

# A Computational Model for Functional Logic Deductive Databases

Jesús M. Almendros-Jiménez<sup>1</sup>\*, Antonio Becerra-Terón<sup>1</sup> and  
Jaime Sánchez-Hernández<sup>2</sup>\*

<sup>1</sup>Dpto. de Lenguajes y Computación. Universidad de Almería

<sup>2</sup>Dpto. de Sistemas Informáticos y Programación. Universidad Complutense de Madrid  
email: {jalmen, abecerra}@ual.es, jaime@sip.ucm.es

**Abstract.** This paper adds the handling of negative information to a functional-logic deductive database language. By adopting as semantics for negation the so-called *CRWLF*, wherein the negation is intended as 'finite failure' of reduction, we will define Herbrand algebras and models for this semantics and a fix point operator to be used in a new goal-directed bottom-up evaluation mechanism based on magic transformations. This bottom-up evaluation will simulate the top-down one of the original program; in fact, it will carry out a goal-directed lazy evaluation.

## 1 Introduction

*Deductive databases* [17] are database management systems whose query language and, usually, storage structure are designed around a logical data model. They offer a rich query language which extends the relational model in many directions (for instance, support for non-first normal form and recursive relations) and they are suited for application in which a large number of data must be accessed and complex queries must be supported.

With respect to the operational semantics, most deductive database systems (for instance, DATALOG [19], CORAL [16], ADITI [20]) use *bottom-up* evaluation instead of *top-down* one like Prolog systems. The reason is that the bottom-up approach allows to use a *set-at-a-time* evaluation, i.e. it processes sets of goals, rather than proceeding one (sub) goal at a time, where operations like relational joins can be made for disk-resident data efficiently. Therefore, when the program is data-intensive, this evaluation method is potentially much more efficient than top-down techniques. The idea of the *goal-directed bottom-up evaluation* is to generate by using the *fix point operator* [3] the subset of the *Herbrand model* of the program relevant to the query solving. With this aim, the bottom-up evaluation in such languages involves a query-program transformation termed *Magic Sets* [5], in such a way that a logic program-query is transformed into a *magic* logic program-query whose bottom-up evaluation is devised to simulate the top-down one of the original program and query. The program is evaluated until no new facts are generated or the answer to the query is found. The transformed program adds new predicates, called *magic predicates*, whose role is to pass information (instantiated and partially instantiated arguments in the predicates of the query) to the program in order to consider only those instances of the program rules relevant to the query solving. Several transformation methods have been studied in the past, for instance, *Generalized Magic Sets* [5], *Generalized Supplementary Magic Sets* [5] and *Magic Templates* [15].

\* The authors have been partially supported by the Spanish CICYT (project TIC 98-0445-C03-02 TREND)

The use of negation in deductive databases allows to increase their expressive power as query languages. The introduction of negation in logic programming (see [4] for a survey), and thus the study of semantic models of the so-called *general logic programs*, have been widely studied in the past. Most relevant semantics for handling of negation are the *stable model semantics* [6] and the *well-founded semantics* [21].

However, the incorporation of negation in the deductive languages implies new problems in the magic transformation. In fact, new magic transformations have been proposed pointing out the one presented for *modularly stratified programs* [18] and the *doubled program* approach [10] by adopting both the well-founded semantics. The problem arises from the three valued nature of the magic predicates that result, and the well-founded model of the transformed magic program and the original one may disagree [10]. In [10] classes of *side-ways information-passing strategies*, also called *sips*, which ensure that the magic sets are two-valued, are defined. These sips, named *well-founded sips*, make sure that the well-founded model of a program is preserved w.r.t. the query in the transformed program. Moreover, they subsume the *left-to-right sips* intended for modularly stratified programs [18]. Finally, they present a new magic transformation by using a doubled program technique which preserves the well-founded model w.r.t. the query regardless of the sips to be used. The drawback of this approach is that the bottom-up evaluation of the program must end.

On the other hand, the integration of functional and logic programming has been widely investigated during the last years. It has led to the recent design of modern programming languages such as CURRY [9] and *TOY* [13]. The basic ideas in functional-logic programming consist in *lazy narrowing* as operational mechanism, following some class of *narrowing strategy* combined with some kind of *constraint solving* and *higher order* features, see [8] for a survey.

In [1], a framework for goal-directed bottom-up evaluation for functional-logic programs without negation has been proposed. As in the logic paradigm, the bottom-up evaluation is based on a magic transformation for a given program-query into a magic program-query. In the cited paper, the semantics adopted for the programs is the *Constructor Based ReWriting Logic (CRWL)* presented in [7]. This bottom-up evaluation is based on the use of a fix point operator over *CRWL*-Herbrand algebras and it simulates the *demand driven strategy* [11] for top-down evaluation of *CRWL*-programs.

Recently, a framework called *Constructor Based ReWriting Logic with Failure (CRWLF)* has been presented in [14], extending the semantics *CRWL* in order to handle negative information and wherein the negation is intended as 'finite failure' of reduction. The formulas that are provable in *CRWL* can also be proved in *CRWLF* but, in addition, *CRWLF* provides 'proofs of unprovability' within *CRWL*. *CRWLF* can only give an approximation to failure in *CRWL* that corresponds to the cases in which unprovability refers to 'finite failure' of reduction.

The aim of this paper is to add the handling of negative information to a functional-logic language, and to present a goal-directed bottom-up evaluation mechanism in order to get a computational model for a "functional-logic" deductive language. With this aim, we will replace the semantics *CRWL* used in [1], by the semantics *CRWLF* and we will present Herbrand algebras and models and a fix point operator which computes the Herbrand model of a *CRWLF*-

program. Finally, we will propose an extension of our goal-directed bottom-up evaluation, based on a new magic transformation and the use of the defined fix point operator.

As an example, a “functional-logic” deductive database can handle a boss hierarchical line as follows:

- (1)  $\text{boss}(\text{jesus}) \rightarrow \text{jaime}$ .    (3)  $\text{member}(X, []) \rightarrow \text{false}$ .
- (2)  $\text{boss}(\text{jaime}) \rightarrow \text{antonio}$ .    (4)  $\text{member}(X, [Y|L]) \rightarrow \text{member}(X, L) \Leftarrow X \not\bowtie Y$ .
- (5)  $\text{member}(X, [Y|L]) \rightarrow \text{true} \Leftarrow X \bowtie Y$ .
- (6)  $\text{superboss}(P) \rightarrow [\text{boss}(P)|\text{superboss}(\text{boss}(P))]$ .

**Goal** :  $\neg \text{member}(\text{jaime}, \text{superboss}(\text{jesus})) \bowtie \text{true}$ .

where  $\bowtie$  and  $\not\bowtie$  refer to a joinability constraint (both sides reduce to the same constructor term) and its logical negation, respectively. Our evaluation method will evaluate the function `superboss`, which defines a possibly infinite data, as far as needed in order to solve the goal. It starts with `superboss(jesus)` as  $\perp$ , which represents the undefined value, and then `superboss(jesus)` is evaluated up to  $[\text{jaime}|\perp]$ , necessary for the goal solving. Moreover, in our framework, we have to solve lazily the negative constraints. For instance, supposing the query `superboss(X)  $\not\bowtie$  superboss(Y)` w.r.t. the above program and starting with `superboss(X)` and `superboss(Y)` as  $\perp$ , the evaluation can bind the variables `X` to `jesus` and `Y` to `jaime` evaluating `superboss(jesus)` up to  $[\text{jaime}|\perp]$  and `superboss(jaime)` up to  $[\text{antonio}|\perp]$  where  $[\text{jaime}|\perp]$  conflicts with  $[\text{antonio}|\perp]$  and therefore obtaining the answer  $X = \text{jesus}, Y = \text{jaime}$ .

As theoretic results of this paper, we will establish the soundness and completeness results of our bottom-up evaluation w.r.t. Herbrand models and the proof-semantics *CRWLF*. Moreover, we will establish correspondences among proofs of a given goal in the cited logic and the “facts” computed by means of the bottom-up evaluation showing the optimality of our method.

The rest of the paper will be organized as follows. In section 2, we will introduce *CRWLF*; section 3 will define the fix point operator and the Herbrand models; section 4 will present the magic transformation; section 5 will establish soundness, completeness and optimality results and, finally, section 6 will describe the conclusions and future work. Due to the lack of space, the full proofs of our results can be found in [2].

## 2 The *CRWLF* Framework

In this section we summarize the *Constructor ReWriting Logic with Failure* presented in [14]. Assuming a signature  $\Sigma = DC \cup FS$  where  $DC = \bigcup_{n \in \mathbb{N}} DC^n$  is a set of *constructor* symbols  $c, d, \dots$  and  $FS = \bigcup_{n \in \mathbb{N}} FS^n$  is a set of *function* symbols  $f, g, \dots$ , all of them with associated arity and such that  $DC \cap FS = \emptyset$ , and also a countable set  $\mathcal{V}$  of *variable* symbols  $X, Y, \dots$  we write *Term* for the set of (total) *terms*  $e, e', \dots$  (also called *expressions*) built up with  $\Sigma$  and  $\mathcal{V}$  in the usual way, and we distinguish the subset *CTerm* of (total) constructor terms or (total) *c-terms*  $s, t, \dots$ , built up only with  $DC$  and  $\mathcal{V}$ . Terms intend to represent possibly reducible expressions, whereas c-terms represent data values, not further reducible. We extend the signature  $\Sigma$  by adding two new constants:  $\perp$  that plays the role of undefined value and  $\text{F}$  that will be used as an explicit representation of failure of reduction. The set *Term* $_{\perp}$  of *partial* terms and the set *CTerm* $_{\perp}$  of *partial* c-terms are defined in a natural way. Partial c-terms represent the result of partially evaluated expressions, and thus they can be considered as

approximations to the value of expressions. Moreover, we will consider the sets  $Term_{\perp, \mathbb{F}}$  and  $CTerm_{\perp, \mathbb{F}}$ . A natural *approximation ordering*  $\leq$  over  $CTerm_{\perp, \mathbb{F}}$  can be defined as the least partial ordering satisfying:  $\perp \leq t$ ,  $X \leq X$  and  $h(t_1, \dots, t_n) \leq h(t'_1, \dots, t'_n)$ , if  $t_i \leq t'_i$  for all  $i \in \{1, \dots, n\}$ ,  $h \in DC \cup FS$ . The intended meaning of  $t \leq t'$  is that  $t$  is less defined or has less information than  $t'$ . Note that the only relations satisfied by  $\mathbb{F}$  are  $\perp \leq \mathbb{F}$  and  $\mathbb{F} \leq \mathbb{F}$ . In particular,  $\mathbb{F}$  is maximal. This is reasonable, since  $\mathbb{F}$  represents ‘failure of reduction’ and this gives a no further refinable information about the result of the evaluation of an expression.

In the context of *CRWLF*, a program  $\mathcal{P}$  is a set of conditional rewrite rules of the form:

$$\underbrace{f(t_1, \dots, t_n)}_{\text{head}} \rightarrow \underbrace{r}_{\text{body}} \Leftarrow \underbrace{C}_{\text{condition}}$$

where  $f \in FS^n$ , and fulfilling the following conditions:  $(t_1, \dots, t_n)$  is a linear tuple (each variable in it occurs only once) with  $t_1, \dots, t_n \in CTerm$ ;  $r \in Term$ ;  $C$  is a set of constraints of the form  $e' \bowtie e''$  (*joinability*),  $e' \diamond e''$  (*divergence*),  $e' \not\bowtie e''$  (*failure of joinability*) or  $e' \not\diamond e''$  (*failure of divergence*) where  $e', e'' \in Term^1$ ; *extra variables* are not allowed<sup>2</sup>, i.e.  $var(r) \cup var(C) \subseteq var(\bar{t})$ . The reading of the rule is:  $f(t_1, \dots, t_n)$  reduces to  $r$  if the condition  $C$  is satisfied. We will need to use *c-instances* of rules, where a *c-instance* of a program rule  $R$  is defined as  $[R]_{\perp, \mathbb{F}} = \{R\theta \mid \theta \in CSubst_{\perp, \mathbb{F}}\}$  with  $CSubst_{\perp, \mathbb{F}} = \{\theta : \mathcal{V} \rightarrow CTerm_{\perp, \mathbb{F}}\}$ .

In our framework and due to we allow non-determinism, in general an expression can be reduced to an infinite set of values, but we will need some finite representation of these sets. For example, we can define the non-deterministic function  $f$  as:  $f \rightarrow zero, f \rightarrow suc(suc(f))$ . It is easy to see that  $f$  can be reduced to the values  $zero, suc(suc(zero)), suc(suc(suc(suc(zero)))) \dots$ . We can use the undefined value  $\perp$  to express that the possible reductions of  $f$  have the form  $zero$  or  $suc(suc(\perp))$ , noted as  $f \triangleleft \{zero, suc(suc(\perp))\}$ . This set of values is a *Sufficient Approximation Set (SAS)* for  $f$  which provides enough information about the values of  $f$  to prove that  $f$  cannot be reduced to  $suc(zero)$ . Of course, an expression will have multiple *SAS*s. Any expression has  $\{\perp\}$  as its simplest *SAS* and, for example, the expression  $f$  has an infinite number of *SAS*s:  $\{\perp\}, \{zero, suc(suc(\perp))\}, \{zero, suc(suc(zero)), suc(suc(suc(suc(\perp))))\}, \dots$

In *CRWLF* five kinds of statements can be deduced (assume  $e \in Term_{\perp, \mathbb{F}}$ ):

- $e \triangleleft C$ :  $C$  is a *SAS* for  $e$ ;
- $e \bowtie e'$  (*joinability*):  $e$  and  $e'$  can be both reduced to some  $t \in CTerm$ ;
- $e \diamond e'$  (*divergence*):  $e$  and  $e'$  can be reduced to some (possibly partial) c-terms  $t$  and  $t'$  having a *DC-clash*.
- $e \not\bowtie e'$ : failure of  $e \bowtie e'$ ;
- $e \not\diamond e'$ : failure of  $e \diamond e'$ .

where given a set of constructors  $S$ , we say that the c-terms  $t$  and  $t'$  have a *S-clash* if they have different constructors of  $S$  at the same position.

<sup>1</sup> The original *CRWL* framework [7] only considered joinabilities; divergence constraints were incorporated in [12] and failures of both joinabilities and divergences are introduced in [14].

<sup>2</sup> In [7] extra variables are allowed, but the use of function nesting and non-deterministic functions can nicely replace them in many cases.

**Table 1.** Rules for *CRWLF*-provability

(1) $\frac{}{e \triangleleft \{\perp\}}$	(2) $\frac{}{X \triangleleft \{X\}} \quad X \in \mathcal{V}$
(3) $\frac{e_1 \triangleleft C_1 \quad \dots \quad e_n \triangleleft C_n}{c(e_1, \dots, e_n) \triangleleft \{c(\bar{t}) \mid \bar{t} \in C_1 \times \dots \times C_n\}} \quad c \in DC^n \cup \{F\}$	
(4) $\frac{e_1 \triangleleft C_1 \quad \dots \quad e_n \triangleleft C_n \quad \dots \quad f(\bar{t}) \triangleleft_R C_{R, \bar{t}} \quad \dots}{f(e_1, \dots, e_n) \triangleleft \bigcup_{R \in \mathcal{P}_f, \bar{t} \in C_1 \times \dots \times C_n} C_{R, \bar{t}}} \quad f \in FS^n$	
(5) $\frac{}{f(\bar{t}) \triangleleft_R \{\perp\}}$	(6) $\frac{r \triangleleft C \quad C}{f(\bar{t}) \triangleleft_R C} \quad (f(\bar{t}) \rightarrow r \Leftarrow C) \in [R]_{\perp, F}$
(7) $\frac{e_i \not\triangleleft e'_i}{f(\bar{t}) \triangleleft_R \{F\}}$	(8) $\frac{}{f(t_1, \dots, t_n) \triangleleft_R \{F\}}$
$(f(\bar{t}) \rightarrow r \Leftarrow \dots, e_i \triangleleft e'_i, \dots) \in [R]_{\perp, F} \quad R \equiv (f(s_1, \dots, s_n) \rightarrow r \Leftarrow C), t_i \text{ and } s_i \text{ have a } DC \cup \{F\}\text{-clash for some } i \in \{1, \dots, n\}$	
(9) $\frac{e \triangleleft C \quad e' \triangleleft C'}{e \bowtie e'}$	(10) $\frac{e \triangleleft C \quad e' \triangleleft C'}{e \diamond e'}$
(11) $\frac{e \triangleleft C \quad e' \triangleleft C'}{e \not\bowtie e'}$	(12) $\frac{e \triangleleft C \quad e' \triangleleft C'}{e \not\diamond e'}$

We will use the symbol  $\diamond$  to refer to any of the constraints  $\bowtie, \diamond, \not\bowtie, \not\diamond$ . The constraints  $\not\bowtie$  and  $\bowtie$  are called the *complementary* of each other; the same holds for  $\not\diamond$  and  $\diamond$ , and we write  $\tilde{\diamond}$  for the complementary of  $\diamond$ . When proving a constraint  $e \diamond e'$ , the calculus *CRWLF* will evaluate a *SAS* for the expressions  $e$  and  $e'$ . These *SAS*'s will consist of c-terms and provability of the constraint  $e \diamond e'$  depends on certain syntactic (hence decidable) relations between these ones defined as follows.

**Definition 1 (Relations over  $CTerm_{\perp, F}$ ).**

- $t \downarrow t' \Leftrightarrow_{def} t = t', t \in CTerm$
- $t \uparrow t' \Leftrightarrow_{def} t \text{ and } t' \text{ have a } DC\text{-clash}$
- $t \not\downarrow t' \Leftrightarrow_{def} t \text{ or } t' \text{ contain } F \text{ as subterm, or they have a } DC\text{-clash}$
- $\not\uparrow$  is defined as the least symmetric relation over  $CTerm_{\perp, F}$  satisfying:
  - i)  $X \not\uparrow X$ , for all  $X \in \mathcal{V}$
  - ii)  $F \not\uparrow t$ , for all  $t \in CTerm_{\perp, F}$
  - iii) if  $t_1 \not\uparrow t'_1, \dots, t_n \not\uparrow t'_n$  then  $c(t_1, \dots, t_n) \not\uparrow c(t'_1, \dots, t'_n)$ , for  $c \in DC^n$

Table 1 shows the rules for the *CRWLF*-calculus. Rule (1) considers  $\{\perp\}$  as the simplest *SAS* for any expression. Rules (2) and (3) consider the case of variables and constructors. In rule (4), when evaluating a function call  $f(e_1, \dots, e_n)$ , we produce *SAS*'s for the arguments. Next, we consider all the possible values  $t$  of the cross product of these *SAS*'s and all the rules  $R$  for  $f$ , denoted by  $\mathcal{P}_f$ , by generating a *SAS* for each combination:  $f(\bar{t}) \triangleleft_R C_{R, \bar{t}}$ . The notation  $\triangleleft_R$  indicates that only the rule  $R$  is used to produce the corresponding *SAS*. By joining the *SAS*'s  $C_{R, \bar{t}}$ , we obtain the final *SAS* for  $f(e_1, \dots, e_n)$ . Rules (5) to (8) consider all the possible ways in which a rule  $R$  can be used to produce a *SAS* for a call  $f(\bar{t})$  where  $t_i \in CTerm_{\perp, F}$ . Rule (5) is the trivial case. Rule (6) applies whether there exists a c-instance of  $R$  with head  $f(\bar{t})$  and the constraints of  $C$  of this c-instance are provable. In this case, the *SAS* will be the one produced by the body of the c-instance. Rules (7) and (8) produce the *SAS*  $\{F\}$  due to a failure of one of the constraints (by proving the complementary) and a failure in parameter passing, respectively. Finally, the rules (9) to (12) deal with constraints and are easily defined by using the relations  $\downarrow, \uparrow, \not\downarrow, \not\uparrow$ .

It can be proved that the relations  $\triangleleft, \bowtie, \diamond, \nabla, \triangleleft$  satisfy some desirable properties such as monotonicity or closure under substitutions. On the other hand, if  $e \diamond e'$  is provable, then  $e \hat{\diamond} e'$  is not provable. Full details about *CRWLF* can be found in [14].

Finally, a *goal*  $\mathcal{G}$  is a set of constraints and a solution  $\theta \in CSubst_{\perp, F}$  of  $\mathcal{G}$  w.r.t. a *CRWLF*-program  $\mathcal{P}$  holds  $\mathcal{P} \vdash_{CRWLF} \mathcal{G}\theta$ .

### 3 Herbrand Models

In this section we present *CRWLF*-Herbrand algebras and a fix point operator for computing the least Herbrand model of a program. We assume the reader has familiarity with basic concepts of model theory on functional and logic programming (see [3, 7] for more details), but now we point up some notions.

Given  $S$ , a *partially ordered set* (in short, *poset*) with a least element *bottom*  $\perp$  (equipped with a partial order  $\leq$ ), the set of all totally defined elements of  $S$  will be noted by  $Def(S)$ . We write  $\mathcal{C}(S)$ ,  $\mathcal{I}(S)$  for the sets of cones and ideals of  $S$ , respectively. The set  $\bar{S} =_{def} \mathcal{I}(S)$  denotes the *ideal completion* of  $S$ , which is also a *poset* under the *set-inclusion* ordering  $\subseteq$ , and there is a natural mapping for each  $x \in S$  into the principal ideal generated by  $x$ ,  $\langle x \rangle =_{def} \{y \in S : y \leq x\} \in \bar{S}$ . Furthermore,  $\bar{S}$  is a *cpo* (i.e. every directed set  $D \subseteq \bar{S}$  has a least upper bound) whose finite elements are the principal ideals  $\langle x \rangle$ ,  $x \in S$ .

**Definition 2 (Herbrand Algebras).** *For any given signature  $\Sigma$ , a Herbrand algebra  $\mathcal{H}$  is an algebraic structure of the form  $\mathcal{H} = (CTerm_{\perp, F}, \{f^{\mathcal{H}}\}_{f \in FS})$  where  $CTerm_{\perp, F}$  is a poset with the approximation ordering  $\leq$  and  $f^{\mathcal{H}} \in [CTerm_{\perp, F}^1 \rightarrow_n CTerm_{\perp, F}]$  for  $f \in FS^1$ , where  $[D \rightarrow_n E] =_{def} \{f : D \rightarrow \mathcal{C}(E) \mid \forall u, u' \in D : (u \leq u' \Rightarrow f(u) \subseteq f(u'))\}$ . From the set  $\{f^{\mathcal{H}}\}_{f \in FS}$ , we can distinguish the deterministic functions  $f \in FS^n$ , holding that  $f^{\mathcal{H}} \in [CTerm_{\perp, F}^n \rightarrow_d CTerm_{\perp, F}]$  where  $[D \rightarrow_d E] =_{def} \{f \in [D \rightarrow_n E] \mid \forall u \in D : f(u) \in \mathcal{I}(E)\}$ . Finally, the elements of  $Def(\mathcal{H})$  are the elements of  $CTerm_{\mathcal{F}}$ .*

**Definition 3 (Herbrand Denotation).** *Let  $\mathcal{H}$  be a Herbrand algebra, a valuation over  $\mathcal{H}$  is any mapping  $\eta : \mathcal{V} \rightarrow CTerm_{\perp, F}$ , and we say that  $\eta$  is totally defined iff  $\eta(X) \in Def(\mathcal{H})$  for all  $X \in \mathcal{V}$ . We denote by  $Val(\mathcal{H})$  the set of all valuations, and by  $DefVal(\mathcal{H})$  the set of all totally defined valuations. The evaluation of an  $e \in Term_{\perp, F}$  in  $\mathcal{H}$  under  $\eta$  yields  $\llbracket e \rrbracket^{\mathcal{H}} \eta \in \mathcal{C}(CTerm_{\perp, F})$  which is defined recursively as follows:*

- $\llbracket \perp \rrbracket^{\mathcal{H}} \eta =_{def} \langle \perp \rangle$ ,  $\llbracket \mathcal{F} \rrbracket^{\mathcal{H}} \eta =_{def} \langle \mathcal{F} \rangle$  and  $\llbracket X \rrbracket^{\mathcal{H}} \eta =_{def} \langle \eta(X) \rangle$ , for  $X \in \mathcal{V}$ .
- $\llbracket c(e_1, \dots, e_n) \rrbracket^{\mathcal{H}} \eta =_{def} \langle c(\llbracket e_1 \rrbracket^{\mathcal{H}} \eta, \dots, \llbracket e_n \rrbracket^{\mathcal{H}} \eta) \rangle$  for all  $c \in DC^n$ .
- $\llbracket f(e_1, \dots, e_n) \rrbracket^{\mathcal{H}} \eta =_{def} f^{\mathcal{H}}(\llbracket e_1 \rrbracket^{\mathcal{H}} \eta, \dots, \llbracket e_n \rrbracket^{\mathcal{H}} \eta)$ , for all  $f \in FS^n$ .

Due to non-determinism, the evaluation of an expression yields a cone rather than an element. It can be proved that for any  $e \in Term_{\perp, F}$  and  $\eta \in Val(\mathcal{H})$  then  $\llbracket e \rrbracket^{\mathcal{H}} \eta \in \mathcal{I}(CTerm_{\perp, F})$  if  $f^{\mathcal{H}}$  is deterministic for every  $f$  occurring in  $e$ , and  $\llbracket e \rrbracket^{\mathcal{H}} \eta \in \mathcal{I}(CTerm_{\mathcal{F}})$  if  $e \in Term_{\mathcal{F}}$  and  $\eta \in DefVal(\mathcal{H})$ .

**Definition 4 (Poset of Herbrand Algebras).** *We can define a partial order over the Herbrand algebras as follows: given  $\mathcal{A}$  and  $\mathcal{B}$ , then  $\mathcal{A} \leq \mathcal{B}$  iff  $f^{\mathcal{A}}(t_1, \dots, t_n) \subseteq f^{\mathcal{B}}(t_1, \dots, t_n)$  for every  $f \in FS^n$  and  $t_i \in CTerm_{\perp, F}$ ,  $1 \leq i \leq n$ . In such a way that the Herbrand algebras with this order are a poset with bottom.*

Moreover, it can be proved that the ideal completion of this poset is a cpo, called  $\mathcal{HALG}$ , and  $\llbracket \cdot \rrbracket$  is continuous in  $\mathcal{HALG}$ .

**Definition 5 (Herbrand Models).** Given a program  $\mathcal{P}$  and a Herbrand algebra  $\mathcal{H}$ . We define:

- $\mathcal{H}$  satisfies a SAS  $e \triangleleft C$  under a valuation  $\eta$  (in symbols,  $(\mathcal{H}, \eta) \models e \triangleleft C$ ) iff  $\llbracket t \rrbracket^{\mathcal{H}} \eta \in \llbracket e \rrbracket^{\mathcal{H}} \eta$  for every  $t \in C$ .
- $\mathcal{H}$  satisfies a joinability  $e \bowtie e'$  under a valuation  $\eta$  (in symbols,  $(\mathcal{H}, \eta) \models e \bowtie e'$ ) iff there exist  $t \in \llbracket e \rrbracket^{\mathcal{H}} \eta \cap CTerm_{\perp, F}$  and  $t' \in \llbracket e' \rrbracket^{\mathcal{H}} \eta \cap CTerm_{\perp, F}$  such that  $t \downarrow t'$ .
- $\mathcal{H}$  satisfies a divergence  $e \diamond e'$  under a valuation  $\eta$  (in symbols,  $(\mathcal{H}, \eta) \models e \diamond e'$ ) iff there exist  $t \in \llbracket e \rrbracket^{\mathcal{H}} \eta \cap CTerm_{\perp, F}$  and  $t' \in \llbracket e' \rrbracket^{\mathcal{H}} \eta \cap CTerm_{\perp, F}$  such that  $t \uparrow t'$ .
- $\mathcal{H}$  satisfies a failure of joinability  $e \nbowtie e'$  under a valuation  $\eta$  (in symbols,  $(\mathcal{H}, \eta) \models e \nbowtie e'$ ) iff for every  $t \in \llbracket e \rrbracket^{\mathcal{H}} \eta \cap CTerm_{\perp, F}$  and  $t' \in \llbracket e' \rrbracket^{\mathcal{H}} \eta \cap CTerm_{\perp, F}$ , then  $t \not\downarrow t'$  holds.
- $\mathcal{H}$  satisfies a failure of divergence  $e \not\diamond e'$  under a valuation  $\eta$  (in symbols,  $(\mathcal{H}, \eta) \models e \not\diamond e'$ ) iff for every  $t \in \llbracket e \rrbracket^{\mathcal{H}} \eta \cap CTerm_{\perp, F}$  and  $t' \in \llbracket e' \rrbracket^{\mathcal{H}} \eta \cap CTerm_{\perp, F}$ , then  $t \not\uparrow t'$  holds.
- $\mathcal{H}$  satisfies a rule  $f(\bar{t}) \rightarrow r \Leftarrow C \in \mathcal{P}$  iff
  - every valuation  $\eta$  such that  $(\mathcal{H}, \eta) \models C$  verifies  $\llbracket f(\bar{t}) \rrbracket^{\mathcal{H}} \eta \supseteq \llbracket r \rrbracket^{\mathcal{H}} \eta$
  - every valuation  $\eta$  such that, for some  $i \in \{1, \dots, n\}$ ,  $l_i$  and  $t_i$  have a  $DC \cup \{F\}$ -clash, where  $l_i \in \llbracket s_i \rrbracket^{\mathcal{H}} \eta$ , verifies  $F \in \llbracket f(\bar{s}) \rrbracket^{\mathcal{H}} \eta$
  - every valuation  $\eta$  such that there exists  $e_i \diamond e'_i \in C$  such that  $(\mathcal{H}, \eta) \models e_i \diamond e'_i$  verifies  $F \in \llbracket f(\bar{t}) \rrbracket^{\mathcal{H}} \eta$
- $\mathcal{H}$  is a model of  $\mathcal{P}$  (in symbols,  $\mathcal{H} \models \mathcal{P}$ ) iff  $\mathcal{H}$  satisfies all the rules in  $\mathcal{P}$ .

**Definition 6 (Fix Point Operator).** Given a Herbrand algebra  $\mathcal{A}$ , and  $f \in FS$ , we define the fix point operator as:

$$\begin{aligned}
T_{\mathcal{P}}(\mathcal{A}, f)(\bar{s}) =_{def} & \{ \llbracket r \rrbracket_{\eta}^{\mathcal{A}} \mid \text{if there exist } f(\bar{t}) \rightarrow r \Leftarrow C \in \mathcal{P}, \text{ and } \eta \in Val(\mathcal{A}) \\
& \text{such that } s_i \in \llbracket t_i \rrbracket_{\eta}^{\mathcal{A}} \text{ and } (\mathcal{A}, \eta) \models C \} \\
& \cup \{ F \mid \text{if there exists } f(\bar{t}) \rightarrow r \Leftarrow C \in \mathcal{P}, \text{ such that for some} \\
& \quad i \in \{1, \dots, n\}, s_i \text{ and } t_i \text{ have a } DC \cup \{F\}\text{-clash} \} \\
& \cup \{ F \mid \text{if there exist } f(\bar{t}) \rightarrow r \Leftarrow C \in \mathcal{P}, \text{ and } \eta \in Val(\mathcal{A}) \text{ such} \\
& \quad \text{that } s_i \in \llbracket t_i \rrbracket_{\eta}^{\mathcal{A}} \text{ and } (\mathcal{A}, \eta) \models e \diamond e' \text{ for some } e \diamond e' \in C \} \\
& \cup \{ \perp \mid \text{otherwise} \}
\end{aligned}$$

Given  $\mathcal{A} \in \mathcal{HALG}$ , there exists an unique  $\mathcal{B} \in \mathcal{HALG}$  denoted by  $T_{\mathcal{P}}(\mathcal{A})$  such that  $f^{\mathcal{B}}(t_1, \dots, t_n) = T_{\mathcal{P}}(\mathcal{A}, f)(t_1, \dots, t_n)$  for every  $f \in FS^n$  and  $t_i \in CTerm_{\perp, F}$ ,  $1 \leq i \leq n$ . The following result which characterizes the least Herbrand model can be ensured.

**Theorem 1.** The fix point operator  $T_{\mathcal{P}}$  is continuous in  $\mathcal{HALG}$  and satisfies:

1. For every  $\mathcal{A} \in \mathcal{HALG}$ :  $\mathcal{A} \models \mathcal{P}$  iff  $T_{\mathcal{P}}(\mathcal{A}) \leq \mathcal{A}$ .
2.  $T_{\mathcal{P}}$  has a least fix point  $\mathcal{M}_{\mathcal{P}} = \mathcal{H}_{\mathcal{P}}^{\omega}$  where  $\mathcal{H}_{\mathcal{P}}^0$  is the bottom in  $\mathcal{HALG}$  and  $\mathcal{H}_{\mathcal{P}}^{k+1} = T_{\mathcal{P}}(\mathcal{H}_{\mathcal{P}}^k)$
3.  $\mathcal{M}_{\mathcal{P}}$  is the least Herbrand model of  $\mathcal{P}$ .

Moreover, we can see that satisfaction in  $\mathcal{M}_{\mathcal{P}}$  can be characterized in terms of  $\vdash_{CRWLF}$  provability: for any constraint  $\varphi$ ,  $\mathcal{P} \vdash_{CRWLF} \varphi$  iff  $(\mathcal{M}_{\mathcal{P}}, \eta) \models \varphi$ , for all  $\eta \in DefVal(\mathcal{H})$ . Therefore,  $\vdash_{CRWLF}$  is sound and complete w.r.t. the Herbrand models, and thus the Herbrand model  $\mathcal{M}_{\mathcal{P}}$  can be regarded as the intended (canonical) model of a program  $\mathcal{P}$ .

## 4 Magic Transformation for Negative Constraints

In order to present our new magic transformation, first we will show the main ideas of the presented one in [1] by using the following example where the proposed goal has as solution  $X = s(s(Z))$ ,  $Y = s(0)$ .

Original Program	Goal
(1) $f(s(X_1), X_2) \rightarrow X_2 \Leftarrow h(X_1, X_2) \bowtie 0, p(X_2) \bowtie s(0)$ .	: $\neg f(g(X), Y) \bowtie s(0)$ .
(2) $g(s(X_3)) \rightarrow s(g(X_3))$ .	
(3) $h(s(X_4), s(0)) \rightarrow 0$ .	
(4) $p(s(0)) \rightarrow s(0)$ .	
Transformed Program	Goal
Filtered Rules	: $\neg f(mg\_g^H(X), Y) \bowtie s(0)$ .
(1) $f(s(X_1), X_2) \rightarrow X_2 \Leftarrow mg\_f^P(s(X_1), X_2) \bowtie true, h(X_1, X_2) \bowtie 0, p(X_2) \bowtie s(0)$ .	
(2) $g(s(X_3)) \rightarrow s(g(X_3)) \Leftarrow mg\_g^P(s(X_3)) \bowtie true$ .	
(3) $h(s(X_4), s(0)) \rightarrow 0 \Leftarrow mg\_h^P(s(X_4), s(0)) \bowtie true$ .	
(4) $p(s(0)) \rightarrow s(0) \Leftarrow mg\_p^P(s(0)) \bowtie true$ .	
Magic Rules	
(5) $mg\_f^P(mg\_g^H(X), Y) \rightarrow true$ .	
(6) $mg\_h^P(X_5, X_6) \rightarrow mg\_f^P(s(X_5), X_6)$ .	
(7) $mg\_p^P(X_8) \rightarrow mg\_f^P(s(X_7), X_8) \Leftarrow h(X_7, X_8) \bowtie 0$ .	
(8) $mg\_f^P(s(mg\_g^H(X_9)), X_{10}) \rightarrow mg\_f^P(mg\_g^H(s(X_9)), X_{10})$ .	
(9) $mg\_h^P(s(mg\_g^H(X_{11})), X_{12}) \rightarrow mg\_h^P(mg\_g^H(s(X_{11})), X_{12})$ .	
Goal Solving Rules	
(10) $f(mg\_g^H(s(X_{13})), X_{14}) \rightarrow f(s(mg\_g^H(X_{13})), X_{14}) \Leftarrow mg\_f^P(mg\_g^H(s(X_{13})), X_{14}) \bowtie true$ .	
(11) $h(mg\_g^H(s(X_{15})), X_{16}) \rightarrow h(s(mg\_g^H(X_{15})), X_{16}) \Leftarrow mg\_h^P(mg\_g^H(s(X_{15})), X_{16}) \bowtie true$ .	

The magic transformation transforms every pair  $(\mathcal{P}, \mathcal{G})$ , where  $\mathcal{P}$  is a program and  $\mathcal{G}$  is a goal, into a pair  $(\mathcal{P}^{MG}, \mathcal{G}^{MG})$  in such a way that the transformed program  $\mathcal{P}^{MG}$ , evaluated by means of the fix point operator, computes solutions for  $\mathcal{G}^{MG}$ , which are also solutions of  $\mathcal{G}$  w.r.t. the program  $\mathcal{P}$ .

Firstly, the so-called *passing magic (boolean) functions* of the form  $mg\_f^P$ , like in logic programming, will activate the evaluation of the functions through the fix point operator (see rules (1), (2), (3) and (4) in the transformed program), whenever there exists a call, passing the arguments from head to body and conditions of every program rule for them (see rules (6) and (7) in the above program obtained from the rule (1) of the original program) by means of left-to-right sips.

Secondly, the transformation process adds magic rules for the *outermost functions* (set denoted by  $outer(\mathcal{P}, \mathcal{G})$ ), which are defined as the leftmost functions occurring either in every side of each constraint of the goal, or in the body and every side of each constraint of every program rule of the outermost functions, or in every side of each constraint of every program rule of functions occurring in the scope of an outermost function. In the above example,  $f$ ,  $h$  and  $p$ . The idea is that whenever a function is in the scope of an outermost function, every program rule for the *inner function* generates a rule, called *nesting magic rule*, for the passing magic function of the outermost one (see rule (8) generated by the nesting occurring in the goal). The head and body of every program rule of each inner function are “turned around” and introduced as arguments of head and body, respectively, of the magic rule for the outermost function, occurring at the same position where they appear in the goal, and filling the rest of arguments with fresh variables. The inner functions are substituted in the magic rules by the so-called *nesting magic constructors* of the form  $mg\_f^N$ , given that patterns in program rules must be c-terms. In order to get a lazy evaluation, the introduction of these nesting magic rules is only achieved whether the nested function is *demand*ed by some of the rules of the nesting function; that is, the

rule pattern is a c-term or it is variable occurring out of scope of a function in the body or in the constraints. Moreover, the same kind of passing magic rules must be introduced for each information passing in the rule. For instance, given that  $\mathbf{g}$  becomes an inner function for  $\mathbf{h}$  due to the information passing from the first argument of  $\mathbf{f}$  to  $\mathbf{h}$  in the rule (1), then the nesting magic rule (9) is also included.

Thirdly, every constraint  $e \bowtie e' \in \mathcal{G}$  is transformed into a new constraint wherein each inner function is replaced by nesting magic constructors. In the above example,  $\mathbf{f}(\mathbf{mg}\text{-}\mathbf{g}^{\mathbf{N}}(\mathbf{X}), \mathbf{Y}) \bowtie \mathbf{s}(0)$ . Moreover, a new rule is generated as seed from the goal of the original program (see rule (5)).

Lastly, whenever there exists a nesting, new rules will include, called *goal solving rules*, which will allow us to get the answer of the new transformed goal and they are similar to the nesting magic rules (see rules (10) and (11)).

In our new magic transformation in order to handle negative information according to *CRWLF*, we have to lazily solve the four kinds of constraints. For instance,  $\bowtie$  should be lazily solved, that is, as far as needed up to  $\downarrow$  holds, like in the following example.

<b>Original Program</b>	<b>Goal</b>
(1) $\mathbf{f}(\mathbf{X}_1) \rightarrow \mathbf{s}(\mathbf{t}(\mathbf{X}_1))$ .	$:- \mathbf{f}(\mathbf{X}) \bowtie \mathbf{g}(\mathbf{X})$ .
(2) $\mathbf{g}(\mathbf{X}_2) \rightarrow 0$ .	
(3) $\mathbf{t}(\mathbf{X}_3) \rightarrow \mathbf{s}(\mathbf{s}(0))$ .	
<b>Transformed Program</b>	<b>Goal</b>
Filtered Rules	$:- \mathbf{f}(\mathbf{X}) \bowtie \mathbf{g}(\mathbf{X})$ .
(1) $\mathbf{f}(\mathbf{X}_1) \rightarrow \mathbf{s}(\mathbf{t}(\mathbf{X}_1)) \Leftarrow \mathbf{mg}\text{-}\mathbf{f}^{\mathbf{P}}(\mathbf{X}_1) \bowtie \mathbf{true}$ .	
(2) $\mathbf{g}(\mathbf{X}_2) \rightarrow 0 \Leftarrow \mathbf{mg}\text{-}\mathbf{g}^{\mathbf{P}}(\mathbf{X}_2) \bowtie \mathbf{true}$ .	
(3) $\mathbf{t}(\mathbf{X}_3) \rightarrow \mathbf{s}(\mathbf{s}(0)) \Leftarrow \mathbf{mg}\text{-}\mathbf{t}^{\mathbf{P}}(\mathbf{X}_3) \bowtie \mathbf{true}$ .	
Magic Rules	
(4) $\mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{mg}\text{-}\mathbf{f}^{\mathbf{N}}(\mathbf{X}), \mathbf{mg}\text{-}\mathbf{g}^{\mathbf{N}}(\mathbf{X})) \rightarrow \mathbf{true}$ .	
(5) $\mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{s}(\mathbf{mg}\text{-}\mathbf{t}^{\mathbf{N}}(\mathbf{X}_4)), \mathbf{X}_5) \rightarrow \mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{mg}\text{-}\mathbf{f}^{\mathbf{N}}(\mathbf{X}_4), \mathbf{X}_5)$ .	
(6) $\mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{X}_6, 0) \rightarrow \mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{X}_6, \mathbf{mg}\text{-}\mathbf{g}^{\mathbf{N}}(\mathbf{X}_7))$ .	
(7) $\mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{s}(\mathbf{s}(0)), \mathbf{X}_8) \rightarrow \mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{mg}\text{-}\mathbf{t}^{\mathbf{N}}(\mathbf{X}_8), \mathbf{X}_8)$ .	
(8) $\mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{X}_{10}, \mathbf{X}_{11}) \rightarrow \mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{s}(\mathbf{X}_{10}), \mathbf{s}(\mathbf{X}_{11}))$ .	
(9) $\mathbf{mg}\text{-}\mathbf{f}^{\mathbf{P}}(\mathbf{X}_{12}) \rightarrow \mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{mg}\text{-}\mathbf{f}^{\mathbf{N}}(\mathbf{X}_{12}), \mathbf{X}_{13})$ .	
(10) $\mathbf{mg}\text{-}\mathbf{g}^{\mathbf{P}}(\mathbf{X}_{15}) \rightarrow \mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{X}_{14}, \mathbf{mg}\text{-}\mathbf{g}^{\mathbf{N}}(\mathbf{X}_{15}))$ .	
(11) $\mathbf{mg}\text{-}\mathbf{t}^{\mathbf{P}}(\mathbf{X}_{16}) \rightarrow \mathbf{mg}\text{-}\bowtie^{\mathbf{P}}(\mathbf{mg}\text{-}\mathbf{t}^{\mathbf{N}}(\mathbf{X}_{16}), \mathbf{X}_{17})$ .	

In this example, the function  $\mathbf{t}$  does not need to be evaluated since the *SAS*'s  $\{\mathbf{s}(\perp)\}$  and  $\{0\}$  for the functions  $\mathbf{f}$  and  $\mathbf{g}$  respectively, are enough to solve the goal  $\mathbf{f}(\mathbf{X}) \bowtie \mathbf{g}(\mathbf{X})$  which holds by DC-clash (see rule (11) of *CRWLF*).

With this aim we will consider the operators  $\bowtie$ ,  $\diamond$ ,  $\bowtie$  and  $\diamond$  as they were outermost symbols, and the functions occurring in the constraints as they were inner symbols<sup>3</sup>. It supposes the introduction of nesting magic rules for the operators and the functions become magic constructors inside of these new nesting magic rules. In the above example, the new seed is the rule (4) which nests both expressions occurring in the goal constraint and thus the nesting magic rules (5) and (6) are generated.

Moreover, the magic transformation analyzes the body of each definition rule of the nested functions  $\mathbf{f}$  and  $\mathbf{g}$  by detecting, in the case of  $\mathbf{f}$ , a constructor in the body, and thus it generates the rule (8). This rule indicates that if the evaluation of both hand-sides of the constraint generates the same outermost constructor, then the bottom-up evaluation has to continue. In the general case, magic rules

<sup>3</sup> Remark that it does not mean that the notion of outermost function will be modified in the rest of the paper.

of this kind for a given constraint will be added for every constructor out of the scope of a function either occurring in the constraint or in the body of some rule of the leftmost functions of the constraint. Finally, the rules (9), (10) and (11) will allow to recover the passing magic functions. In the general case, one of these will be added for each outermost function symbol. The bottom-up evaluation of this program is as follows:

- ⊙  $\mathcal{H}_{p, \mathcal{M}G}^0 = \perp$
- ⊙  $\mathcal{H}_{p, \mathcal{M}G}^1 = \{\text{mg\_}\bowtie^p(\text{mg\_f}^h(X), \text{mg\_g}^h(X)) \triangleleft \{\text{true}\}, \dots\}$ .
- ⊙  $\mathcal{H}_{p, \mathcal{M}G}^2 = \{\text{mg\_}\bowtie^p(X) \triangleleft \{\text{true}\}, \text{mg\_}\bowtie^p(\text{s}(\text{mg\_t}^h(X)), \text{mg\_g}^h(X)) \triangleleft \{\text{true}\}, \text{mg\_g}^p(X) \triangleleft \{\text{true}\}, \text{mg\_}\bowtie^p(\text{mg\_f}^h(X), 0) \triangleleft \{\text{true}\}, \dots\}$ .
- ⊙  $\mathcal{H}_{p, \mathcal{M}G}^3 = \{\text{f}(X) \triangleleft \{\text{s}(\perp)\}, \text{g}(X) \triangleleft \{0\}, \text{mg\_}\bowtie^p(\text{s}(\text{mg\_t}^h(X)), 0) \triangleleft \{\text{true}\}, \dots\}$ .

By following with the new magic transformation, this one is not the only modification w.r.t. [1]. Negative constraints can be satisfied not only by DC-clash but also when a failure value for functions appears. Failures of reduction can appear in the following cases: (a) failure of the condition of a rule, by rule (7) of *CRWLF*, or (b) failure in the parameter passing, by rule (8) of *CRWLF*. In the case (a), for the outermost functions, we have the same problems as logic programming, and the magic transformation must be modified in order to avoid that some magic functions cannot be evaluated to true due to the information passing including constraints with undefined functions. Our transformation will ensure that the magic functions are two-valued, that is, their *SAS*'s will include, at least, true or  $\varepsilon$ , like in the following example:

<b>Original Program</b>	<b>Goal</b>
(1) $\text{f}(X_1, X_2) \rightarrow X_2$	$:- \text{f}(X, Y) \bowtie^p \text{s}(0).$
(2) $\text{h} \rightarrow \text{h}$	
(3) $\text{p}(0) \rightarrow \text{s}(\text{s}(0)).$	
(4) $\text{p}(\text{s}(0)) \rightarrow \text{s}(0).$	
<b>Transformed Program</b>	<b>Goal</b>
<b>Filtered Rules</b>	$:- \text{f}(X, Y) \bowtie^p \text{s}(0).$
(1) $\text{f}(X_1, X_2) \rightarrow X_2$	$\triangleleft \text{mg\_f}^p(X_1, X_2) \bowtie^p \text{true}, \text{h} \bowtie^p 0, \text{p}(X_2) \bowtie^p \text{s}(0).$
(2) $\text{h} \rightarrow \text{h}$	$\triangleleft \text{mg\_h}^p \bowtie^p \text{true}.$
(3) $\text{p}(0) \rightarrow \text{s}(\text{s}(0)).$	$\triangleleft \text{mg\_p}^p(0) \bowtie^p \text{true}.$
(4) $\text{p}(\text{s}(0)) \rightarrow \text{s}(0).$	$\triangleleft \text{mg\_p}^p(\text{s}(0)) \bowtie^p \text{true}.$
<b>Magic Rules</b>	
(5) $\text{mg\_}\bowtie^p(\text{mg\_f}^h(X, Y), \text{s}(0)) \rightarrow \text{true}.$	
(6) $\text{mg\_}\bowtie^p(X_4, X_5) \rightarrow \text{mg\_}\bowtie^p(\text{mg\_f}^h(X_3, X_4), X_5)$	$\triangleleft \text{h} \bowtie^p 0, \text{p}(X_4) \bowtie^p \text{s}(0).$
(7) $\text{mg\_}\bowtie^p(\text{mg\_h}^h, X_6) \rightarrow \text{mg\_}\bowtie^p(\text{mg\_h}^h, X_6).$	
(8) $\text{mg\_}\bowtie^p(\text{s}(\text{s}(0)), X_7) \rightarrow \text{mg\_}\bowtie^p(\text{mg\_p}^h(0), X_7).$	
(9) $\text{mg\_}\bowtie^p(\text{s}(0), X_8) \rightarrow \text{mg\_}\bowtie^p(\text{mg\_p}^h(\text{s}(0)), X_8).$	
(10) $\text{mg\_f}^p(X_9, X_{10}) \rightarrow \text{mg\_}\bowtie^p(\text{mg\_f}^h(X_9, X_{10}), X_{11}).$	
(11) $\text{mg\_h}^p \rightarrow \text{mg\_}\bowtie^p(\text{mg\_h}^h, X_{12}).$	
(12) $\text{mg\_p}^p(X_{13}) \rightarrow \text{mg\_}\bowtie^p(\text{mg\_p}^h(X_{13}), X_{14}).$	
(13) $\text{mg\_}\bowtie^p(\text{mg\_h}^h, 0) \rightarrow \text{mg\_f}^p(X_{15}, X_{16}).$	
(14) $\text{mg\_}\bowtie^p(\text{mg\_p}^h(X_{18}), \text{s}(0)) \rightarrow \text{mg\_f}^p(X_{17}, X_{18}).$	

In the previous example, the proposed goal has as answers  $Y = 0$  and  $Y = \text{s}(\text{s}(Z))$ , due to the failure in the instances  $\text{p}(0) \bowtie^p \text{s}(0)$  and  $\text{p}(\text{s}(\text{s}(Z))) \bowtie^p \text{s}(0)$  of the conditions of  $\text{f}$ .

The idea consists in detecting the outermost functions in negative constraints, and for each definition rule of these functions, to avoid the left-to-right sips (see rules (13) and (14) obtained from the rule (1) of the original program).

By considering the left-to-right sips presented in [1], the rule (14) would be replaced by the rule  $\text{mg\_}\bowtie^p(\text{mg\_p}^h(X_{18}), \text{s}(0)) \rightarrow \text{mg\_f}^p(X_{17}, X_{18}) \triangleleft \text{h} \bowtie^p 0$  taking into account the constraint  $\text{h} \bowtie^p 0$  for the information passing to  $\text{p}(X_{18})$ . Given that  $\text{h}$  has as unique *SAS*  $\{\perp\}$ , then the *SAS*  $\text{mg\_p}^p(0) \triangleleft \{\text{true}\}$  cannot be generated by using the previous rule and the rule (12). Therefore, the bottom-

up evaluation could not obtain the *SAS*  $p(0) \triangleleft \{s(s(0))\}$  which allows to get the *SAS*  $f(X, 0) \triangleleft \{F\}$  which can be used to satisfy the goal. In this way, the evaluation of the program could not generate the answer  $Y = 0$ , and thus the use of left-to-right sips produces uncompleteness w.r.t. *CRWLF*.

In the case (b), for the outermost functions, the failure of reduction will be computed by means of the fix-point operator (see definition 6). In the above example, the evaluation would compute the *SAS*  $p(\widehat{s(0)}) \triangleleft \{F\}$ <sup>4</sup> by the rule (4), where  $\widehat{s(0)} =_{\text{def}} \{0, s(s(Z)), F\}$  allowing to generate the additional answer  $Y = s(s(Z))$  by applying the rule (1) of the transformed program. The bottom-up evaluation for the above program is as follows:

- $\mathcal{H}_{p, \mathcal{M}G}^0 = \perp$
- $\mathcal{H}_{p, \mathcal{M}G}^1 = \{\text{mg-}\not\triangleleft^p(\text{mg-f}^H(X, Y), s(0)) \triangleleft \{\text{true}\}, p(\widehat{0}) \triangleleft \{F\}, p(\widehat{s(0)}) \triangleleft \{F\}, p(0) \triangleleft \{F, \perp\}, p(s(0)) \triangleleft \{F, \perp\}, \dots\}$ .
- $\mathcal{H}_{p, \mathcal{M}G}^2 = \{\text{mg-}\not\triangleleft^p(X, Y) \triangleleft \{\text{true}\}, f(X, s(s(Z))) \triangleleft \{F\}, \dots\}$ .
- $\mathcal{H}_{p, \mathcal{M}G}^3 = \{\text{mg-}\not\triangleleft^p(\text{mg-h}^H, 0) \triangleleft \{\text{true}\}, \text{mg-}\not\triangleleft^p(\text{mg-p}^H(Y), s(0)) \triangleleft \{\text{true}\}, \dots\}$ .
- $\mathcal{H}_{p, \mathcal{M}G}^4 = \{\text{mg-h}^H \triangleleft \{\text{true}\}, \text{mg-p}^H(Y) \triangleleft \{\text{true}\}, \text{mg-}\not\triangleleft^p(s(s(0)), s(0)) \triangleleft \{\text{true}\}, \text{mg-}\not\triangleleft^p(s(0), s(0)) \triangleleft \{\text{true}\}, \dots\}$ .
- $\mathcal{H}_{p, \mathcal{M}G}^5 = \{p(0) \triangleleft \{F, s(s(0))\}, p(s(0)) \triangleleft \{F, s(0)\}, \dots\}$ .
- $\mathcal{H}_{p, \mathcal{M}G}^6 = \{f(X, 0) \triangleleft \{F\}, \dots\}$ .

and the instances of the goal  $f(X, 0) \not\triangleleft s(0)$ ,  $f(X, s(s(Z))) \not\triangleleft s(0)$  are satisfied in the Herbrand algebra.

However, it is not enough, given that the failures of reduction can also appear at the same cases due to inner functions, that is due to failure (c) in the condition and (d) in parameter passing of a rule, like in the following example.

Original Program	Goal
(1) $f(X_1) \rightarrow s(X_1)$ .	$:-f(g(X)) \not\triangleleft s(0)$ .
(2) $g(s(0)) \rightarrow 0$	$\Leftarrow p(0) \triangleright 0$ .
(3) $p(0) \rightarrow s(s(0))$ .	
Transformed Program	Goal
Filtered Rules	$:-f(\text{mg-g}^H(X)) \not\triangleleft s(0)$ .
(1) $f(X_1) \rightarrow s(X_1) \Leftarrow \text{mg-f}^P(X_1) \triangleright \text{true}$ .	
(2) $g(s(0)) \rightarrow 0 \Leftarrow \text{mg-g}^P(s(0)) \triangleright \text{true}, p(0) \triangleright 0$ .	
(3) $p(0) \rightarrow s(s(0)) \Leftarrow \text{mg-p}^P(0) \triangleright \text{true}$ .	
Magic Rules	
(4) $\text{mg-}\not\triangleleft^P(\text{mg-f}^H(\text{mg-g}^H(X)), s(0)) \rightarrow \text{true}$ .	
(5) $\text{mg-}\not\triangleleft^P(s(X_2), X_3) \rightarrow \text{mg-}\not\triangleleft^P(\text{mg-f}^H(X_2), X_3)$ .	
(6) $\text{mg-}\not\triangleleft^P(s(s(0)), X_4) \rightarrow \text{mg-}\not\triangleleft^P(\text{mg-p}^H(0), X_4)$ .	
(7) $\text{mg-}\not\triangleleft^P(\text{mg-p}^H(0), 0) \rightarrow \text{mg-f}^P(\text{mg-g}^H(s(0)))$ .	
(8) $\text{mg-}\not\triangleleft^P(\text{mg-f}^H(0), X_5) \rightarrow \text{mg-}\not\triangleleft^P(\text{mg-f}^H(\text{mg-g}^H(s(0))), X_5) \Leftarrow p(0) \triangleright 0$ .	
(9) $\text{mg-}\not\triangleleft^P(\text{mg-f}^H(F), X_6) \rightarrow \text{mg-}\not\triangleleft^P(\text{mg-f}^H(\text{mg-g}^H(0)), X_6)$ .	
(10) $\text{mg-}\not\triangleleft^P(\text{mg-f}^H(F), X_7) \rightarrow \text{mg-}\not\triangleleft^P(\text{mg-f}^H(\text{mg-g}^H(s(s(Z))))), X_7)$ .	
(11) $\text{mg-f}^P(X_8) \rightarrow \text{mg-}\not\triangleleft^P(\text{mg-f}^H(X_8), X_9)$ .	
(12) $\text{mg-p}^P(X_{10}) \rightarrow \text{mg-}\not\triangleleft^P(\text{mg-p}^H(X_{10}), X_{11})$ .	
Goal Solving Rules	
(13) $f(\text{mg-g}^H(s(0))) \rightarrow f(0) \Leftarrow \text{mg-f}^P(\text{mg-g}^H(s(0))) \triangleright \text{true}$ .	
(14) $f(\text{mg-g}^H(0)) \rightarrow f(F) \Leftarrow \text{mg-f}^P(\text{mg-g}^H(0)) \triangleright \text{true}$ .	
(15) $f(\text{mg-g}^H(s(s(Z)))) \rightarrow f(F) \Leftarrow \text{mg-f}^P(\text{mg-g}^H(s(s(Z)))) \triangleright \text{true}$ .	

As you can see, we have the answers  $X = 0$ ,  $X = s(0)$  and  $X = s(s(Z))$  for the proposed goal. In this example, the failure of reduction in  $g(X)$  which is nested by  $f$ , is due to the reasons (c): failure in the condition of  $g(p(0) \triangleright 0)$  and (d): failure in the parameter passing ( $g(0)$  and  $g(s(s(Z)))$ ) w.r.t. the head  $g(s(0))$  of the rule (2). The case (c) is handled by the magic rules for the outermost functions since the condition of the rules for the inner functions will appear as condition of these magic rules (see rule (8)). This failure generates  $\text{mg-}\not\triangleleft(\text{mg-f}^H(0), s(0)) \triangleleft \{F\}$  by

<sup>4</sup>  $\hat{t}$  denotes the set  $\{t' \in CTerm_{\perp, F} \mid t \text{ and } t' \text{ have a } DC \cup \{F\}\text{-clash}\}$ .

applying the rule (8) which allows to obtain the answer  $\mathbf{X} = \mathbf{s}(0)$  by applying the rules (11), (1) and (13), respectively. The case (d) is handled by introducing rules for representing the failure of parameter passing of the nested functions. For instance the rules (9) and (10) are introduced by the rule (2) in order to obtain  $f(\text{mg\_g}^N(0)) \triangleleft \{\mathbf{s}(F)\}$  and  $f(\text{mg\_g}^N(\mathbf{s}(\mathbf{s}(Z)))) \triangleleft \{\mathbf{s}(F)\}$ , by using also (11) and (1) and, finally (14) and (15).

Next, we present an algorithm **Magic\_Alg** for this transformation. It uses an auxiliary algorithm **Nesting** in order to generate nesting magic and goal solving rules. They are shown in tables 2 and 3, wherein

- $\bar{t}|_i$  represents the subterm of  $\bar{t}$  at position  $i$ ;  $\bar{e}[e']_i$  represents  $\bar{e}$  replacing the subexpressions at position  $i$  by  $e'$ ;  $\text{safe}(\varphi)$  represents the subexpressions of  $\varphi$  out of the scope of a function.
- $e^N$  is defined as  $X^N =_{def} X, c(\bar{e})^N =_{def} c(\bar{e}^N)$  and  $f(\bar{e})^N =_{def} f(\overline{e^{mg^N}})$  and  $X^{mg^N} =_{def} X, c(\bar{e})^{mg^N} =_{def} c(\overline{e^{mg^N}})$  and  $f(\bar{e})^{mg^N} =_{def} mg\_f^N(\overline{e^{mg^N}})$ ; and  $e^P$  is defined as  $X^P =_{def} X, c(\bar{e})^P =_{def} c(\bar{e}^P)$  and  $f(\bar{e})^P =_{def} mg\_f^P(\overline{e^{mg^N}})$ .
- The functions  $f_{ind}$  are auxiliary ones added whenever  $f$  has nested constructors in the patterns of its rules.
- $\bar{e}$  represents the sequence of expressions to be considered;  $h(\bar{t})$  is the head of a program rule;  $f$  is a function, representing that  $\bar{e}$  is in the scope of  $f$ ; the boolean  $Nested?$  is true whenever the parameter  $f$  has been input;  $M_g$  represents the computed set of magic rules;  $P_g$  represents the computed set of program rules;  $ind$  indicates the position of  $\bar{e}$  in the scope of  $f$ ;  $G$  represents a set of triples  $(f, g, i)$  whose meaning is that the function  $g$  is nested by  $f$  at position  $i$ ;  $pos\_op$  indicates the position of  $\bar{e}$  in a constraint, which can be either left ( $\bar{e} \diamond e'$ ) or right ( $e' \diamond \bar{e}$ ) hand-side (it can take two values, 1 and 2); and  $op$  represents the operator which is being considered.

The algorithm is applied as follows where “\_” denotes arguments not needed for the calling:

```

Mg := ∅; Pg := P; G := ∅; C := ∅;
for every e◇e' ∈ G do
  if ((◇ ≡ ↯) or (◇ ≡ ≡)) then Mg := Mg ∪ {◇(e, e')^P → true};
  else Mg := Mg ∪ {◇(e, e')^P → true ≡ C^N};
endif
Magic_Alg(e, -, -, false, Mg, Pg, ind, G, 1, ◇, ((◇ ≡ ↯) or (◇ ≡ ≡)));
Magic_Alg(e', -, -, false, Mg, Pg, ind, G, 2, ◇, ((◇ ≡ ↯) or (◇ ≡ ≡)));
C := C ∪ {e◇e'}
endfor

```

Once the algorithm has been applied,  $\mathcal{P}^{MG}$  consists of  $P_g^P \cup M_g$ , where  $(f(\bar{t}) \rightarrow r \Leftarrow C)^P$  is of the form  $f(\bar{t}) \rightarrow r \Leftarrow mg\_f^P(\bar{t}) \bowtie true, C^N$ , and  $\Sigma^{MG}$  consists of  $DC \cup DC^{MG}$  and  $FS \cup FS^{MG}$  where  $DC^{MG}$  and  $FS^{MG}$  denote the set of nesting magic constructors and passing magic functions, respectively.  $\mathcal{G}^{MG}$  consists in the set of constraints  $e^N \diamond e'^N$  where  $e \diamond e' \in \mathcal{G}$ .

## 5 Soundness, Completeness and Optimality Results

Our first result about our evaluation method establishes the equivalence among both the original and the transformed program w.r.t. the given goal.

**Theorem 2 (Soundness and Completeness).**

$$\mathcal{P} \vdash_{CRWLF} \mathcal{G}\theta \Leftrightarrow \mathcal{P}^{MG} \vdash_{CRWLF} \mathcal{G}^N\theta.$$

Table 2. Magic Algorithm

```

Magic_Alg(in  $\bar{e}$  : tuple(Expression); in  $h(\bar{t})$  : Expression; in  $f$  : FunctionSymbol;
in Nested? : Bool; in/out  $M_g$  : Program; in/out  $P_g$  : Program; in/out  $ind$  : Index;
in/out  $G$  : set(tuple(FunctionSymbol, FunctionSymbol, Index)); in  $pos\_op$  : Index;
in  $op$  : Operator; in Failure? : Bool)
var  $C'$  : Constraint;
if Nested? then
   $ind := 0$ ;
  for every  $e_1, \dots, e_n$  do
    case  $e_i$  of
       $X, X \in \mathcal{V}$  :
        if  $(\bar{t}_j \equiv X)$  then
          for every  $((h, k, j) \in G)$  do
             $G := G \cup \{(f_{ind}, k, i)\}$ ;
            Nesting( $k, f, M_g, P_g, ind, i, G, pos\_op, op, Failure?$ );
          endfor;
        endif;
       $c(\bar{e}')$ ,  $c \in DC^{M_g}$  :
        for every  $f_{ind}(\bar{t}) \rightarrow r \leftarrow C \in P_g$  do
          if  $(t_i \equiv c(\bar{t}'))$  and not  $(\bar{t}' \equiv \bar{x} \text{ and } \bar{x} \cap (\text{safe}(r) \cup \text{safe}(C)) = \emptyset)$  then
             $P_g := P_g \cup \{f_{ind+1}(\bar{t}[\bar{t}']_i) \rightarrow r \leftarrow C^H, f_{ind}(\bar{x}[c(\bar{v})]_i) \rightarrow f_{ind+1}(\bar{x}[\bar{v}]_i)^H\}$ ;
             $M_g := M_g \cup \{op(Y_1, Y_2[f_{ind+1}(\bar{x}[\bar{v}]_i)]_{pos\_op})^P \rightarrow op(Y_1, Y_2[f_{ind}(\bar{x}[c(\bar{v})]_i)]_{pos\_op})^P\}$ ;
             $M_g := M_g \cup \{f_{ind+1}(\bar{v})^P \rightarrow op(Z_1, Z_2[f_{ind+1}(\bar{v})]_{pos\_op})^P\}$ ;
          endif;
          if  $(t_i \in \mathcal{V} \text{ and } t_i \in \text{safe}(r) \cup \text{safe}(C))$  then
             $P_g := P_g \cup \{f_{ind+1}(\bar{t}) \rightarrow r \leftarrow C^H, f_{ind}(\bar{x}[c(\bar{v})]_i) \rightarrow f_{ind+1}(\bar{x}[\bar{v}]_i)^H\}$ ;
             $M_g := M_g \cup \{op(Y_1, Y_2[f_{ind+1}(\bar{x}[\bar{v}]_i)]_{pos\_op})^P \rightarrow op(Y_1, Y_2[f_{ind}(\bar{x}[c(\bar{v})]_i)]_{pos\_op})^P\}$ ;
             $M_g := M_g \cup \{f_{ind+1}(\bar{v})^P \rightarrow op(Z_1, Z_2[f_{ind+1}(\bar{v})]_{pos\_op})^P\}$ ;
          endif;
        endfor;
         $ind := ind + 1$ ;
        Magic_Alg( $\bar{e}'$ ,  $h(\bar{t}), f, true, M_g, P_g, ind, G, pos\_op, op, Failure?$ );
       $k(\bar{e}')$ ,  $k \in FS$  :
        if  $((f_{ind}, k, i) \notin G)$  then
           $G := G \cup \{(f_{ind}, k, i)\}$ ; Nesting( $k, f, M_g, P_g, ind, i, G, pos\_op, op, Failure?$ );
          Magic_Alg( $mg\_k^H(\bar{e}')$ ,  $h(\bar{t}), f, true, M_g, P_g, ind, G, pos\_op, op, Failure?$ );
        endif;
    endcase;
  endfor;
else
  for every  $e_1, \dots, e_n$  do
    case  $e_i$  of
       $c(\bar{e}')$ ,  $c \in DC^H$  :
         $M_g := M_g \cup \{op(X_j, Y_j)^P \rightarrow op(c(\bar{x}), c(\bar{y}))^P, 1 \leq j \leq m\}$ ;
        Magic_Alg( $\bar{e}'$ ,  $h(\bar{t}), -, false, M_g, P_g, ind, G, pos\_op, op, Failure?$ );
       $k(\bar{e}')$ ,  $k \in FS$  :
         $M_g := M_g \cup \{k(\bar{v})^P \rightarrow op(X_1, X_2[k(\bar{v})]_{pos\_op})^P\}$ ;
        Magic_Alg( $\bar{e}'$ ,  $h(\bar{t}), k, true, M_g, P_g, ind, G, pos\_op, op, Failure?$ );
        for every  $k(\bar{s}) \rightarrow r \leftarrow C \in P_g$  do
           $M_g := M_g \cup \{op(X_1, X_2[r]_{pos\_op})^P \rightarrow op(X_1, X_2[k(\bar{s})]_{pos\_op})^P \leftarrow C^H\}$ ;
          Magic_Alg( $r, k(\bar{s}), -, false, M_g, P_g, ind, G, pos\_op, op, Failure?$ );
           $C' := \emptyset$ ;
          for every  $e \diamond e' \in C$  do
            if Failure? then  $M_g := M_g \cup \{\diamond(e, e')^P \rightarrow k(\bar{s})^P\}$ ;
            else  $M_g := M_g \cup \{\diamond(e, e')^P \rightarrow k(\bar{s})^P \leftarrow C^H\}$ ;
          endif
          Magic_Alg( $e, k(\bar{s}), -, false, M_g, P_g, ind, G, 1, \diamond, ((\diamond \equiv \not\Leftarrow) \text{ or } (\diamond \equiv \Leftarrow))$ );
          Magic_Alg( $\bar{e}', k(\bar{s}), -, false, M_g, P_g, ind, G, 2, \diamond, ((\diamond \equiv \not\Leftarrow) \text{ or } (\diamond \equiv \Leftarrow))$ );
           $C' := C' \cup \{e \diamond e'\}$ ;
        endfor;
    endcase;
  endfor;
endif;

```

The second result ensures optimality of our bottom-up evaluation method, in the sense that every computed *SAS* corresponds either with a subproof of some solution for the goal or with a function call from the goal.

**Table 3.** Nesting Transformation

<pre> Nesting(in k : FunctionSymbol; in f : FunctionSymbol; in/out M<sub>g</sub> : Program; in/out P<sub>g</sub> : Program; in/out ind : Index; in i : Index; in/out G : set(tuple(FunctionSymbol, FunctionSymbol, Index)); in pos_op : Index; in op : Operator; in Failure? : Bool) var C'' : Constraint; for every f<sub>ind</sub>(t̄) → r ← C ∈ P<sub>g</sub> do   if (t<sub>i</sub> ∉ V) or ((t<sub>i</sub> ∈ V) and (t<sub>i</sub> ∈ safe(r) ∪ safe(C))) then     for every k(ṡ) → r' ← C' ∈ P<sub>g</sub> do       M<sub>g</sub> := M<sub>g</sub> ∪ {op(Y<sub>1</sub>, Y<sub>2</sub>[f<sub>ind</sub>(X[r']<sub>i</sub>)]<sub>pos_op</sub>)<sup>P</sup> → op(Y<sub>1</sub>, Y<sub>2</sub>[f<sub>ind</sub>(X[k(ṡ)]<sub>i</sub>)]<sub>pos_op</sub>)<sup>P</sup> ← C'};       M<sub>g</sub> := M<sub>g</sub> ∪ {op(Y<sub>1</sub>, Y<sub>2</sub>[f<sub>ind</sub>(X[F]<sub>i</sub>)]<sub>pos_op</sub>)<sup>P</sup> → op(Y<sub>1</sub>, Y<sub>2</sub>[f<sub>ind</sub>(X[k(ṡ)]<sub>i</sub>)]<sub>pos_op</sub>)<sup>P</sup> ← C'};       P<sub>g</sub> := P<sub>g</sub> ∪ {f<sub>ind</sub>(X[k(ṡ)]<sub>i</sub>)<sup>N</sup> → f<sub>ind</sub>(X[r']<sub>i</sub>)<sup>N</sup> ← C'};       P<sub>g</sub> := P<sub>g</sub> ∪ {f<sub>ind</sub>(X[k(ṡ)]<sub>i</sub>)<sup>N</sup> → f<sub>ind</sub>(X[F]<sub>i</sub>)<sup>N</sup> ← C'};       MagicAlg(r', k(ṡ), true, f, M<sub>g</sub>, P<sub>g</sub>, ind, G, pos_op, op, Failure?);       C'' := ∅;     for every e◇e' ∈ C' do       if Failure? then M<sub>g</sub> := M<sub>g</sub> ∪ {◇(f<sub>ind</sub>(X[e]<sub>i</sub>), f<sub>ind</sub>(X[e']<sub>i</sub>))<sup>P</sup> → f<sub>ind</sub>(X[k(ṡ)]<sub>i</sub>)<sup>P</sup>};       else M<sub>g</sub> := M<sub>g</sub> ∪ {◇(f<sub>ind</sub>(X[e]<sub>i</sub>), f<sub>ind</sub>(X[e']<sub>i</sub>))<sup>P</sup> → f<sub>ind</sub>(X[k(ṡ)]<sub>i</sub>)<sup>P</sup> ← C''<sup>M</sup>};     endif;       MagicAlg(e, f<sub>ind</sub>(X[k(ṡ)]<sub>i</sub>), -, false, M<sub>g</sub>, P<sub>g</sub>, ind, G, 1, ◇, ((◇ ≡ ⊢) or (◇ ≡ ⊢)));       MagicAlg(e, f<sub>ind</sub>(X[k(ṡ)]<sub>i</sub>), -, false, M<sub>g</sub>, P<sub>g</sub>, ind, G, 2, ◇, ((◇ ≡ ⊢) or (◇ ≡ ⊢)));       C'' := C'' ∪ {e◇e'};     endfor;   endif; endfor; endfor; endfor; </pre>
--

We denote by  $neg(\mathcal{P}, \mathcal{G})$  the subset of  $outer(\mathcal{P}, \mathcal{G})$  containing the functions occurring in negative constraints.

**Theorem 3 (Optimality).**

- If  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} f(\bar{e})^N \triangleleft C$ ,  $C \neq \{\perp\}$ , then  $f \in outer(\mathcal{P}, \mathcal{G})$  and either:
  - there exist  $\theta$  and a proof  $\mathcal{P} \vdash_{CRWLF} G\theta$  of minimal size and a subproof of the form  $\mathcal{P} \vdash_{CRWLF} f(\bar{e}) \triangleleft C$ , or
  - there exist  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} g(\bar{e}')^N \triangleleft C'$ , a program rule instance  $g(\bar{t}) \rightarrow r \leftarrow C, e \diamond e', \dots \in [R]_{\perp, F}$ , and proofs  $\mathcal{P} \vdash_{CRWLF} e'_i \triangleleft C_i$  where  $t_i \in C_i$ , and either a proof of minimal size  $\mathcal{P} \vdash_{CRWLF} e \triangleleft C_e$  or a proof of minimal size  $\mathcal{P} \vdash_{CRWLF} e' \triangleleft C_{e'}$ , containing a subproof  $\mathcal{P} \vdash_{CRWLF} f(\bar{e}) \triangleleft C$ , and if  $g \notin neg(\mathcal{P}, \mathcal{G})$  then there exists a proof  $\mathcal{P} \vdash_{CRWLF} C$ .
- If  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} f_k(\bar{e})^N \triangleleft C$ , then there exists a proof  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} f(\bar{e}') \triangleleft C$  containing subproofs  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} e_i \triangleleft C_i$  for some  $C_i$ .
- If  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} \diamond(e, e')^P \triangleleft C_0$ , then there exist proofs  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} e^N \triangleleft C$  for some  $C$  and  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} e'^N \triangleleft C'$  for some  $C'$ .
- If  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} f(\bar{e})^P \triangleleft C_0$ , then there exists a proof  $\mathcal{P}^{\mathcal{M}\mathcal{G}} \vdash_{CRWLF} f(\bar{e})^N \triangleleft C$  for some  $C$ .

**6 Conclusions and Future Work**

In this paper, we have presented a goal-directed bottom-up evaluation for functional-logic programs with negative information. By adopting the *CRWLF*-semantics [14] for our language, we have defined Herbrand models for this semantics and a fix-point operator which computes the Herbrand model of *CRWLF*-programs. Moreover, we have modified the magic transformation presented in [1] in order to handle negative constraints avoiding the known problems related to the magic transformations. As future work we go toward the incorporation of grouping and aggregation operators in our language, as well as the investigation of an extension of the relational algebra for it. Also, we are starting the implementation of this language, which will be based on the use of traditional indexing techniques.

## References

1. J. M. Almendros-Jiménez and A. Becerra-Terón. A Framework for Goal-Directed Bottom-Up Evaluation of Functional Logic Programs. In *Proc. FLOPS'01*, LNCS 2024, pages 153–169.
2. J. M. Almendros-Jiménez, A. Becerra-Terón, and J. Sánchez-Hernández. A Computational Model for Functional Logic Deductive Databases, available in <http://www.ual.es/~jalmen>. Technical report, Universidad de Almería, 2001.
3. K. R. Apt. Logic programming. In *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 10, pages 493–574. MIT Press, 1990.
4. K. R. Apt and R. N. Bol. Logic Programming and Negation: A Survey. *JLP*, 19,20:9–71, 1994.
5. C. Beeri and R. Ramakrishnan. On the Power of Magic. *JLP*, 10(3,4):255–299, 1991.
6. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. ICLP/SLP'88*, pages 1070–1080. MIT Press.
7. J. C. González-Moreno, M. T. Hortalá-González, F. López-Fraguas, and M. Rodríguez-Artalejo. An Approach to Declarative Programming Based on a Rewriting Logic. *JLP*, 1(40):47–87, 1999.
8. M. Hanus. The Integration of Functions into Logic Programming: From Theory to Practice. *JLP*, 19,20:583–628, 1994.
9. M. Hanus. Curry, An Integrated Functional Logic Language, Version 0.7.1. Technical report, University of Kiel, Germany, June 2000.
10. D. B. Kemp, D. Srivastava, and P. J. Stuckey. Bottom-up Evaluation and Query Optimization of Well-Founded Models. *TCS*, 146(1,2):145–184, 1995.
11. R. Loogen, F. J. López-Fraguas, and M. Rodríguez-Artalejo. A Demand Driven Computation Strategy for Lazy Narrowing. In *Proc. PLILP'93*, LNCS 714, pages 184–200.
12. F. J. López-Fraguas and J. Sánchez-Hernández. Disequalities may help to Narrow. In *Proc. APPIA-GULP-PRODE'99*, pages 89–104.
13. F. J. López-Fraguas and J. Sánchez-Hernández. *TOY*: A Multiparadigm Declarative System. In *Proc. RTA'99*, LNCS 1631, pages 244–247.
14. F. J. López-Fraguas and J. Sánchez-Hernández. Proving Failure in Functional Logic Programs. In *Proc. CL'00*, LNCS 1861, pages 179–193.
15. R. Ramakrishnan. Magic Templates: A Spellbinding Approach to Logic Programs. *JLP*, 11(3,4):189–216, 1991.
16. R. Ramakrishnan, D. Srivastava, S. Sudarshan, and P. Seshadir. The CORAL Deductive Database System. *VLDB*, 3(2):161–210, 1994.
17. R. Ramakrishnan and J. Ullman. A Survey of Deductive Database Systems. *JLP*, 23(2):125–149, 1995.
18. K. Ross. Modular Stratification and Magic Sets for Datalog Programs with Negation. *ACM*, 41(6):1216–1266, 1994.
19. J. D. Ullman. Bottom-up Beats Top-down for Datalog. In *Procs. PODS'89*, pages 140–149. ACM Press.
20. J. Vaghani, K. Ramamohanarao, D. B. Kemp, Z. Somogyi, P. J. Stuckey, T. S. Leask, and J. Harland. The Aditi Deductive Database System. *VLDB*, 3(2):245–288, 1994.
21. A. van Gelder, K. Ross, and J. Schlipf. The Well-Founded Semantics for General Logic Programs. *ACM*, 38(3):620–650, 1991.