Conditional Narrowing with Constructive Negation*

María José Ramírez[1], Moreno Falaschi[2]

[1] Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia,
Camino de Vera s/n, Apdo. 22012, 46020 Valencia, Spain.

[2] Dipartimento di Informatica,
Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy

Abstract. In this paper, we present a narrower for conditional equational theories whose clauses allow disequations in their bodies (*normal theories*). Our approach deals with disequations in a constructive manner and thus allows non ground negative queries. We give a formal operational semantics for normal theories and define the notion of *completion* and *stratification* of a normal theory. We show that there exists one minimal model for the completion of a stratified normal theory. Then we prove the correctness of the operational semantics with respect to the *completion* of a *stratified* normal theory.

Keywords: Equational logic programming, constructive negation, term rewriting systems, narrowing.

1 Introduction

In this paper we are concerned with an algorithm specialized in solving equations and negated equations (*disequations*) with respect to a *normal equational theory*, i.e. a conditional equational theory which allows disequations in the conditions of the clauses (but not in the head). The ability to deal with disequations is not only of theoretical interest but also desirable in practice. Different approaches for handling disequations have been proposed. For instance, in the framework of logic programmig Colmerauer solves disequations in the algebra of rational trees for the semantic definition of Prolog II ([13]). Lassez, Maher and Marriott prove the solvability of systems of equations and disequations on the Herbrand Universe ([27]). In this paper we focus on conditional equations whose conditions may contain disequations. We solve disequations with an algorithm which follows an idea similar to that defined by Stuckey in [40]. On the other hand, our approach is different since we refer to equational theories, while [40] focuses on the use of constructive negation in Constraint Logic Programming.

* This work has been partially supported by CICYT under grant TIC 92-0793

The introduction of negation has been widely investigated in the framework of logic programming [12, 16, 26, 29] and has also been recently investigated in the framework of equational programming [5, 6, 25]. Negation as failure (NAF) is the mechanism traditionally used in logic programming to implement negation. However, NAF can only handle ground negative subgoals since it cannot generate any new bindings for query variables. *Constructive Negation* [10, 11, 40] can be used to find answers for negative subgoals in the same way as for positive ones. The basic idea is that the answer for a negative goal is simply the negation of the set of answers for its positive version. In order to evaluate $\sim Q$, a subderivation with Q as the top query is executed until it reduces to a simplified formula F. Then, a normalisation and negation procedure is applied to F to compute the simplified formula for $\sim Q$. This operational mechanism subsumes negation as failure. We propose to use this idea in the context of equational programming. This allows to solve negative non-ground queries as well as positive ones. Kaplan [25] investigates the algebraic properties of positive/negative conditional term rewriting systems and defines the class of *reducing* systems for which there exists an algebra of normal forms as *quasi-initial* model. Bachmair and Ganzinger [5, 6] consider general equational clauses and give a criterion, based on a total ordering on ground terms, to prove that a consistent clause equality theory has a unique *perfect* model [37, 38]. Moreover, [6] introduces the concept of saturation as an extension of the Knuth-Bendix completion method to transform a set of conditional equations with negative and positive conditions into a confluent system. All these approaches [5, 6, 25] consider ground systems and the results can be easily applied to negative ground queries. Our approach aims to deal with negative non-ground queries as well.

We aim to develop a system which may also be suitable for the integration of different programming paradigms, such as logic and equational programming, in a single framework. There have been several proposals [7, 14, 18, 19, 32] on this, but the treatment of negation has not been widely investigated in any of them. One way to the integration of equational and logic programming is the use of narrowing strategies, see e.g. [18, 19, 32]. Narrowing is a strategy to solve a system of equations within an equational theory $\mathcal{E}$: given an equational query, narrowing is able to find a complete set of $\mathcal{E}$-unifiers for it. However, narrowing does not handle systems which include negation.

We present an algorithm (*extended narrower*) which solves equations and disequations with respect to a normal equational theory. Equations and disequations are handled in a similar way, following the idea that a simplified form of a negated formula can be obtained from its complementary positive one. We show that the answer to a conjunction of equations and disequations in an equational theory with constructors is a complex system involving negation and quantifiers, where the symbols of equality and disequality can be syntactically interpreted.

We give a formal operational semantics for normal theories and define the notion of *completion* and *stratification* of a normal theory. These definitions have similarities with those for logic programming [3, 4, 12]. Roughly speaking, completion for us means that we replace the implications in a conditional equational theory with co-implications, so that the resulting completed theory allows negative logical consequences. Completed theories are not necessarily consistent. Thus, we introduce the concept of *stratified* theory, which essentially means introducing some syntactic restrictions to disallow recursion through negation. We show that there exists one minimal model for the completion of a stratified normal theory. Then we prove the correctness of the operational semantics with respect to the *completion* of a *stratified* normal theory.

The paper is organized as follows. Section 2 contains a short introduction to equations, conditional rewrite systems and universal unification. In Section 3, we define normal equational theories. In Section 4 we give a formal operational semantics for our algorithm. The notions of completion and stratification for normal equational theories are given in Sections 5 and 6, respectively. In Section 7 we prove the correctness of the operational semantics. Finally, Section 8 concludes and mentions further research.

2 Preliminaries

In this section we briefly review some basic concepts about equations, conditional rewrite systems and $\mathcal{E}$-unification. For any concept which is not explicitly defined the reader may refer to [21, 39].

We will use the symbols V, Σ and Π to denote the sets of variables, function symbols and predicate symbols respectively. $\tau(\Sigma)$ and $\tau(\Sigma \cup V)$ denote the sets of ground terms and terms built on Σ and V, respectively. $\tau(\Sigma)$ is usually called the Herbrand Universe ($\mathcal{H}_\Sigma$) over Σ and it will be denoted by $\mathcal{H}$. The set of variables occurring in a term t is denoted $Var(t)$. This notation extends naturally to other syntactic objects.

A *substitution* is defined as an almost identical mapping from the set of variables V into the set of terms $\tau(\Sigma \cup V)$.

Let θ be a substitution and let $W \subseteq V$ be a set of variables. We define the restriction of θ to W by $\theta \upharpoonright_W(x) = x\theta$ if $x \in W$, otherwise $\theta \upharpoonright_W(x) = x$.

A substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ can be represented by a conjunction of equations $\hat{\theta} = (x_1 = t_1, \ldots, x_n = t_n)$, where x_i is a variable and t_i is a term over $\tau(\Sigma \cup V)$. $\hat{\theta}$ is called the equational representation of θ. The empty substitution is denoted by ϵ and its equational representation is $\emptyset$. The notions of application and composition are defined in the usual way [27, 33].

A sequence of terms $t_1, \ldots, t_n$ will be denoted by $\tilde{t}$. By considering the usual representation of a term by a labelled tree, the occurrences are sequences of integers denoting an access path in a term. $O(t)$ and $\bar{O}(t)$ denote respectively the set of occurrences and non-variable occurrences of t. By $t[u \leftarrow t']$ we denote the replacement of the subterm of t at the occurrence u by the term t'. The definitions of occurrence sets, non-variable occurrence set, subterms and subterm replacement naturally extend to systems of equations [23].

A Σ-equation $s = t$ is a pair of terms s and t ($s, t \in \tau(\Sigma \cup V)$). A Horn equational Σ-theory $\mathcal{E}$ is a finite set of equational Horn clauses of the form

$l = r \ \Leftarrow e_1, \ldots, e_n$. with $n \geq 0$ where e_i is an equation, $1 \leq i \leq n$.

The Σ-equation in the head is implicitely oriented from left to right and the literals e_i in the body are ordinary non-oriented Σ-equations. A Σ-disequation is a formula $s \neq t$ where s, t are terms. The prefix Σ will be usually dropped when clear from the context.

Each Horn equational theory $\mathcal{E}$ generates a smallest congruence ($=_\mathcal{E}$) over the set of terms $\tau(\Sigma \cup V)$. We will denote by $\mathcal{H}/\mathcal{E}$ the finest partition $\tau(\Sigma)/_{=_\mathcal{E}}$ induced by $\mathcal{E}$ over the set of ground terms $\tau(\Sigma)$. A Horn equational theory $\mathcal{E}$ can be viewed as a conditional term rewriting system R, where the rules are the heads and the conditions are the respective bodies. If all clauses in $\mathcal{E}$ have an empty body then $\mathcal{E}$ and R are unconditional, otherwise they are conditional. The equational theory $\mathcal{E}$ is said to be canonical if the binary one-step rewriting relation $\rightarrow_R$ defined by R is noetherian and confluent. Confluence can be ensured by syntactical characterizations [8, 28, 32]. We now recall the concept of $\mathcal{E}$-unification, i.e. the problem of existence of a common instance of two terms under an equational theory $\mathcal{E}$.

Let $t = s$ be an equation. A substitution σ is called an $\mathcal{E}$-unifier of $t = s$ iff $\mathcal{E} \models t\sigma = s\sigma$, or equivalenty, $t\sigma =_\mathcal{E} s\sigma$. In this case, t and s are said $\mathcal{E}$-unifiable.

An $\mathcal{E}$-unification algorithm defines a procedure to solve an equation $t = s$ within the theory $\mathcal{E}$. A solution is represented by an $\mathcal{E}$-unifier. An $\mathcal{E}$-unification algorithm is complete if it generates a complete set of $\mathcal{E}$-unifiers for all $\mathcal{E}$-unifiable input terms. Complete unification algorithms are known for various theories. *Narrowing* is a complete unification algorithm for unconditional canonical theories [15, 22] and conditional canonical theories satisfying different restrictions [20, 31]. [39] presents a classification of equational theories from the point of view of the universal unification.

Let $\mathcal{E}$ be a Horn equational theory. A function symbol $f \in \Sigma$ is *irreducible* iff there is no clause $(l = r \Leftarrow e_1, \ldots, e_n) \in \mathcal{E}$ such that $l \in V$ or f occurs as the outermost function symbol in l. f is a *defined* function symbol if it is not irreducible.

In theories where the above distinction is made the signature Σ is partitioned as $\Sigma = C \biguplus F$, where C is the set of irreducible symbols and F is the set of defined function symbols.

Let $\mathcal{E}$ be a Horn equational theory and f be a defined function symbol. Then f is said to be completely defined (everywhere defined [9]) over its domain iff it does not occur in any ground term in normal form. $\mathcal{E}$ is said to be completely defined iff each defined function symbol is completely defined. In a completely defined theory the set of normal ground terms is the set of terms over C ($\tau(C)$).

A completely defined Horn equational theory $\mathcal{E}$ will be called a Horn equational theory with constructors iff a further condition is satisfied: each equation in the left hand side of the head of any clause is of the form $f(t_1, t_2, \ldots, t_n) = t$ where $f \in F$ and no defined symbols occur in any of $t_1, \ldots, t_n$.

3 Normal Equational Theories

In the following we consider conjunctions of equations and disequations whose solvability will be checked with respect to a normal equational theory $\mathcal{E}^\sim$. Roughly speaking, a normal equational theory is an equational theory whose clauses may contain negation in their body. Formally:

Definition 1. A normal equational clause is a clause of the form
$t = s \Leftarrow e_1, \ldots, e_m, d_1, \ldots, d_n.$
where the equation in the head is implicitely oriented from left to right, $e_1, \ldots, e_m$ are non-oriented equations, $d_1, \ldots, d_n$ are non-oriented disequations and $m, n \geq 0$.

Definition 2. A normal equational theory $\mathcal{E}^{\sim}$ is a finite set of normal equational clauses.

We notice that when the bodies of the clauses in the theory $\mathcal{E}^{\sim}$ do not contain disequations then $\mathcal{E}^{\sim}$ is a Horn equational theory.

Definition 3. A normal goal is a clause with no head.

Example 1. Let us consider a normal equational theory which defines the non-negative rational numbers represented as pairs of natural numbers [35]. $/, +, *$ are rational valued function symbols denoting division, addition and multiplication respectively, and $+_{\mathrm{N}}, *_{\mathrm{N}}$ are the natural valued addition and multiplication functions.

$$\begin{aligned}\mathcal{E}_1^{\sim} = \{\ & 0/m = 0 \Leftarrow m \neq 0.\\ & m/c = n/d \Leftarrow m *_{\mathrm{N}} d = c *_{\mathrm{N}} n, c \neq 0, d \neq 0.\\ & (m/c) + (n/d) = ((m *_{\mathrm{N}} d) +_{\mathrm{N}} (c *_{\mathrm{N}} n))/(c *_{\mathrm{N}} d).\\ & (m/c) * (n/d) = (m *_{\mathrm{N}} n)/(c *_{\mathrm{N}} d).\ \}\end{aligned}$$

A normal equational theory $\mathcal{E}^{\sim}$ can be viewed as a conditional term rewriting system R where the rules are the heads and the conditions (containing disequations) are the respective bodies. The basis for the treatment of these rewriting systems is provided by [25][3]. In particular, he has proved that in the case of reducing P/N systems[4] there exists a well-defined rewrite relation which can be constructed as a limit. Some properties of reducing P/N systems are also discussed in [25]. Thus, these systems are proved to be finitely terminating and the question of confluence is addressed by providing a complete critical pair criterion a la Knuth-Bendix. Another characterization of the confluence of normal equational theories can also be stated following [8, 34].

For the reasons we will state in the following section, throughout the paper we require normal equational theories to be theories with constructors.

4 Operational semantics

In this section we present our approach to solve normal goals in a normal equational theory $\mathcal{E}^{\sim}$. Our operational mechanism, $\mathcal{C}$onditional $\mathcal{N}$arrowing with $\mathcal{C}$onstructive $\mathcal{N}$egation, is an adaptation of the innermost conditional narrowing algorithm to solve negated subgoals.

[3] Such systems are called positive/negative systems (P/N systems) by Kaplan.

[4] A P/N system R is called reducing if there exists a reduction ordering $>$ such that if $t \to s \Leftarrow \tilde{B} \in R$ and σ is a substitution then $t\sigma > s\sigma, u\sigma, v\sigma, u'\sigma, v'\sigma$, for all $u = v, u' \neq v'$ in $\tilde{B}$.

4.1 Conditional Narrowing Calculus

Let us briefly recall the conditional narrowing algorithm. A goal is an expression of the form (g *with* θ) where g is a set of equations and θ is a substitution which is assumed to be restricted to the variables of the initial goal. To solve g the algorithm starts with the goal (g *with* ϵ) and tries to derive goals (recording applied substitution in the *with*-parts) until a terminal goal ($\emptyset$ *with* θ) is rechead. Each substitution θ in a terminal goal is an $\mathcal{E}$-unifier of g.

The children of a goal (g *with* θ) in a tree of narrowing (from now on, a search tree) for it are given by:

1. If g sintactically unifies with mgu σ then there is a unique child
$$\emptyset \textit{ with } \theta\sigma$$
2. Otherwise, if $u \in \bar{O}(g)$ then there is a child
$$(g[u \leftarrow r_k], \tilde{B}_k)\sigma \textit{ with } \theta\sigma$$
for each clause
$$l_k = r_k \leftarrow \tilde{B}_k$$
from $\mathcal{E}$ such that $\sigma = mgu(g/u, l_k)$[5].

Finally, we introduce our concept of frontier of a search tree which will be employed in the definition of the equational constructive negation rule.

Definition 4. [40] A *frontier* of a search tree for a goal G (or frontier of G) is the set of nodes at depth one in the search tree.

4.2 Equational Constructive Negation

We show that disequations in equational theories can be solved in a constructive way in equational theories, following a mechanism similar to that of [40], using a suitable conditional narrowing algorithm. This algorithm introduces negation and quantification in the constructed answer to a goal. The computed answer is a complex system involving elementary formulae. An *elementary formula* is either a (possibly quantified) equation ($x = t$ or $\tilde{\forall}\ x = t$) or a (possibly quantified) disequation ($x \neq t$ or $\tilde{\forall}\ x \neq t$) where x is a variable and t is a term distinct from x. A computed answer is a conjunction whose elements are elementary formulae or negated conjunctions of elementary formulae. Similarly, the goals involve throughout a computation a form which also includes negation and quantification. We extend the syntax of goals as follows:

Definition 5. An extended subgoal is recursively defined as either

– an equation,

[5] The variables in g and in the clauses are assumed to be disjoint.

– or $\sim \exists \tilde{Z}(g_1, \ldots, g_n \Box E)$, where every g_i, $i = 1, \ldots n$, is an extended subgoal, E is a complex system involving elementary formulae and $\tilde{Z}$ denotes some variables in $(g_1, \ldots, g_n \Box E)$. Notice that $(g_1, \ldots, g_n \Box E)$ is shorthand for $(g_1 \wedge \ldots \wedge g_n \wedge E)$.

The extended subgoals of the form $\sim \exists \tilde{Z}(g_1, \ldots, g_n \Box E)$ are referred to as *negated subgoals*. The rest of the subgoals, the equations, are called *non-negated subgoals*. We define the *complementary subgoal* for a negated subgoal $\sim \exists \tilde{Z}(g_1, \ldots, g_n \Box E)$ as $(g_1, \ldots, g_n \Box E)$.

A remark concerning the choice of the notation: as in constraint logic programming, we use the $\Box$ symbol in the extended subgoals since the system E will act as a constraint in our computational mechanism.

Definition 6. An extended goal is a conjunction of extended subgoals.

It is obvious to extend the syntax of a normal equational theory with bodies that are conjunctions of extended subgoals since a disequation, $t \neq s$, can be expressed as the extended subgoal $\sim (t = s \Box \emptyset)$. Similarly, a goal g can be shown as an extended goal g' where each disequation, $t \neq s$, of g has the form $\sim (t = s \Box \emptyset)$ in g'.

The conditional narrowing algorithm can be modified to solve negated subgoals in a constructive way. Roughly speaking, when a negated subgoal is selected to be solved, a sub-derivation for the complementary subgoal is invoked. Then the first level of the search tree is considered. The substitution applied in each derivation step (represented as a system of equations) is added to the node of this branch as a constraint. Thus, we consider the nodes of the search tree containing a part which represents the subgoals yet to be solved and another one (the constraint) which represents a partial answer to the goal. Then, we define the search tree at depth one of the negated subgoal as a suitable transformation of these nodes. In what follows, $g \Box E$ denotes a node of the search tree, where g is an extended goal and E is a partial answer of the initial goal. To solve a goal g_0, the algorithm starts with the extended goal $g_0 \Box \emptyset$. We will often represent the set of nodes of a search tree as a disjunction. We will use $E \cdot E'$ to denote the conjunction of the systems E and E'.

Let us define the constructive negation algorithm to solve negated subgoals with respect to normal equational theories.

Definition 7. *Equational Constructive Negation Rule*
Let $s \equiv\sim \exists \tilde{Z}(g \Box E)$ be a selected negated subgoal to be solved. Then the frontier of the search tree for s is defined as follows:

– If g only contains equations, has no narrowable occurrences and does not syntactically unify, then the frontier of the search tree for s is empty.
– If g only contains equations, has no narrowable occurrences and syntactically unifies with mgu σ then the frontier of the search tree for s has as node $(\emptyset \Box \forall \tilde{Z} \sim E \cdot \hat{\sigma})$ if $\sim \exists \tilde{Z}\ E \cdot \hat{\sigma}$ is solvable.
– Otherwise, let $(g_1 \Box E \cdot \hat{\theta}_1), \ldots, (g_m \Box E \cdot \hat{\theta}_m)$ be the frontier of the search tree for $g \Box E$, where every g_i is an extended goal and θ_i is the substitution applied in the reduction from g to g_i. Consider the disjunction:

$$\exists \tilde{Z}, \tilde{Y}_1(g_1 \Box E_1) \ \vee \ \dots \ \vee \ \exists \tilde{Z}, \tilde{Y}_m(g_m \Box E_m)$$

where E_i is $E \cdot \hat{\theta}_i$ and $\tilde{Y}_i$, $1 \leq i \leq m$, denotes the variables in $(g_i \Box E_i)$ which do not appear in $(g \Box E)$. Then apply the following rules [40]:

1. Negation of the disjunction. Apply the following De Morgan law:

$$\sim \bigvee_{j=1}^{m} \exists \tilde{Z}, \tilde{Y}_j \, (g_j \Box E_j) = \bigwedge_{j=1}^{m} \sim \exists \tilde{Z}, \tilde{Y}_j \, (g_j \Box E_j)$$

2. Transformation of each conjunction. Then, apply the rule of decomposition:

$$\sim \exists \tilde{Z}, \tilde{Y}_k \, (g_k \Box E_k) \leftrightarrow (\emptyset \Box \forall \tilde{Z}, \tilde{Y}_k \sim E_k) \ \vee \ (\sim \exists \tilde{Z}, \tilde{Y}_k \, (g_k \Box E_k))$$

Finally, take the disjunctive normal form of the formulae resulting from (2).

The general form of each node of the frontier of the search tree for a negated subgoal $\sim \exists \tilde{Z}(g \Box E)$, obtained applying the equational constructive negation rule to the frontier $\{g_1 \Box E_1, \dots, g_m \Box E_m\}$ of its complementary subgoal, is a conjunction of extended subgoals yet to be solved. In the first node of this frontier, $\emptyset \Box \forall \tilde{Z}, \tilde{Y}_1 \sim E_1 \cdots \dots \cdots \forall \tilde{Z}, \tilde{Y}_m \sim E_m$, the conjunction of subgoals to be solved is empty. Therefore, if the system $\forall \tilde{Z}, \tilde{Y}_1 \sim E_1 \cdots \dots \cdots \forall \tilde{Z}, \tilde{Y}_m \sim E_m$ is solvable then this system is an answer for the negated subgoal. Indeed, for each application of the equational constructive negation rule, the system in the first node of this frontier is a possible answer to the negated subgoal. Thus, if a negated subgoal has a solution, then it is always found in a finite number of derivation steps. The other nodes of the frontier contain subgoals to be solved and (in their $\Box$ part) information about the answers to the negated subgoal. Figure 1 shows the relationship between the frontiers of the search trees of a negated subgoal,$\sim \exists \tilde{Z}, \tilde{Y} \; g \Box E$, and its complementary one, $g \Box E$, for the particular case $m = 2$. For the sake of readability, we use the concise notation $\tilde{\exists}$ *subgoal* and $\tilde{\forall}$ *subgoal* instead of the more precise $\exists \tilde{Z}, \tilde{Y}$ *subgoal* and $\forall \tilde{Z}, \tilde{Y}$ *subgoal*.

The transformation applied in the definition of the frontier of the search tree for a negated subgoal from the frontier of the search tree for its complementary subgoal introduces negation and quantification in the answer system. We are only interested in answer systems whose equations and disequations only involve variables and constructors terms, because answers with this form can be thought of as systems of syntactic equations and disequations whose solvability can be checked using well-known algorithms (such as [30]). If the narrowing algorithm uses a strategy which selects only innermost redexes and we restrict us to equational theories with constructors any answer is a system which satisfies the above requirement.[6]

Let us define the operational mechanism to solve extended goals wrt normal equational theories.

[6] Fribourg ([16]) has shown that for canonical Horn equational theories with constructors and using an innermost conditional narrowing algorithm, any computed answer is a substitution involving only terms formed solely from constructors and variables.

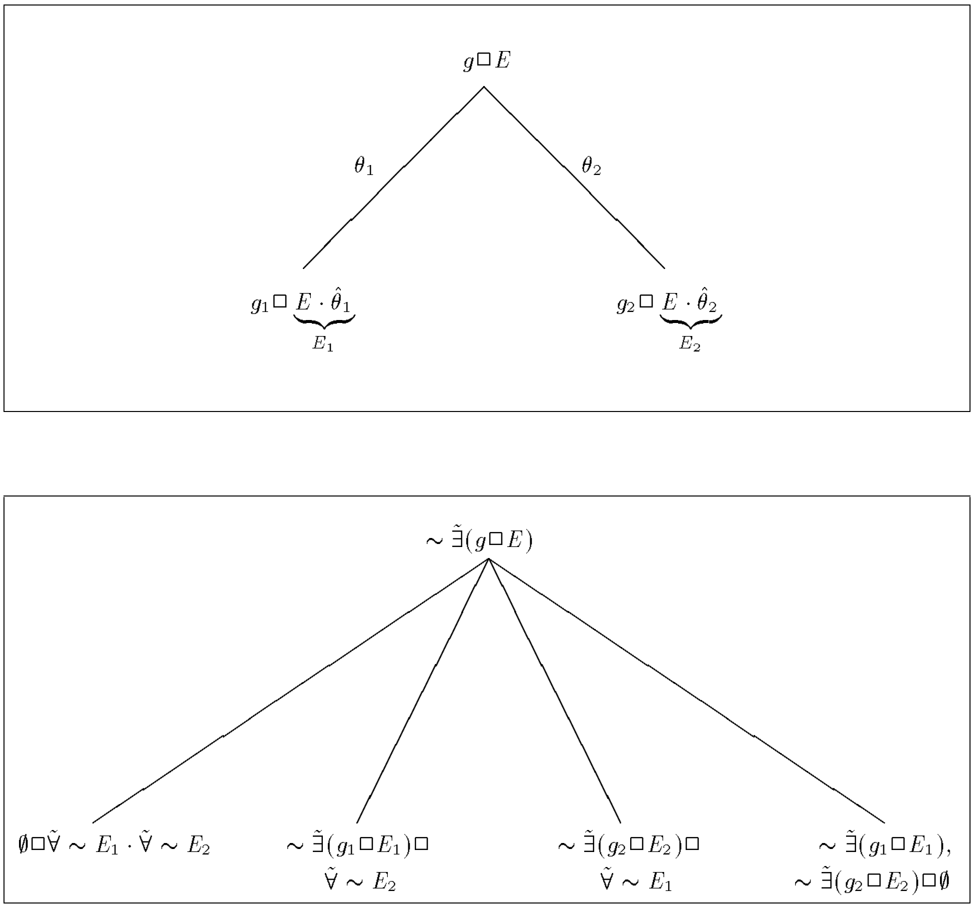

Fig. 1. Relationship between a negated subgoal and its complementary subgoal.

Definition 8. *Operational mechanism solving extended goals.*
Given a goal $(g \square E)$ where $g = g_1, \ldots, g_n$ is an extended goal, and g_j a selected extended subgoal of g to be solved, the frontier of $(g \square E)$ is given by:

- If g_j is a negated subgoal, then the equational constructive negation rule is applied. If the search tree for g_j is empty then the unique node of the frontier of $(g \square E)$ is
$$g_1, \ldots, g_{j-1}, g_{j+1}, \ldots, g_n \square E$$
Otherwise, if $(\bigvee_{i=1}^{p} f_i \square E_i)$ represents the frontier of g_j, then the disjunction $\bigvee_{i=1}^{p} g'_i \square E \cdot E_i$ represents the frontier of $(g \square E)$, where g'_i is the extended goal obtained replacing g_j by f_i in g.

– If g_j is an equation, then the frontier of $(g \Box E)$ is given by the innermost conditional narrowing algorithm.

Example 2. Consider the following normal equational theory which defines the sum of natural numbers:

$$\mathcal{E}^{\sim} = \{ \begin{array}{ll} 0 + y & = y. \\ s(x) + y & = s(x + y). \} \end{array}$$

and the normal goal $c \equiv x + y \neq 0$, represented by $\sim (x + y = 0 \Box \emptyset)$. The algorithm starts with the goal $g_0 \equiv \sim (x + y = 0 \Box \emptyset) \Box \emptyset$.

The sub-derivation for the complementary subgoal of $\sim (x + y = 0 \Box \emptyset)$, has two nodes at depth one of the search tree

$$\{(y = 0 \Box x = 0), (s(x' + y) = 0 \Box x = s(x'))\}.$$

The application of the equational constructive negation rule gives the following frontier of the search tree of the negated subgoal $\sim (x + y = 0 \Box \emptyset)$:

$$\begin{array}{l} (\emptyset \Box x \neq 0 \cdot \forall x' \; x \neq s(x')) \; \vee \\ (\sim (y = 0 \Box x = 0) \Box \forall x' \; x \neq s(x')) \; \vee \\ (\sim \exists x' (s(x' + y) = 0 \Box x = s(x')) \Box x \neq 0 \;) \; \vee \\ (\sim (y = 0 \Box x = 0), \sim \exists x' (s(x' + y) = 0 \Box x = s(x')) \Box \emptyset) \end{array}$$

The search tree at depth one for g_0 coincides with the frontier of the search tree of the negated subgoal $\sim (x + y = 0 \Box \emptyset)$, where the first node is pruned since the system $x \neq 0, \forall x' \; x \neq s(x')$ is not solvable in the Herbrand universe over the signature of the natural numbers. Thus, the three nodes of the search tree for g_0 at depth one are

$$\begin{array}{ll} (i) & \sim (y = 0 \Box x = 0) \Box \forall x' \; x \neq s(x') \\ (ii) & \sim \exists x' (s(x' + y) = 0 \Box x = s(x')) \Box x \neq 0 \\ (iii) & \sim (y = 0 \Box x = 0), \sim \exists x' (s(x' + y) = 0 \Box x = s(x')) \Box \emptyset \end{array}$$

The next derivation step for the first node has $\sim (y = 0 \Box x = 0)$ as goal to be solved. Its search tree has a unique node $\emptyset \Box \sim (x = 0, y = 0)$ since in the complementary subgoal, $y = 0 \Box x = 0$, the equation $y = 0$ has no narrowable occurrences and syntactically unifies with mgu $\sigma = \{y/0\}$.

Hence, the node of the search tree of the selected goal (i) is

$$(i.1) \; \emptyset \Box \forall x' \; x \neq s(x') \cdot \sim (x = 0, y = 0)$$

whose system of equations and disequations is the first answer to $x + y \neq 0$. Let us notice that this system represents an infinite set of solutions of the form $x = 0, y = s(n)$ where $n \in N$.

4.3 $\mathcal{C}$onditional $\mathcal{N}$arrowing with $\mathcal{C}$onstructive $\mathcal{N}$egation

In this section we present an innermost (conditional) narrowing algorithm to solve normal goals, which handles the negative subgoals following the approach presented in section 4.2. The algorithm is presented as a transition system, called $\mathcal{I}$nnermost $\mathcal{C}$onditional $\mathcal{N}$arrowing with $c$onstructive $n$egation ($\mathcal{ICN}cn$), following the structural operational semantic approach introduced by Plotkin [36].

Definition 9. An $\mathcal{ICN}cn$-state is a tuple $\langle S, L\rangle$, where S is a list of systems involving elementary formulae and L is a (possibly empty) list of narrowing goals $g\Box E$ where g is an extended goal and E is a complex system which represents a partial solution of the initial goal.

The first component of an $\mathcal{ICN}cn$-state represents the list of answers for the initial goal and the second component represents the list of the nodes of the search tree yet to be narrowed. This list is treated as a queue to emulate a breadth-first strategy.

Definition 10. Initial $\mathcal{ICN}cn$-state.
Let g be a goal. Then $\langle \emptyset, g\Box\emptyset\rangle$ is an initial $\mathcal{ICN}cn$-state.

Definition 11. Terminal $\mathcal{ICN}cn$-states.
A terminal $\mathcal{ICN}cn$-state is a state of the form $\langle S, L\rangle$ where S is non-empty.

Roughly speaking, in order to solve an extended goal g, the algorithm selects an extended subgoal of g. If a non-negated subgoal is selected, then standard narrowing is performed at one innermost occurrence and the resulting list of the nodes of the search tree is added to the current list of goals. If the selected subgoal is a negated subgoal then the equational constructive negation rule is applied to obtain its frontier which is represented as a list. Then, each element in this list replaces the selected negated subgoal in the actual goal, resulting in the new goals. The list of these goals is then added to the current list of goals. Finally, when a terminal state $\langle S, L\rangle$ is achieved, each system E of the list S is an answer for g. The $\mathcal{ICN}cn$ calculus is defined by means of a stratified transition system. $\rightarrow_{E\mathcal{CN}}$ formalizes the rules for constructive negation. $\rightarrow_f$ is built over $\rightarrow_{E\mathcal{CN}}$ and formalizes the rules for the selection of subgoals and expansion of the frontier of the search tree. $\rightarrow_{\mathcal{ICN}cn}$ is built over $\rightarrow_f$, controls the expansion of the search tree (by means of $\rightarrow_f$, rule (9)) and returns the answers for the initial subgoal. Formally:

Definition 12. $\mathcal{ICN}cn$ transition relation[7].

Unification Rules for Constructive Negation

[7] We assume that in the following rules g is non-empty

$$(1)\ \frac{\begin{array}{l} g \text{ only contains equations and} \\ \text{syntactically unifies with } mgu\ \sigma \ \wedge\ solvable(\sim \exists \tilde{Z}\ E \cdot \hat{\sigma}) \end{array}}{\sim \exists\ \tilde{Z}(g \Box E) \rightarrow_{E\mathcal{CN}} [\emptyset \Box \forall \tilde{Z}\ \sim (E \cdot \hat{\sigma})]}$$

$$(2)\ \frac{\begin{array}{l} g \text{ only contains equations and} \\ \text{syntactically unifies with } mgu\ \sigma \wedge\ \sim solvable(\sim \exists \tilde{Z}\ E \cdot \hat{\sigma}) \end{array}}{\sim \exists\ \tilde{Z}(g \Box E) \rightarrow_{E\mathcal{CN}} [\emptyset \Box false]}$$

Constructive Negation Rules

$$(3)\ \frac{\langle g \Box E \rangle \ \rightarrow_f\ L\ \wedge\ e\_c\_n(L, \sim \exists\ \tilde{Z}(g \Box E)) = L'}{\sim \exists\ \tilde{Z}(g \Box E) \rightarrow_{E\mathcal{CN}}\ L'}$$

$$(4)\ \frac{(1), (2) \text{ and } (3) \text{ do not apply}}{\sim \exists\ \tilde{Z}(g \Box E) \rightarrow_{E\mathcal{CN}} [\,]}$$

Frontier expansion Rules

$$(5)\ \frac{\begin{array}{l} e \in g\ \wedge\ u \in Inner(e) \wedge \\ s\_to\_l(\{\langle i, \sigma \rangle :\ narred(e, u, i, \sigma)\})\ = [\langle i_j, \sigma_j \rangle]_{j=1}^{n} \end{array}}{\langle g \Box E \rangle\ \rightarrow_f\ \ [(g \setminus \{e\} \cup \{e[u \leftarrow r_{i_j}]\} \cup \tilde{B}_{i_j})\sigma_j \Box E \cdot \hat{\sigma}_j]_{j=1}^{n}}$$

$$(6)\ \frac{\sim \exists\ \tilde{Z}(g' \Box E') \in g \wedge\ \sim \exists\ \tilde{Z}(g' \Box E') \rightarrow_{E\mathcal{CN}}\ \ [g'_i \Box E'_i]_{i=0}^{n}}{\langle g \Box E \rangle\ \rightarrow_f\ [g \setminus \{\sim \exists\ \tilde{Z}(g' \Box E')\} \cup \{g'_i\} \Box E \cdot E'_i]_{i=0}^{n}}$$

$$(7)\ \frac{\sim \exists\ \tilde{Z}(g' \Box E') \in g \wedge\ \sim \exists\ \tilde{Z}(g' \Box E') \rightarrow_{E\mathcal{CN}}\ \ [\emptyset \Box false]}{\langle g \Box E \rangle\ \rightarrow_f\ [\,]}$$

Unification Rule for Equations

$$(8)\ \frac{\begin{array}{l} g \text{ only contains equations and} \\ \text{syntactically unifies with } mgu\ \sigma\ \wedge\ solvable(E \cdot \hat{\sigma}) \end{array}}{\langle S, (g \Box E) \bullet L \rangle \rightarrow_{\mathcal{ICN}ccn} \langle E \cdot \hat{\sigma} \bullet S, L \rangle}$$

Narrowing Rules

$$(9)\ \frac{\langle g \Box E \rangle \rightarrow_f\ L'}{\langle S, (g \Box E) \bullet L \rangle \rightarrow_{\mathcal{ICN}ccn} \langle S,\ \ L \circ L' \rangle}$$

$$(10)\ \frac{(6) \text{ and } (7) \text{ do not apply}}{\langle S, (g \Box E) \bullet L \rangle \rightarrow_{\mathcal{ICN}ccn} \langle S, L \rangle}$$

where:

1. $Inner(e)$ is the set of innermost non-variable occurrences of e.
2. $narred(e, u, i, \sigma) \Leftrightarrow (\ u \in \bar{O}(e)\ \wedge\ \ (l_i = r_i \Leftarrow \tilde{B}_i) \in \mathcal{E}^{\sim}\ \wedge\ \sigma = mgu(e/u, l_i)\ )$, where e is an equation.
3. $solvable(E)$ is true if E is a solvable system involving elementary formulae. As we have mentioned in Section 4.2 there exist known algorithms [30] checking the solvability of these systems.
4. $e\_c\_n(L, g)$ transforms the list L which represents the frontier of the search tree for the complementary subgoal of g into another list which represents the frontier of the search tree for g, following the process described in the equational constructive negation rule (Definition 7).
5. $\bullet$ (front and rear of a list), $\circ$ (concatenation of two lists) and s_to_l (list of elements of a set) are defined as usual.

5 Completion of Normal Equational Theories

One way to reason about negative information is by introducing the concept of completion [12]. In this section we define the completion of a normal equational theory which involves adding to the theory the only if halves of the definitions of the function symbols[8] together with an equality theory.

Definition 13. *Completed definition of function symbol*

Let $\mathcal{E}^{\sim}$ be a normal equational theory. Consider the following sequence of transformation on the set of clauses in $\mathcal{E}^{\sim}$. Let

$$f(t_1, \dots, t_n) = t \Leftarrow \tilde{B}.$$

be any theory clause in $\mathcal{E}^{\sim}$. We transform this clause into

$$f(x_1, \dots, x_n) = w \Leftarrow \exists\, y_1, \dots, y_k\ (\,(x_1 = t_1)\ \wedge\ \dots\ \wedge\ (x_n = t_n)\ \wedge\ (w = t)\ \wedge\ \tilde{B}\,)$$

where $y_1, \dots, y_k$ are the variables of the original clause, and $x_1, \dots, x_n, w$ are distinct variables that do not appear anywhere in $\mathcal{E}^{\sim}$. Now all formulas obtained in the previous step with a function symbol f on the left-hand side of the head

$$\begin{gathered} f(x_1, \dots, x_n) = w \Leftarrow F_1 \\ \vdots \\ f(x_1, \dots, x_n) = w \Leftarrow F_m \end{gathered}$$

are replaced by

$$f(x_1, \dots, x_n) = w \Leftarrow F_1\ \vee\ \dots\ \vee\ F_m$$

The completed definition of f is then the formula

$$\forall\, x_1, \dots, x_n, w\ (f(x_1, \dots, x_n) = w \Leftrightarrow F_1\ \vee\ \dots\ \vee\ F_m)$$

Example 3. The completed definition of the function symbol $+$ from Example 2 is

$$\begin{aligned} \forall\, x_1, x_2, w\ \ (+(x_1, x_2) = w \Leftrightarrow\ & \exists\, x, y\ \ ((x_1 = 0) \wedge (x_2 = y) \wedge (w = y)) \vee \\ & \exists\, x, y\ \ ((x_1 = s(x)) \wedge (x_2 = y) \wedge (w = s(+(x, y))))) \end{aligned}$$

We now define the completion of a normal equational theory.

Definition 14. Let $\mathcal{E}^{\sim}$ be a normal equational theory. The completion of $\mathcal{E}^{\sim}$, denoted by $comp(\mathcal{E}^{\sim})$, is the collection of completed definitions of function symbols in $\mathcal{E}^{\sim}$ together with the Clark's equality theory (CET) where the freeness axioms are defined over irreducible function symbols.

[8] If $f(t_1, \dots, t_n) = r \Leftarrow \tilde{B}$. is a clause in an equational theory, then f is called the name of the clause. The set of all clauses with the same name f is called the definition of f.

The model theoretic semantics of a normal equational theory $\mathcal{E}^{\sim}$ is based on considering models of $comp(\mathcal{E}^{\sim})$ rather than models of $\mathcal{E}^{\sim}$ itself. Hence, the concept of a correct answer is defined as follows:

Definition 15. *Correct answer*

Let $\mathcal{E}^{\sim}$ be a normal equational theory, g a goal and E an answer system for g with respect to $\mathcal{E}^{\sim}$. We say E is a correct answer for g with respect to $comp(\mathcal{E}^{\sim})$ if for all substitution θ such that $CET \models E\theta$ then $g\theta$ is a logical consequence of $comp(\mathcal{E}^{\sim})$.

The completion will play an important part in the soundness results for the equational constructive negation.

6 Stratified Equational Theories

Hölldobler has developed a model theory for equational theories giving a fixpoint characterization ([20]). He shows that the least fixpoint of the operator T defined as a mapping between interpretations is the least Herbrand E-model[9] of the theory. However, this result can not be generalized to normal equational theories since the operator T does not need to be monotonic in presence of negation. It has been shown ([25]) that normal equational theories may have minimal models but no least model. Still, only one of these minimal models correspond to the intended meaning of the theory[10].

On the other hand, even if every normal equational theory is consistent, its completion may not be consistent. Let suppose, e.g, the following normal equational theory: $\mathcal{E}^{\sim}\ :\ a = b \Leftarrow a \neq b$, which is inconsistent since $(a = b \Leftrightarrow a \neq b)$ is a formula of $comp(\mathcal{E}^{\sim})$.

For this reason we introduce some syntactic restrictions on the theory. Similarly to logic programming [3, 4], we propose a treatment of negation which is based on the restriction of normal equational theories to be stratified. Informally, first some function symbols should be defined in terms of themselves without the use of negation and, next, some new function symbols can be defined in terms of themselves without the use of negation and in terms of the previous function symbols, possibly with the use of negation. Note that, with these restrictions, recursion by negation is not allowed. The theory $\mathcal{E}^{\sim}$ in the above example does not satisfy these conditions, hence it is not stratified.

Definition 16. *Stratified equational theories*

A normal equational theory $\mathcal{E}^{\sim}$ is called stratified if there exists a partition

$$\mathcal{E}^{\sim} = \mathcal{E}_1^{\sim} \,\dot{\cup}\, \ldots \,\dot{\cup}\, \mathcal{E}_n^{\sim}$$

9 In this context, the Herbrand base of an equational theory is the set of ground equations that can be build from the elements of the Herbrand universe. An interpretation is a subset of the Herbrand base and an E-interpretation is an interpretation which obeys the axioms of equality.

10 This coincides with to the concept of supported model in logic programming ([3, 4]).

such that for every instance of the clauses of $\mathcal{E}^{\sim}$, the following two conditions hold for $i = 1, \ldots, n$:

1. If a defined function symbol occurs in the head of a clause in $\mathcal{E}_i^{\sim}$ (except those that $\mathcal{E}_i^{\sim}$ defines) or in an equation in the body of a clause in $\mathcal{E}_i^{\sim}$, then the definition of this function symbol is contained within $\bigcup_{j \leq i} \mathcal{E}_j^{\sim}$.
2. If a defined function symbol occurs in a disequation in the body of a clause in $\mathcal{E}_i^{\sim}$, then the definition of this function symbol is contained within $\bigcup_{j<i} \mathcal{E}_j^{\sim}$.

We say that $\mathcal{E}^{\sim}$ is stratified by $\mathcal{E}_1^{\sim} \dot\cup \ldots \dot\cup \mathcal{E}_n^{\sim}$ and each $\mathcal{E}_i^{\sim}$ is called an *equational stratum* of $\mathcal{E}^{\sim}$. An equational stratum defines new function symbols in terms of themselves (positively) and in terms of the function symbols from the previous strata (possibly negatively).

Example 4. Let $\mathcal{E}^{\sim}$ be the following normal equational theory:

$$\begin{aligned} &r = a. \\ &q = b \Leftarrow r \neq b. \\ &p = a \Leftarrow q \neq a. \end{aligned}$$

Then $\mathcal{E}^{\sim}$ is stratified by $\mathcal{E}^{\sim} = \{r = a.\} \cup \{q = b \Leftarrow r \neq b.\} \cup \{p = a \Leftarrow q \neq a.\}$.

Example 5. The following normal equational theory is not stratified:

$$f(x) = a \Leftarrow x \neq a.$$

because there exists instances of this clause, for instance $f(f(a)) = a \Leftarrow f(a) \neq a$, that do not satisfy condition 2.

Since we consider stratified equational theories we give a standard fixpoint characterization of them following the fixpoint theory of non-monotonic operators which is described in [3, 4] for logic programming with negation.

Definition 17. Given a normal equational theory $\mathcal{E}^{\sim}$ stratified by $\mathcal{E}^{\sim} = \mathcal{E}_1^{\sim} \dot\cup \ldots \dot\cup \mathcal{E}_n^{\sim}$ we define a sequence of Herbrand E-interpretations by putting

$$\begin{aligned} M_1 &= T_{\mathcal{E}_1^{\sim}} \Uparrow \omega(\emptyset) \\ M_2 &= T_{\mathcal{E}_2^{\sim}} \Uparrow \omega(M_1) \\ &\vdots \\ M_n &= T_{\mathcal{E}_n^{\sim}} \Uparrow \omega(M_{n-1}) \end{aligned}$$

where clearly $M_1 \subseteq M_2 \subseteq \cdots \subseteq M_n$.

Let $M_{\mathcal{E}^{\sim}} = M_n$. The following theorem establishes the main result of this section.

Theorem 18. *Let $\mathcal{E}^\sim$ be a canonical normal equational theory stratified by $\mathcal{E}^\sim = \mathcal{E}_1^\sim \,\dot\cup\, \cdots \,\dot\cup\, \mathcal{E}_n^\sim$. Then*

1. *$M_{\mathcal{E}^\sim}$ is a Herbrand E-model of $\mathcal{E}^\sim$.*
2. *$M_{\mathcal{E}^\sim}$ is a minimal Herbrand E-model of $\mathcal{E}^\sim$.*
3. *$M_{\mathcal{E}^\sim}$ is a Herbrand E-model of $comp(\mathcal{E}^\sim)$.*

The proof of this theorem can be found in the appendix.

7 Soundness

In this section we give the soundness results of our equational constructive negation rule with respect to the completed program. Let us recall that the systems in the leaves of the search tree have a form which allows to test their solvability in CET.

First, we show that a canonical normal equational theory is a logical consequence of its completion. The proof can be found in the appendix.

Proposition 19. *Let $\mathcal{E}^\sim$ be a canonical normal equational theory. Then $\mathcal{E}^\sim$ is a logical consequence of $comp(\mathcal{E}^\sim)$.*

To state the soundness results we need the following lemmata.

Lemma 20. *If $\mathcal{E}$ is a canonical Horn equational theory with constructors and $\{G_1, \ldots, G_n\}$ is a frontier of a goal G then*

$$comp(\mathcal{E}) \models G \longleftrightarrow (\exists\, Y_1\, G_1) \;\vee\; \ldots \;\vee\; (\exists\, Y_n\, G_n)$$

where Y_i is the set of variables in G_i not in G.

Proof. The lemma follows straightforwardly from proposition 19 and from the fact that if $\{G_1, \ldots, G_n\}$ is a frontier for G then every derivation for G passes through one frontier node since the operational mechanism which is used to build the search tree is correct and complete ([17]).

Lemma 21. *If $\mathcal{E}^\sim$ is a canonical stratified normal equational theory with constructors, G is a goal and $\{G_1, \ldots, G_n\}$ is a frontier of G then*

$$comp(\mathcal{E}^\sim) \models G \longleftrightarrow (\exists\, Y_1\, G_1) \;\vee\; \ldots \;\vee\; (\exists\, Y_n\, G_n)$$

where Y_i is the set of variables in G_i not in G.

Proof. We prove the lemma by induction on the number of call to negative sub-derivations. In the base case, the selected subgoal of G is an equation and $\mathcal{E}^\sim$ is a canonical Horn equational theory with constructors, then the lemma holds by lemma 20. For the induction step, the interesting case is if the selected subgoal of

G is a negated subgoal $\sim \exists\, \tilde{Z}\ (g \Box E)$, then by the induction hypothesis the frontier $\{(g_1 \Box E_1), \ldots, (g_n \Box E_n)\}$ for the derivation of $(g \Box E)$ is such that

$$comp(\mathcal{E}^{\sim}) \models g \Box E \longleftrightarrow \exists\, Y_1\ (g_1 \Box E_1)\ \lor\ \ldots\ \lor\ \exists\, Y_n\ (g_n \Box E_n)$$

Now, if $F_1, \ldots, F_p$ are the goals returned after negating the frontier, we have

$$comp(\mathcal{E}^{\sim}) \models \sim \exists\, \tilde{Z}\ (g \Box E) \longleftrightarrow F_1\ \lor\ \ldots\ \lor\ F_p$$

Thus, every goal G is equivalent to the disjunction of its immediate children. So that, in the general case the lemma holds by building the frontier iteratively.

Theorem 22. Soundness
Let $\mathcal{E}^{\sim}$ be a canonical stratified normal equational theory with constructors and g a normal goal. If $\langle [], [g \Box \emptyset] \rangle \rightarrow_{\mathcal{ICN}ccn} \langle S, L \rangle$ then every system E_i in S is a correct answer for g.

Proof. If $\langle [], [g \Box \emptyset] \rangle \rightarrow_{\mathcal{ICN}ccn} \langle S, L \rangle$ then there exists a system $E_i \in S$ and a node in the search tree for $g \Box \emptyset$ with frontier $\{g'_1, \ldots, g'_p\}$ such that g'_j is of the form $\emptyset \Box E_i$. Then by lemma 21

$$comp(\mathcal{E}^{\sim}) \models g \Box \emptyset \longleftrightarrow \exists\, Y_1\ g'_1\ \lor\ \ldots\ \lor\ \exists\, Y_j\ (\emptyset \Box E_i)\ \lor\ \ldots\ \lor\ \exists\, Y_p\ g'_p$$

Now, since $E_i \in S$ then E_i is solvable. Thus, for all substitution θ such that $CET \models \exists\, Y_j\ E_i\theta$ the above disjuntion is true. Hence, $comp(\mathcal{E}^{\sim}) \models (g \Box \emptyset)\theta$.
Clearly then each computed answer $E_i \in S$ is a correct answer for $g \Box \emptyset$.

8 Conclusion and further research

We have presented an algorithm based on constructive negation which extends narrowing to deal with systems of equations and disequations. We have then defined the notions of *completion* and *stratified* equational theories and have proved the soundness of the operational semantics with respect to the completion of a canonical stratified normal theory with constructors. We are currently investigating the completeness of our computation rule and how to use the extended narrower as a kernel for a language which integrates equational and logic programming as an instance of the Constraint Logic Programming Scheme [1, 2, 24].

References

1. M. Alpuente and M. Falaschi. Narrowing as an Incremental Constraint Satisfaction Algorithm. In J. Maluszyński and M. Wirsing, editors, *Proc. of PLILP'91*, volume 528 of *Lecture Notes in Computer Science*, pages 111–122. Springer-Verlag, Berlin, 1991.
2. M. Alpuente, M. Falaschi, and G. Levi. Incremental Constraint Satisfaction for Equational Logic Programming. Technical report, Dipartimento di Informatica, Università di Pisa, 1991. submitted for publication.

3. K. R. Apt. Introduction to Logic Programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics. Elsevier, Amsterdam and The MIT Press, Cambridge, 1990.
4. K. R. Apt, H. Blair, and A. Walker. Towards a Theory of Declarative Knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, Los Altos, Ca., 1988.
5. L. Bachmair and H. Ganzinger. On restrictions of ordered paramodulation with simplification. In E. Y. Shapiro, editor, *Proc. 10th Int'l Conf. on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 427–441, 1990.
6. L. Bachmair and H. Ganzinger. Perfect Model Semantics for Logic Programs with Equality. In K. Furukawa, editor, *Proc. Eighth Int'l Conf. on Logic Programming*, pages 645–659. The MIT Press, Cambridge, Mass., 1991.
7. M. Bellia and G. Levi. The relation between logic and functional languages. *Journal of Logic Programming*, 3:217–236, 1986.
8. J.A. Bergstra and J.W. Klop. Conditional Rewrite Rules: confluence and termination. *Journal of Computer and System Sciences*, 32:323–362, 1986.
9. P. Bosco, E. Giovannetti, and C. Moiso. Narrowing vs. SLD-resolution. *Theoretical Computer Science*, 59:3–23, 1988.
10. D. Chan. Constructive Negation Based on the Completed Database. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth Int'l Conf. on Logic Programming*, pages 111–125. The MIT Press, Cambridge, Mass., 1988.
11. D. Chan. An Extension of Constructive Negation and its Application in Coroutining. In E. Lusk and R. Overbeck, editors, *Proc. North American Conf. on Logic Programming'89*, pages 477–493. The MIT Press, Cambridge, Mass., 1989.
12. K.L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and data bases*. Plenum Press, New York, 1978.
13. A. Colmerauer. Equations and Inequations on Finite and Infinite Trees. In *Proc. Int'l Conf. on Fifth Generation Computer Systems*, pages 85–99. ICOT, Tokio, 1984.
14. N. Dershowitz and A. Plaisted. Logic Programming cum Applicative Programming. In *Proc. First IEEE Int'l Symp. on Logic Programming*, pages 54–66. IEEE, 1984.
15. M. Fay. First Order Unification in an Equational Theory. In *4th Int'l Conf. on Automated Deduction*, pages 161–167, 1979.
16. M. Fitting. A Kripke-Kleene semantics for logic programs. In *Journal of Logic Programming*, volume 2, pages 295–312, 1985.
17. L. Fribourg. Slog: a logic programming language interpreter based on clausal superposition and rewriting. In *Proc. Second IEEE Int'l Symp. on Logic Programming*, pages 172–185. IEEE, 1985.
18. E. Giovannetti, G. Levi, C. Moiso, and C. Palamidessi. Kernel Leaf: A Logic plus Functional Language. *Journal of Computer and System Sciences*, 42, 1991.
19. J. Goguen and J. Meseguer. Eqlog: Equality, Types and Generic Modules for Logic Programming. In D. de Groot and G. Lindstrom, editors, *Logic Programming, Functions, Relations and Equations*, pages 295–363. Prentice-Hall, 1986.
20. S. Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, 1989.
21. G. Huet and D.C. Oppen. Equations and Rewrite Rules: a Survey. In *Formal Languages: perspectives and open problems*, pages 349–405. Academic Press, 1980.
22. J.M. Hullot. Canonical Forms and Unification. In *5th Int'l Conf. on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334. Springer-Verlag, Berlin, 1980.
23. H. Hussman. Unification in conditional-equational theories. Technical report, Fakultät für Mathematik und Informatik, Universität Passau, 1986.

24. J. Jaffar, J.-L. Lassez, and M.J. Maher. A logic programming language scheme. In D. de Groot and G. Lindstrom, editors, *Logic Programming, Functions, Relations and Equations*, pages 441–468. Prentice Hall, Englewood Cliffs, NJ, 1986.
25. S. Kaplan. Positive/negative conditional rewriting. In S.Kaplan and J.-P. Jouannaud, editors, *Conditional Tern Rewriting Systems*, volume 308 of *Lecture Notes in Computer Science*, pages 129–143. Springer-Verlag, Berlin, 1987.
26. K. Kunen. Negation in logic programming. *Journal of Logic Programming*, 4:289–308, 1987.
27. J.-L. Lassez, M. J. Maher, and K. Marriott. Unification Revisited. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 587–625. Morgan Kaufmann, Los Altos, Ca., 1988.
28. G. Levi, C. Palamidessi, P.G. Bosco, E. Giovannetti, and C. Moiso. A complete semantics caracterization of K-LEAF, a logic language with partial functions. In *Proc. Second IEEE Symp. on Logic In Computer Science*, pages 318–327. IEEE Computer Society Press, 1987.
29. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, 1987. Second edition.
30. M. J. Maher. Complete Axiomatizations of the Algebras of Finite, Rational and Infinite Trees. In *Proc. Third IEEE Symp. on Logic In Computer Science*, pages 348–357. Computer Science Press, New York, 1988.
31. A. Middeldorp and E. Hamoen. Counterexamples to Completeness Results for Basic Narrowing. Technical report, Centre for Mathematics and Computer Science, Amsterdam, 1991. in preparation.
32. J.J. Moreno and M. Rodriguez-Artalejo. BABEL: A Functional and Logic Programming Language based on a constructor discipline and narrowing. In I. Grabowski, P. Lescanne, and W. Wechler, editors, *Algebraic and Logic Programming*, volume 343 of *Lecture Notes in Computer Science*, pages 223–232. Springer-Verlag, Berlin, 1988.
33. W. Nutt, P. Réty, and G. Smolka. Basic narrowing revisited. *Journal of Symbolic Computation*, 7:295–317, 1989.
34. M. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1977.
35. P. Padawitz. *Computing in Horn Clause Theories*, volume 16 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1988.
36. G. Plotkin. A structured approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
37. T. Przymusinski. On the Declarative Semantics of Deductive Databases and Logic Programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, Los Altos, Ca., 1988.
38. T. Przymusinski. On the Declarative and Procedural Semantics of Logic Programs. *Journal of Automated Reasoning*, 5(2):201–228, 1989.
39. J.H. Siekmann. Universal unification. In *7th Int'l Conf. on Automated Deduction*, volume 170 of *Lecture Notes in Computer Science*, pages 1–42. Springer-Verlag, Berlin, 1984.
40. P. J. Stuckey. Constructive Negation for Constraint Logic Programming. In *Proc. Sixth IEEE Symp. on Logic In Computer Science*. IEEE Computer Society Press, 1991.