4.4, 4,13).

Notice 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 Milkowska 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 non-ground) instance  $H \leftarrow B_1, \dots, B_n$  ( $n \geq 0$ ) of each clause of  $P$

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

Then  $P$  with  $Q$  is correct w.r.t.  $\langle pre, post \rangle$ .

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

It remains to define the magic transformation. It adds new predicate symbols to the alphabet  $\mathcal{L}$  of programs and queries; for each predicate symbol  $p$ , the unique new symbol  $\bullet p$  is added. In a simple version, for instance that of (Nilsson and Maluszynski 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 of the selected positions, as it is irrelevant for the correctness of magic transformation.) Let  $\bullet \mathcal{P}$  denote the set of new predicate symbols. If  $A = p(t_1, \dots, t_n)$  is an atom over  $\mathcal{L}$  then  $\bullet A$  denotes  $\bullet p(t_{i_1}, \dots, t_{i_{k_p}})$ , where  $i_1, \dots, i_{k_p}$  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  $\mathcal{L}$ . (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  $Q$  an atomic query. The magic program  $magic(P, Q)$  for  $P$  and  $Q$  is the program containing

1. a clause  $H \leftarrow \bullet H, B_1, \dots, B_n$  for each clause  $H \leftarrow B_1, \dots, B_n$  in  $P$ ,
2. a clause  $\bullet B_i \leftarrow \bullet H, B_1, \dots, B_{i-1}$  for each clause  $H \leftarrow B_1, \dots, B_n$  in  $P$  and each  $i = 1, \dots, n$ ,
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  $\mathcal{L}$ .

*Lemma 1*

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

PROOF: Consider a proof tree  $\mathcal{T}$  for  $\text{magic}(P, Q)$  and  $A$ , where  $A$  is an atom from  $R$ . Removing from  $\mathcal{T}$  each atom  $\bullet B$  results in (a set of trees containing) a proof tree for  $P$  and  $A$ . Thus by Th. 2, if  $\text{magic}(P, Q) \models R$  then  $P \models R$ .  $\square$

*Lemma 2*

$P$  with  $Q$  is correct w.r.t. a call-success specification  $\langle \text{pre}, \text{post} \rangle$  given by

$$\begin{aligned} \text{pre} &= \{ A \mid \text{magic}(P, Q) \models \bullet A \}, \\ \text{post} &= \{ A \mid \text{magic}(P, Q) \models A \}. \end{aligned}$$

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

PROOF (outline): Notice that the magic program is an encoding of the correctness conditions from Th. 3.  $\square$

PROOF (detailed): The magic program contains  $\bullet Q \leftarrow$ , hence  $Q \in \text{pre}$ . Consider an instance  $(H \leftarrow B_1, \dots, B_n)\theta$  of a clause of  $P$ . Assume that  $H\theta \in \text{pre}$  and  $B_1\theta, \dots, B_{i-1}\theta \in \text{post}$  ( $0 < i \leq n+1$ ). Then  $\text{magic}(P, Q) \models \bullet H\theta, B_1\theta, \dots, B_{i-1}\theta$ . If  $i = n+1$  then  $\text{magic}(P, Q) \models H\theta$  (by the clause from case 1 of Def. 2). If  $i \leq n$  then  $\text{magic}(P, Q) \models \bullet B_i\theta$  (by the clause from case 2 of Def. 2). Thus the sufficient condition for correctness (from Th. 3) is satisfied.  $\square$

*Corollary 1*

If  $P \models Q\sigma$  then  $\text{magic}(P, Q) \models Q\sigma$ .

PROOF: By completeness of LD-resolution,  $Q\sigma$  is an instance of a computed answer  $Q\theta$  for  $P$  and  $Q$ . By Lemma 2,  $Q\theta \in \text{post}$ . Hence  $Q\sigma \in \text{post}$ .  $\square$

From Lemma 1 and the corollary it immediately follows:

*Theorem 4 (Correctness of the transformation)*

Let  $P$  be a program,  $Q$  an atomic query, and  $\theta$  a substitution. Then

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

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

*Corollary 2*

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

**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  $H \leftarrow \bullet H, \dots$  from case 1 of Def. 2, the body atom  $\bullet H$  may be removed; 2. some body atom(s) from a clause  $\bullet B_i \leftarrow \dots$  (Def. 2, case 2) may be removed (Nilsson and Maluszynski 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 holds also for this case.<sup>3</sup>

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  $\bar{Q}$  (by applying a selected renaming of its predicate symbol). The two programs are equivalent in the sense that  $P \models Q\theta$  iff  $P^{ad} \models \bar{Q}\theta$ . The new magic program is obtained by applying the magic transformation from Df. 2 to the adorned program:  $magic'(P, Q) = magic(P^{ad}, \bar{Q})$ . From Th. 4 we obtain<sup>4</sup> correctness of this magic transformation:  $P \models Q\theta$  iff  $magic'(P, Q) \models \bar{Q}\theta$ .

#### 4 Concluding remarks

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

Mascellani and Pedreschi (2002) prove the equivalence  $\mathcal{M}_P \cap [Q] = \mathcal{M}_{magic(P, Q)} \cap [Q]$  of Corollary 2. The proof employs Herbrand interpretations. In particular it studies the intersection of the least Herbrand models (of  $magic(P, Q)$  and of  $P$ ) 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  $Q \leftarrow \bullet Q, B_1, \dots, B_n$  of a clause of  $magic(P, Q)$ . By the inductive assumption, there exist trees for  $magic(P, B_i)$  and  $B_i$ . To construct trees for  $magic(P, Q)$  and each  $B_i$ , one needs to show that

<sup>3</sup> To show the equivalence, let  $P'$  be the program  $magic(P, Q)$  modified as described. Any clause of  $P'$  can be seen as  $C' = A \leftarrow \bullet H, B_1, \dots, B_{i-1}, F$ , where  $C = A \leftarrow \bullet H, B_1, \dots, B_{i-1}$  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  $magic(P, Q) \models P'$ . 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  $C_{B_j} = \bullet B_j \leftarrow \bullet H, B_1, \dots, B_{j-1}$  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  $P' \models C$ .

<sup>4</sup> The proof is:  $P \models Q\theta$  iff  $P^{ad} \models \bar{Q}\theta$  iff (by Th. 4)  $magic(P^{ad}, \bar{Q})$ .

$magic(P, Q) \models \bullet B_i$ . 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 Th. 4. He showed correspondence between the declarative semantics<sup>5</sup> 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 Th. 4 was employing the correctness proof method of Th. 3 (Drabent and Miłkowska 2005, Section 3.2). The method deals with properties of LD-derivations. Such a property may be non-declarative (i.e. inexpressible by means of the declarative semantics). The sufficient condition from Th. 3 was initially proposed by Bossi and Cocco (1989), and is a central concept of (Apt 1997, Chapter 8). (Programs/queries satisfying the condition are called there *well-asserted*.) Formally, Th. 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.<sup>6</sup> So we give its proof in the Appendix.

The method of Th. 3 is a special case of that of (Drabent and Małuszyński 1988)<sup>7</sup>. 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 non-declarative 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, Sections 3.1, 3.2) for a presentation, further references, and for a comparison with methods dealing with non-declarative properties.

## Appendix A

Here we present a formal definition of procedure calls and successes, and a soundness proof for the method of proving programs correct w.r.t. call-success specifications (Th. 3). The definition follows that of (Drabent and Małuszyński 1988).

<sup>5</sup> More precisely, the s-semantics (Bossi et al. 1994).

<sup>6</sup> Thus the proof method of (Apt 1997, Chapter 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 Th. 4.

<sup>7</sup> In (Apt and Marchiori 1994) it is shown that the sufficient condition of Th. 3 is a special case of that of (Drabent and Małuszyński 1988).

*Definition 3 (Calls and successes)*

Let  $Q_0, Q_1, Q_2, \dots$  be the sequence of queries and  $\theta_1, \theta_2, \dots$  the sequence of mgu's of an LD-derivation  $\mathcal{D}$ . Let  $\theta_{i,j} = \theta_{i+1} \cdots \theta_j$  for  $i < j$ .

An atom  $A$  is a *procedure call* in  $\mathcal{D}$  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  $\mathcal{D}$  iff

- $Q_i = A, \mathbf{B}$  for some  $i \geq 0$ ,
- $Q_j = \mathbf{B}\theta_{i,j}$  for some  $j > i$ ,
- and  $A' = A\theta_{i,j}$  for the least such  $j$ .

Notice 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  $Q_i, \dots, Q_j$  as above, by removing  $\mathbf{B}\theta_{i,l}$  from each query  $Q_l = Q'_l, \mathbf{B}\theta_{i,l}$ , for  $l = i \dots, j$  (where  $\theta_{i,i}$  stands for  $\epsilon$ , 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, Corollary 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*.  $\square$

## 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  $P$  w.r.t.  $I$ .

We will use term interpretations (Apt 1997, Section 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  $\eta$ , 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  $\eta$  iff  $A\eta \in I$ . Thus  $I \models A$  iff each instance of  $A$  is in  $I$ . For a clause  $C = H \leftarrow B_1, \dots, B_n$  we have:  $I \models H \leftarrow B_1, \dots, B_n$  iff  $B_1\eta, \dots, B_n\eta \in I$  implies  $H\eta \in I$  for each instance  $C\eta$  of  $C$ .

PROOF (of Corollary 1): Let us abbreviate  $MP = \text{magic}(P, Q)$ . 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:

$$\text{If } A \in I \text{ then } MP \models \bullet A \text{ implies } MP \models A. \quad (\text{B1})$$

We show  $I \models P$  (hence  $P$  is correct w.r.t.  $I$ ). Let  $H \leftarrow B_1, \dots, B_n \in P$ . Assume  $B_1\eta, \dots, B_n\eta \in I$ . We have to show that  $H\eta \in I$ . Notice first that  $MP \models \bullet H\eta$ ,  $MP \models B_1\eta$ ,  $\dots$ ,  $MP \models B_{i-1}\eta$  imply  $MP \models \bullet B_i\eta$  (by a clause of  $MP$  from case 2 of Def. 2), and hence  $MP \models B_i\eta$ , by (B1). By simple induction we

obtain that  $MP \models \bullet H\eta$  implies  $MP \models B_1\eta, \dots, MP \models B_n\eta$ , and thus it implies  $MP \models H\eta$  (by the clause from case 1 of Def. 2). If  $MP \not\models \bullet H\eta$  then  $H\eta \in I$  (by the definition of  $I$ ). Otherwise, by the implication above,  $MP \models H\eta$ ; 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  $MP \models Q\sigma$ .  $\square$

## References

- APT, K. R. 1997. *From Logic Programming to Prolog*. International Series in Computer Science. Prentice-Hall.
- APT, K. R. AND MARCHIORI, E. 1994. Reasoning about Prolog programs: From modes through types to assertions. *Formal Asp. Comput.* 6, 6A, 743–765.
- BEERI, C. AND RAMAKRISHNAN, R. 1991. On the power of magic. *J. Log. Program.* 10, 1/2/3&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, 96–110.
- BOSSI, A., GABRIELLI, M., LEVI, G., AND MARTELLI, M. 1994. The s-semantic approach: Theory and applications. *J. Log. Program.* 19/20, 149–197.
- CLARK, K. L. 1979. Predicate logic as computational formalism. Tech. Rep. 79/59, Imperial College, London. December.
- COLUSSI, L. AND MARCHIORI, E. 1991. Proving correctness of logic programs using axiomatic semantics. In *Logic Programming, Proceedings of the Eighth International Conference*, K. Furukawa, Ed. MIT Press, 629–642.
- DERANSART, P. 1993. Proof methods of declarative properties of definite programs. *Theor. Comput. Sci.* 118, 2, 99–166.
- DRABENT, W. 1997. A Floyd-Hoare method for Prolog. *Linköping Electronic Articles in Computer and Information Science* 2. <http://www.ep.liu.se/ea/cis/1997/013/>.
- DRABENT, W. AND MALUSZYŃSKI, 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, 83–108.
- NILSSON, U. 1995. Abstract interpretation: A kind of magic. *Theor. Comput. Sci.* 142, 1, 125–139.
- NILSSON, U. AND MALUSZYŃSKI, J. 1995. *Logic, Programming and Prolog* (2ed). Previously published by John Wiley & Sons Ltd. <http://www.ida.liu.se/~ulfni/lpp/>.
- RAMAKRISHNAN, R. 1991. Magic templates: A spellbinding approach to logic programs. *J. Log. Program.* 11, 3&4, 189–216.