Negative Hyper-Resolution as Procedural Semantics of Disjunctive Logic Programs *

Linh Anh Nguyen

Institute of Informatics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

Abstract. We prove that negative hyper-resolution is a sound and complete procedure for answering queries in disjunctive logic programs. In our formulation, answers of queries are defined using disjunctive substitutions, which are more flexible than answer literals used in theorem proving systems.

1 Introduction

Resolution can be used not only to prove theorems but also to answer questions. This was first shown by Green in [5], where he introduced *answer literals* and a planning method using resolution. His technique has become popular in AI.

Since resolution was introduced by Robinson [13] in 1965, many refinements of resolution have been proposed by researchers in the field in order to cut down the search space and increase efficiency. One of the most important refinements of resolution is hyper-resolution, which was also introduced by Robinson [12] in the same year 1965. Hyper-resolution constructs a resolvent of a number of clauses at each step. Thus it contracts a sequence of bare resolution steps into a single inference step and eliminates interactions among intermediary resolvents, and interactions between them and other clauses.

There are many completeness results in the literature for various refinements of resolution, but these results usually derive refutation completeness, i.e. the empty clause will be derived if the input clauses are inconsistent. For question-answering systems, we want a stronger result called *answer completeness*: for every correct answer there exists a more general computed answer.

A refinement of resolution for the Horn fragment, called SLD-resolution in [1], was first described by Kowalski [6] for logic programming. It is a sound and complete procedure for answering queries in definite logic programs. In [9], Lobo et al gave a linear resolution method with a selection function, called SLO-resolution, for answering goals in disjunctive logic programs. SLO-resolution is an extension of SLD-resolution, and both of them are answer complete under any selection function.

* In J.J. Alferes and J.A. Leite (Eds.): Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004, Proceedings. LNCS 3229, pages 565–577, Springer, 2004.

SLO-resolution extends SLD-resolution in a natural way, and in our opinion, it is a potential framework for developing efficient proof procedures. However, queries and derivations formulated in SLO-resolution allow only definite answers, and in fact, SLO-resolution is answer complete only for a certain class of queries. Consider an example of [9] : given the program $P = \{p(a) \vee p(b) \leftarrow\}$ and the query $Q = \leftarrow p(x)$, there is no computed answer in SLO-resolution for $P \cup Q$, while there exists a disjunctive answer $\{\{x/a\}, \{x/b\}\}$. Of course, if we rewrite Q to $Q' = \leftarrow p(x) \vee p(y)$ then there is a computed answer $\{x/a, y/b\}$, but if the considered program is larger, it is difficult to know when and where we need to rewrite goals, and furthermore, rewriting goals is inconvenient for users.

There are also other goal oriented proof procedures proposed for disjunctive logic programming: nearHorn-Prolog procedures by Loveland [10], SLI-resolution by Lobo et al [8], and restart model elimination (RME) by Baumgartner et al [2]. The nearHorn-Prolog procedures extend SLD-resolution and Prolog style for disjunctive logic programs, but they are of interest only when the considered program contains very few non-Horn clauses. Both of SLI-resolution and RME are variants of the model elimination procedure. SLI-resolution is related to SLO-resolution, while RME is related to hyper tableaux.

In our opinion, it is very difficult for programmers to imagine behaviors of disjunctive logic *programs* as is possible when writing Prolog programs. Perhaps we should adopt the approach by Loveland and use mechanisms of theorem proving for non-Horn fragments of disjunctive logic programs. But as mentioned before, the nearHorn-Prolog procedures proposed by Loveland have advantages only for logic programs containing very few non-Horn clauses. For general cases, why don't we just use strongest theorem provers as proof procedures for disjunctive logic programming?

In this work, we formulate a negative hyper-resolution calculus as a proof procedure for disjunctive logic programming. In our formulation, every clause set can be divided into a disjunctive logic program, which consists of non-negative clauses, and a query. We define answers as disjunctive substitutions. To each goal appearing in a derivation we attach a disjunctive substitution keeping bindings of variables of the initial query. Our definition of answers is more flexible than answer literals used in theorem proving systems. In [3], Brass and Lipeck also defined disjunctive answer as a set of normal substitutions, but they did not give further properties of disjunctive substitutions as we do. Our definition of correct answers is compatible with the semantics of answer literals given by Kunen [7]. The theory of answer literals was discussed earlier in [5, 11, 4], but in those works the authors assume that answer literals appear only in one clause.

As far as we know, answer completeness of negative hyper-resolution in our setting of queries has not previously been studied. Here, we prove that negative hyper-resolution is a sound and complete procedure for answering queries in disjunctive logic programs.

This paper is organized as follows: In Section 2, we give definitions for disjunctive substitutions, disjunctive logic programs, queries, and correct answers. In Section 3, we specify a negative hyper-resolution calculus as procedural se-

mantics of disjunctive logic programs. In Section 4, we prove answer soundness of that calculus. We give a *reverse* fixpoint semantics for disjunctive logic programs in Section 5 and use it in Section 6 to prove answer completeness of the calculus. The relationship between disjunctive substitutions and answer literals is considered in Section 7. Section 8 concludes this work.

2 Preliminaries

First-order logic is considered in this work and we assume that the reader is familiar with it. We now give the most important definitions for our work.

By $\forall(\varphi)$ we denote the *universal closure* of φ, which is the closed formula obtained by adding a universal quantifier for every free variable of φ.

An *expression* is either a term or a formula. If E is an expression, then by $Var(E)$ we denote the set of all variables occurring in E.

The *Herbrand universe* U_Γ of a formula set Γ is the set of all ground terms that can be formed from the constants and function symbols in Γ : if no constants occur in Γ then some arbitrary constant is used instead.

The *Herbrand base* B_Γ of a formula set Γ is the set consisting of all ground atoms that can be formed from the predicate symbols in Γ and the terms in U_Γ. When Γ is clear from the context, for $M \subseteq B_\Gamma$, we write $\overline{M}$ to denote the set $B_\Gamma - M$.

2.1 Disjunctive Substitutions

A *normal substitution* is a finite set $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$, where $x_1, \ldots, x_n$ are different variables, $t_1, \ldots, t_n$ are terms, and $t_i \neq x_i$ for all $1 \leq i \leq n$. By ε we denote the *empty normal substitution*. The set $Dom(\theta) = \{x_1, \ldots, x_n\}$ is called the *domain* of θ. By $Ran(\theta)$ we denote the set of all variables occurring in $t_1, \ldots, t_n$. Define $Var(\theta) = Dom(\theta) \cup Ran(\theta)$. For a set X of variables, the *restriction* of θ to X, denoted by $\theta_{|X}$, is the substitution $\{x/t \mid x/t \in \theta \text{ and } x \in X\}$.

Let $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ be a normal substitution and E be an expression. Then $E\theta$, the *instance* of E by θ, is the expression obtained from E by simultaneously replacing each occurrence of the variable x_i in E by the term t_i, for $1 \leq i \leq n$.

Let $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\delta = \{y_1/s_1, \ldots, y_m/s_m\}$ be normal substitutions. Then the *composition* $\theta\delta$ of θ and δ is the substitution obtained from the set $\{x_1/t_1\delta, \ldots, x_n/t_n\delta, y_1/s_1, \ldots, y_m/s_m\}$ by deleting any binding $x_i/t_i\delta$ for which $x_i = t_i\delta$ and deleting any binding y_j/s_j for which $y_j \in \{x_1, \ldots, x_n\}$.

If θ and δ are normal substitutions such that $\theta\delta = \delta\theta = \varepsilon$, then we call them *renaming substitutions* and use θ^{-1} to denote δ (which is unique w.r.t. θ).

A *disjunctive substitution* Θ is a finite and non-empty set of normal substitutions. Define $Dom(\Theta) = \bigcup_{\theta\in\Theta} Dom(\theta)$, $Ran(\Theta) = \bigcup_{\theta\in\Theta} Ran(\theta)$, and $Var(\Theta) = Dom(\Theta) \cup Ran(\Theta)$. For $X \subseteq Dom(\Theta)$, the *restriction* of Θ to X

is denoted by $\Theta_{|X}$ and defined as $\{\theta_{|X} \mid \theta \in \Theta\}$. We treat a normal substitution θ also as the disjunctive substitution $\{\theta\}$.

If φ is a formula then $\varphi\Theta =_{def} \{\varphi\theta \mid \theta \in \Theta\}$. If Γ is a set of formulas then $\Gamma\Theta =_{def} \{\varphi\theta \mid \varphi \in \Gamma, \theta \in \Theta\}$. The *composition* $\Theta\Delta$ of disjunctive substitutions Θ and Δ is the disjunctive substitution $\{\theta\delta \mid \theta \in \Theta, \delta \in \Delta\}$.

A disjunctive substitution Θ is *more general* than Δ if there exists a normal substitution σ such that for $X = Dom(\Theta) \cup Dom(\Delta)$, $(\Theta\sigma)_{|X} \subseteq \Delta$.

As some properties of disjunctive substitutions, for an expression E and disjunctive substitutions Θ, Θ_1, Θ_2, Θ_3, we have: $\Theta\varepsilon = \varepsilon\Theta = \Theta$, $(E\Theta_1)\Theta_2 = E(\Theta_1\Theta_2)$, and $(\Theta_1\Theta_2)\Theta_3 = \Theta_1(\Theta_2\Theta_3)$.

2.2 Disjunctive Logic Programs and Queries

A *clause* is a formula of the form

$$\forall x_1 \ldots \forall x_h (A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_m)$$

where $x_1, \ldots, x_h$ are all the variables occurring in the rest of the formula, $n \geq 0$, $m \geq 0$, and A_i and B_j are atoms. We write such a clause in the form

$$A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$$

We call $A_1 \vee \ldots \vee A_n$ the *head* and $B_1 \wedge \ldots \wedge B_m$ the *body* of the clause. If $n = 0$ and $m = 0$ then the clause is *empty* and denoted by $\bot$. If $n = 0$ and $m > 0$ then the clause is a *goal*. If $n > 0$ and $m = 0$ then the clause is *positive*. If $n > 0$ then the clause is a *(disjunctive) program clause*.

A *(disjunctive) logic program* is a finite set of disjunctive program clauses. A *(disjunctive) query* is a finite set of goals.

Let P be a logic program and $Q = \{ \leftarrow \varphi_1, \ldots, \leftarrow \varphi_n \}$ be a query. We say that a disjunctive substitution Θ with $Dom(\Theta) \subseteq Var(Q)$ is a *correct answer* of $P \cup Q$ if $P \models \forall(\bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta} \varphi_i\theta)$.

For example, if $P = \{ \ p(f(x)) \vee p(g(x)) \leftarrow \ \}$ and $Q = \{ \leftarrow p(y) \ \}$, then $\Theta = \{\{y/f(x)\}, \{y/g(x)\}\}$ is a correct answer of $P \cup Q$.

In [7], Kunen characterized the semantics of answer literals used in theorem proving systems by the following theorem: Let Σ be a set of sentences, $\exists\overline{x}\,\varphi(\overline{x})$ be a sentence, and $\Sigma' = \Sigma \cup \forall(ans(\overline{x}) \leftarrow \varphi(\overline{x}))$. If each $\overline{\tau}_i$, for $i = 1, \ldots, k$, is a tuple of terms of the same length of $\overline{x}$, then $\Sigma' \models \forall(ans(\overline{\tau}_1) \vee \ldots \vee ans(\overline{\tau}_k))$ (this specifies an answer) iff $\Sigma \models \forall(\varphi(\overline{\tau}_1) \vee \ldots \vee \varphi(\overline{\tau}_k))$.

Our definition of correct answers is compatible with the semantics of answer literals by Kunen. To see the compatibility, take $\Sigma = P$, assume that $\varphi_1, \ldots, \varphi_n$ have disjoint sets of variables, and let $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$.

3 Negative Hyper-Resolution Semantics

An *informative goal* is a pair $\varphi : \Theta$, where φ is a goal and Θ is a disjunctive substitution. Informally, Θ keeps the disjunctive substitution that has been applied

to variables of the initial query in the process of deriving φ. We will ignore the word "informative" when it is clear from the context. An informative goal $\varphi : \Theta$ is said to be *ground* if φ is ground.

Let $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ be a program clause (i.e. $n > 0$) and $\varphi_1 : \Theta_1, \ldots, \varphi_n : \Theta_n$ be goals. Let $\varphi_i = \leftarrow \xi_i \wedge \zeta_i$, for $1 \leq i \leq n$, where ξ_i is a non-empty set of atoms called the *selected* atoms of φ_i. If there exists an mgu σ such that $A_i\sigma = A'_j\sigma$ for every $1 \leq i \leq n$ and every $A'_j \in \xi_i$, then we call the goal

$$\leftarrow (B_1 \wedge \ldots \wedge B_m \wedge \zeta_1 \wedge \ldots \wedge \zeta_n)\sigma : (\Theta_1 \cup \ldots \cup \Theta_n)\sigma$$

a *hyper-resolvent* of φ and $\varphi_1 : \Theta_1, \ldots, \varphi_n : \Theta_n$.

Note that "factoring" is hidden in our definition.

Before defining derivation and refutation we specify the process of standardizing variables apart. Denote the original set of variables of the language by $\mathcal{X}$, and assume that variables occurring in the given logic program, the given query, or considered correct answers all belong to $\mathcal{X}$. Let $\mathcal{X}'$ be an infinite set of variables disjoint with $\mathcal{X}$. We will use elements of $\mathcal{X}'$ for renaming variables.

Let φ be a program clause and $\varphi_1 : \Theta_1, \ldots, \varphi_n : \Theta_n$ be goals. A *standardized variant* of the set $\{\varphi, \varphi_1 : \Theta_1, \ldots, \varphi_n : \Theta_n\}$ is a set $\{\varphi\delta, \varphi_1\delta_1 : \Theta_1\delta_1, \ldots, \varphi_n\delta_n : \Theta_n\delta_n\}$ where $\delta, \delta_1, \ldots, \delta_n$ are renaming substitutions such that $Dom(\delta) = Var(\varphi)$ and $Ran(\delta) \subset \mathcal{X}'$, $Dom(\delta_i) = Var(\varphi_i) \cup Ran(\Theta_i)$ and $Ran(\delta_i) \subset \mathcal{X}'$ for all $1 \leq i \leq n$, and the sets $Ran(\delta)$, $Ran(\delta_1)$, $\ldots$, $Ran(\delta_n)$ are disjoint. Assume that standardizing variants is done by some unspecified deterministic procedure.

Let P be a logic program and Q a query. A *derivation* from $P \cup Q$ is a sequence $\varphi_1 : \Theta_1, \ldots, \varphi_n : \Theta_n$ of goals such that for each $1 \leq i \leq n$:

1. either φ_i is a clause of Q and $\Theta_i = \varepsilon$;
2. or $\varphi_i : \Theta_i$ is a hyper-resolvent of a program clause φ' and goals $\varphi'_{i,1} : \Theta'_{i,1}$, $\ldots$, $\varphi'_{i,n_i} : \Theta'_{i,n_i}$, where $\{\varphi', \varphi'_{i,1} : \Theta'_{i,1}, \ldots, \varphi'_{i,n_i} : \Theta'_{i,n_i}\}$ is a standardized variant of $\{\varphi, \varphi_{i,1} : \Theta_{i,1}, \ldots, \varphi_{i,n_i} : \Theta_{i,n_i}\}$, φ is a program clause of P, and $\varphi_{i,1} : \Theta_{i,1}, \ldots, \varphi_{i,n_i} : \Theta_{i,n_i}$ are goals from the sequence $\varphi_1 : \Theta_1, \ldots, \varphi_{i-1} : \Theta_{i-1}$.

For simplicity, Condition 2 of the above definition will be also stated as $\varphi_i : \Theta_i$ is a hyper-resolvent of a standardized variant of a program clause φ of P and standardized variants of some goals from $\varphi_1 : \Theta_1, \ldots, \varphi_{i-1} : \Theta_{i-1}$.

A *refutation* of $P \cup Q$ is a derivation from $P \cup Q$ with the last goal of the form $\bot : \Theta$. The disjunctive substitution $\Theta_{|Var(Q)}$ is called the *computed answer* of $P \cup Q$ w.r.t. that refutation.

Example 1. Let P be the program consisting of the following clauses:

(1) $s(x, a) \leftarrow p(x)$
(2) $s(x, b) \leftarrow q(x)$
(3) $p(x) \vee q(x) \leftarrow r(x)$
(4) $r(c) \leftarrow$

and let Q be the query consisting of the only following goal:

(5) $\leftarrow s(x, y)$

Here is a refutation of $P \cup Q$:

(6) $\leftarrow s(x, y) : \varepsilon$ from (5)
(7) $\leftarrow p(x_2) : \{x/x_2, y/a, x_1/x_2, y_2/a\}$ (1),(6)
(8) $\leftarrow q(x_4) : \{x/x_4, y/b, x_3/x_4, y_4/b\}$ (2),(6)
(9) $\leftarrow r(x_5) : \{\{x/x_5, y/a, x_1/x_5, y_2/a, x_2/x_5, x_6/x_5, x_7/x_5\}, \{x/x_5, y/b, x_3/x_5, y_4/b, x_4/x_5, x_6/x_5, x_7/x_5\}\}$ (3),(7),(8)
(10) $\bot : \{\{x/c, y/a, x_1/c, y_2/a, x_2/c, x_6/c, x_7/c, x_5/c, x_8/c\}, \{x/c, y/b, x_3/c, y_4/b, x_4/c, x_6/c, x_7/c, x_5/c, x_8/c\}\}$ (4),(9)

The computed answer is $\{\{x/c, y/a\}, \{x/c, y/b\}\}$.

4 Answer Soundness

In this section, we show that for every logic program P and every query Q, every computed answer of $P \cup Q$ is a correct answer of $P \cup Q$.

Lemma 1. *Let $\leftarrow \psi : \Theta$ be a hyper-resolvent of a program clause φ and goals $\leftarrow \varphi_1 : \Theta_1, \ldots, \leftarrow \varphi_n : \Theta_n$ with σ being the involved mgu. Let M be a model of φ. Then $M \vDash \bigvee_{i=1}^{n}(\psi \rightarrow \varphi_i\sigma)$. In particular, if ψ is empty then $M \vDash \bigvee_{i=1}^{n} \varphi_i\sigma$.*

Proof. Let $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ and $\varphi_i = \xi_i \wedge \zeta_i$, where ξ_i is the set of selected atoms of $\leftarrow \varphi_i$, for $1 \leq i \leq n$. Then $\psi = (B_1 \wedge \ldots \wedge B_m \wedge \zeta_1 \wedge \ldots \wedge \zeta_n)\sigma$. Let V be an arbitrary variable assignment. Suppose that $M, V \vDash \psi$. Because M is a model of φ and $M, V \vDash \psi$, it follows that $M, V \vDash (A_1 \vee \ldots \vee A_n)\sigma$. Hence $M, V \vDash \bigvee_{i=1}^{n}(A_i \wedge \zeta_i)\sigma$, since $M, V \vDash \psi$. Thus $M, V \vDash \bigvee_{i=1}^{n} \varphi_i\sigma$. Since V is an arbitrary variable assignment, we conclude that $M \vDash \bigvee_{i=1}^{n}(\psi \rightarrow \varphi_i\sigma)$.

Lemma 2. *Let P be a logic program, $Q = \{\leftarrow \varphi_1, \ldots, \leftarrow \varphi_n\}$ be a query, and $\leftarrow \psi : \Theta$ be the last goal in a derivation from $P \cup Q$. Let M be a model of P. Then $M \vDash \bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta}(\psi \rightarrow \varphi_i\theta)$. In particular, if ψ is empty then $M \vDash \bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta} \varphi_i\theta$.*

Proof. We prove this lemma by induction on the length of the derivation. The case when $\leftarrow \psi$ is a clause of Q and $\Theta = \varepsilon$ is trivial. Suppose that $\leftarrow \psi : \Theta$ is derived as a hyper-resolvent of a standardized variant of a program clause φ and standardized variants of goals $\leftarrow \psi_1 : \Theta_1, \ldots, \leftarrow \psi_m : \Theta_m$. Let σ be the involved mgu and $\delta, \delta_1, \ldots, \delta_m$ be the involved renaming substitutions. By the inductive assumption, we have $M \vDash \bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta_j}(\psi_j \rightarrow \varphi_i\theta)$ for all $1 \leq j \leq m$. Thus $M \vDash \bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta_j}(\psi_j\delta_j\sigma \rightarrow \varphi_i\theta\delta_j\sigma)$, and hence $M \vDash \bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta_j\delta_j\sigma}(\psi_j\delta_j\sigma \rightarrow \varphi_i\theta)$, for all $1 \leq j \leq m$. Note that $\Theta_j\delta_j\sigma \subseteq \Theta$. By Lemma 1, $M \vDash \bigvee_{j=1}^{m}(\psi \rightarrow \psi_j\delta_j\sigma)$. These two assertions together imply that $M \vDash \bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta}(\psi \rightarrow \varphi_i\theta)$.

Theorem 1 (Soundness). *Let P be a logic program, Q a query, and Θ a computed answer of $P \cup Q$. Then Θ is a correct answer of $P \cup Q$.*

Proof. Let $Q = \{\leftarrow \varphi_1, \ldots, \leftarrow \varphi_n\}$ and let $\bot : \Theta'$ be the last goal in a refutation of $P \cup Q$ such that $\Theta = \Theta'_{|Var(Q)}$. Let M be an arbitrary model of P. By Lemma 2, $M \vDash \bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta'} \varphi_i\theta$, and hence $M \vDash \bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta} \varphi_i\theta$. Since M is an arbitrary model of P, we derive $P \vDash \forall(\bigvee_{i=1}^{n} \bigvee_{\theta \in \Theta} \varphi_i\theta)$, which means that Θ is a correct answer of $P \cup Q$.

5 Reverse Fixpoint Semantics

The fixpoint semantics of definite logic programs was first introduced by van Emden and Kowalski [14] using the direct consequences operator T_P. This operator is monotonic, continuous, and has the least fixpoint $T_P \uparrow \omega = \bigcup_{n=0}^{\omega} T_P \uparrow n$, which forms the least Herbrand model of the given logic program P. In [9], Lobo et al extended the fixpoint semantics to disjunctive logic programs. Their direct consequences operator, denoted by T_P^I, iterates over model-states, which are sets of disjunctions of ground atoms. This operator is also monotonic, continuous, and has a least fixpoint which is a least model-state characterizing the given program P.

In this section, we study a reversed analogue of the "direct consequences" operator called the *direct derivation operator*. The results of this section will be used to prove *answer completeness* of the negative hyper-resolution semantics.

Let P be a logic program, Q a query, and Γ the set obtained from $P \cup Q$ by replacing every positive clause $(A_1 \vee \ldots \vee A_n \leftarrow)$ by $(A_1 \vee \ldots \vee A_n \leftarrow \top)$, where $\top$ is a special atom not occurring in P and Q.

The *direct derivation operator* D_Γ is a function that maps a set G of informative goals to another set of informative goals that can be directly derived from Γ and G. It is formally defined as follows: $D_\Gamma(G)$ is the set of all goals $\varphi : \Theta$ such that either φ is a clause of Q and $\Theta = \varepsilon$ or $\varphi : \Theta$ is a hyper-resolvent of a program clause ψ' and goals $\psi'_1 : \Theta'_1, \ldots, \psi'_n : \Theta'_n$, where $\{\psi', \psi'_1 : \Theta'_1, \ldots, \psi'_n : \Theta'_n\}$ is the standardized variant of $\{\psi, \psi_1 : \Theta_1, \ldots, \psi_n : \Theta_n\}$, ψ is a program clause of Γ, and $\psi_1 : \Theta_1, \ldots, \psi_n : \Theta_n$ are goals from G.

Lemma 3. *The operator D_Γ is monotonic, compact, and hence also continuous. It has the least fixpoint $D_\Gamma \uparrow \omega = \bigcup_{n=0}^{\omega} D_\Gamma \uparrow n$, where $D_\Gamma \uparrow 0 = \emptyset$ and $D_\Gamma \uparrow (n+1) = D_\Gamma(D_\Gamma \uparrow n)$.*

The first assertion of the above lemma clearly holds. The second assertion immediately follows from the first one, by the Kleene theorem.

Let G_Γ denote the set of all ground goals φ such that there exists an informative goal $\varphi' : \Theta' \in D_\Gamma \uparrow \omega$ such that φ is a ground instance of φ' (i.e. φ is obtained from φ' by uniformly substituting variables by terms from U_Γ).

A *negated representative* of G_Γ is a set Φ of pairs (φ, A) such that: $\varphi \in G_\Gamma$ and A is an atom of φ; and for every $\psi \in G_\Gamma$, there exists exactly one atom B of ψ (a negated representative of ψ) such that $(\psi, B) \in \Phi$.

Clearly, every G_Γ has at least one negated representative.

Let Φ be a negated representative of G_Γ. A set M of ground atoms is called a *minimal refinement* of Φ (w.r.t. G_Γ) if the following conditions hold:

1. for each $A \in M$ there exists $(\varphi, A) \in \Phi$ for some φ;
2. for each $\varphi \in G_\Gamma$ there exists $A \in M$ such that A is an atom of φ;
3. for each $A \in M$ there exists $\varphi \in G_\Gamma$ such that for every atom B of φ different from A, we have $B \notin M$.

Condition 1 states that members of M come from Φ. Condition 2 states that every Herbrand model disjoint with M satisfies G_Γ; in particular, $\overline{M} \models G_\Gamma$. Condition 3 states that M is a minimal set satisfying the two preceding conditions.

Lemma 4. *Every negated representative Φ of G_Γ has a minimal refinement.*

Proof. Start from $M = \{A \mid (\varphi, A) \in \Phi \text{ for some } \varphi\}$ and keeping in mind that M will always satisfy the first two conditions of the definition of minimal refinement, do the following: if M is not a minimal refinement of Φ due to some A that violates the last condition of the definition, then remove that A from M. This operator has a fixpoint which is a minimal refinement of Φ.

Theorem 2. *Let Φ be a negated representative of G_Γ and M a minimal refinement of Φ. Then $\overline{M}$ is a maximal Herbrand model of Γ.*

Proof. Since M is a minimal refinement of Φ, due to Condition 3 of its definition, it is sufficient to prove that $\overline{M}$ is a model of Γ. Let $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ be a ground instance of some clause φ' of Γ by a substitution σ. It suffices to show that $\overline{M} \models \varphi$. Suppose that $\overline{M} \not\models A_1 \vee \ldots \vee A_n$. We show that $\overline{M} \not\models B_1 \wedge \ldots \wedge B_m$.

Since each A_i is a ground atom and $\overline{M} \not\models A_1 \vee \ldots \vee A_n$, we must have $A_i \in M$ for all $1 \leq i \leq n$. Since M is a minimal refinement of Φ, it follows that for every $1 \leq i \leq n$ there exists $(\varphi_i, A_i) \in \Phi$ such that φ_i can be written as $\leftarrow A_i \wedge \zeta_i$ and ζ_i is false in M. Since Φ is a negated representative of G_Γ, for all $1 \leq i \leq n$, there exist a goal $\varphi'_i : \Theta'_i \in D_\Gamma \uparrow \omega$ and a substitution σ_i such that $\varphi_i = \varphi'_i \sigma_i$. For $1 \leq i \leq n$, let ξ'_i be the set of all atoms A''_i of φ'_i such that $A''_i \sigma_i = A_i$, and let ζ'_i be the set of the remaining atoms of φ'_i. We have $\varphi'_i = \leftarrow \xi'_i \wedge \zeta'_i$.

Let $\{\varphi'', \varphi''_1 : \Theta''_1, \ldots, \varphi''_n : \Theta''_n\}$ be the standardized variant of $\{\varphi', \varphi'_1 : \Theta'_1, \ldots, \varphi'_n : \Theta'_n\}$ with δ, δ_1, $\ldots$, δ_n being the involved renaming substitutions. For $1 \leq i \leq n$, let ξ''_i be the set of atoms of φ''_i originated from ξ'_i. Let $\psi' : \Theta'$ be a hyper-resolvent of the program clause φ'' and the goals $\varphi''_1 : \Theta''_1$, $\ldots$, $\varphi''_n : \Theta''_n$ with ξ''_i as the set of selected atoms of φ''_i. Thus $\psi' : \Theta' \in D_\Gamma \uparrow \omega$. We have $\varphi = \varphi'\sigma = \varphi''\delta^{-1}\sigma$ and $\varphi_i = \varphi'_i\sigma_i = \varphi''_i\delta_i^{-1}\sigma_i$, for all $1 \leq i \leq n$. Hence $\psi = \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \zeta_1 \wedge \ldots \wedge \zeta_n$ is a ground instance of ψ'.

Since $\psi' : \Theta' \in D_\Gamma \uparrow \omega$, we have $\psi \in G_\Gamma$. By Condition 2 of the definition of minimal refinement, we have $\overline{M} \models G_\Gamma$. It follows that $\overline{M} \models \psi$, which means that $\overline{M} \models \neg B_1 \vee \ldots \vee \neg B_m \vee \neg\zeta_1 \vee \ldots \vee \neg\zeta_n$. Since ζ_i is false in M for all $1 \leq i \leq n$, it follows that $\overline{M} \models \neg B_1 \vee \ldots \vee \neg B_m$, and hence $\overline{M} \not\models B_1 \wedge \ldots \wedge B_m$.

Corollary 1. *Every maximal model of G_Γ is a model of Γ.*

Sketch. Every maximal model of G_Γ is the compliment of a minimal refinement of some negated representative of G_Γ, and hence is a model of Γ.

6 Answer Completeness

In this section, we show that for every correct answer Θ of $P \cup Q$, where P is a logic program and Q is a query, there exists a computed answer of $P \cup Q$ which is more general than Θ.

Lemma 5 (Lifting Lemma). *Let P be a logic program, Q a query, and Θ a disjunctive substitution. Let $\varphi'_1 : \Theta'_1, \ldots, \varphi'_k : \Theta'_k$ be a derivation from $P \cup Q\Theta$. Then there exist a derivation $\varphi_1 : \Theta_1, \ldots, \varphi_k : \Theta_k$ from $P \cup Q$ and substitutions σ_i, for $1 \leq i \leq k$, such that $\varphi_i\sigma_i = \varphi'_i$ and $(\Theta_i\sigma_i)_{|\mathcal{X}} \subseteq (\Theta\Theta'_i)_{|\mathcal{X}}$.*

Proof. Simulate the derivation $\varphi'_1 : \Theta'_1, \ldots, \varphi'_k : \Theta'_k$ from $P \cup Q\Theta$ for $P \cup Q$ so that, for $\psi \in Q$ and $\theta \in \Theta$, the goal $\psi\theta \in Q\Theta$ is replaced by ψ. Let the resulting derivation be $\varphi_1 : \Theta_1, \ldots, \varphi_k : \Theta_k$.

We prove the assertion of this lemma by induction on i. The case when φ_i is a clause from Q and $\Theta_i = \varepsilon$ is trivial. Suppose that $\varphi_i : \Theta_i$ is derived as a hyper-resolvent of a standardized variant of a program clause $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ of P and standardized variants of goals $\varphi_{j_1} : \Theta_{j_1}, \ldots, \varphi_{j_n} : \Theta_{j_n}$, where $j_1, \ldots, j_n$ belong to $1..(i-1)$. Let $\delta, \delta_1, \ldots, \delta_n$ be the involved renaming substitutions (for standardizing variants) and σ be the involved mgu. Let $\varphi_{j_t} = \leftarrow \xi_{j_t} \wedge \zeta_{j_t}$ with ξ_{j_t} as the set of selected atoms, for $1 \leq t \leq n$. We have $A_t\delta\sigma = A'_t\delta_t\sigma$ for every $1 \leq t \leq n$ and every atom A'_t of ξ_{j_t}. The hyper-resolvent $\varphi_i : \Theta_i$ is equal to

$$\leftarrow (B_1\delta \wedge \ldots \wedge B_m\delta \wedge \zeta_{j_1}\delta_1 \wedge \ldots \wedge \zeta_{j_n}\delta_n)\sigma : (\Theta_{j_1}\delta_1 \cup \ldots \cup \Theta_{j_n}\delta_n)\sigma$$

By the inductive assumption, $\varphi_{j_t}\sigma_{j_t} = \varphi'_{j_t}$, for all $1 \leq t \leq n$. Hence $\varphi'_i : \Theta'_i$ is a hyper-resolvent of a standardized variant of φ and standardized variants of $\varphi_{j_1}\sigma_{j_1} : \Theta'_{j_1}, \ldots, \varphi_{j_n}\sigma_{j_n} : \Theta'_{j_n}$. Let $\delta', \delta'_1, \ldots, \delta'_n$ be the involved renaming substitutions (for standardizing variants) and σ' be the involved mgu. We have $A_t\delta'\sigma' = A'_t\sigma_{j_t}\delta'_t\sigma'$ for every $1 \leq t \leq n$ and every atom A'_t of ξ_{j_t}. The hyper-resolvent $\varphi'_i : \Theta'_i$ is equal to

$$\leftarrow (B_1\delta' \wedge \ldots \wedge B_m\delta' \wedge \zeta_{j_1}\sigma_{j_1}\delta'_1 \wedge \ldots \wedge \zeta_{j_n}\sigma_{j_n}\delta'_n)\sigma' : (\Theta'_{j_1}\delta'_1 \cup \ldots \cup \Theta'_{j_n}\delta'_n)\sigma'$$

Let γ be the normal substitution specified as below

$$\gamma = (\delta^{-1}\delta')_{|Dom(\delta^{-1})} \cup (\delta_1^{-1}\sigma_{j_1}\delta'_1)_{|Dom(\delta_1^{-1})} \cup \ldots \cup (\delta_n^{-1}\sigma_{j_n}\delta'_n)_{|Dom(\delta_n^{-1})}$$

Let $1 \leq t \leq n$ and let A'_t be an atom of ξ_{j_t}. We have $A_t\delta\gamma = A_t\delta'$ and $A'_t\delta_t\gamma = A'_t\sigma_{j_t}\delta'_t$. Since $A_t\delta'\sigma' = A'_t\sigma_{j_t}\delta'_t\sigma'$, it follows that $A_t\delta\gamma\sigma' = A'_t\delta_t\gamma\sigma'$. Because σ is an *mgu* such that $A_t\delta\sigma = A'_t\delta_t\sigma$ for every $1 \leq t \leq n$ and every atom A'_t of ξ_{j_t}, there exists σ_i such that $\gamma\sigma' = \sigma\sigma_i$.

For $1 \leq s \leq m$, we have $B_s\delta\sigma\sigma_i = B_s\delta\gamma\sigma' = B_s\delta'\sigma'$, and for $1 \leq t \leq n$, we have $\zeta_{j_t}\delta_t\sigma\sigma_i = \zeta_{j_t}\delta_t\gamma\sigma' = \zeta_{j_t}\sigma_{j_t}\delta'_t\sigma'$. Hence $\varphi_i\sigma_i = \varphi'_i$.

For all $1 \leq t \leq n$, we have $(\Theta_{j_t}\delta_t\sigma\sigma_i)_{|\mathcal{X}} = (\Theta_{j_t}\delta_t\gamma\sigma')_{|\mathcal{X}} = ((\Theta_{j_t}\delta_t\gamma)_{|\mathcal{X}}\,\sigma')_{|\mathcal{X}} = ((\Theta_{j_t}\sigma_{j_t}\delta'_t)_{|\mathcal{X}}\,\sigma')_{|\mathcal{X}} = (\Theta_{j_t}\sigma_{j_t}\delta'_t\sigma')_{|\mathcal{X}}$. By the inductive assumption, $(\Theta_{j_t}\sigma_{j_t})_{|\mathcal{X}} \subseteq (\Theta\Theta'_{j_t})_{|\mathcal{X}}$, and hence $(\Theta_{j_t}\sigma_{j_t}\delta'_t\sigma')_{|\mathcal{X}} \subseteq (\Theta\Theta'_{j_t}\delta'_t\sigma')_{|\mathcal{X}}$. We also have $\Theta'_{j_t}\delta'_t\sigma' \subseteq \Theta'_i$, which implies that $(\Theta\Theta'_{j_t}\delta'_t\sigma')_{|\mathcal{X}} \subseteq (\Theta\Theta'_i)_{|\mathcal{X}}$. Hence $(\Theta_{j_t}\delta_t\sigma\sigma_i)_{|\mathcal{X}} \subseteq (\Theta\Theta'_i)_{|\mathcal{X}}$. Therefore $(\Theta_i\sigma_i)_{|\mathcal{X}} \subseteq (\Theta\Theta'_i)_{|\mathcal{X}}$, which completes the proof.

Theorem 3 (Completeness). *Let P be a logic program, Q a query, and Θ a correct answer of $P \cup Q$. Then there exists a computed answer Θ' of $P \cup Q$ which is more general than Θ.*

Proof. Let $Q = \{\varphi_1, \ldots, \varphi_n\}$ and $Y = Var(Q) \cup Ran(\Theta)$. For each $x \in Y$, let a_x be a fresh constant symbol. Let $\delta = \{x/a_x \mid x \in Y\}$ and $Q' = Q\Theta\delta$. Since Θ is a correct answer of $P \cup Q$, it follows that $P \cup Q'$ is unsatisfiable.

Let P' be the set obtained from P by replacing every positive clause $(A_1 \vee \ldots \vee A_n \leftarrow)$ by $(A_1 \vee \ldots \vee A_n \leftarrow \top)$, and let $\Gamma = P' \cup Q'$. Since $P \cup Q'$ is unsatisfiable, we have $\Gamma \models \neg\top$.

We first show that $(\leftarrow \top) \in G_\Gamma$. Suppose oppositely that for every $\varphi \in G_\Gamma$, $\varphi \neq (\leftarrow \top)$. Then there exists a negated representative Φ of G_Γ which does not contain $\top$. Let M be a minimal refinement of Φ. We have that $\overline{M}$ contains $\top$. By Theorem 2, $\overline{M} \models \Gamma$, which contradicts with $\Gamma \models \neg\top$.

The above assertion states that there exists a derivation from Γ with the last goal of the form $\leftarrow \top : \Delta$. By simulating that derivation for $P \cup Q'$ with each $(A_1 \vee \ldots \vee A_n \leftarrow \top)$ replaced by $(A_1 \vee \ldots \vee A_n \leftarrow)$, we obtain a refutation with $\bot : \Delta$ as the last goal.

Since $Q' = Q\Theta\delta$, by Lemma 5, there exists a refutation of $P \cup Q$ with the last goal of the form $\bot : \Theta''$ and a substitution σ'' such that $(\Theta''\sigma'')_{|\mathcal{X}} \subseteq (\Theta\delta\Delta)_{|\mathcal{X}}$. We have that $\Theta' = \Theta''_{|Var(Q)}$ is a computed answer of $P \cup Q$. Since $\mathcal{X}' \cap \mathcal{X} = \emptyset$, we have $Var(\Delta) \cap Var(Q) = \emptyset$ and $Var(\Delta) \cap Var(\Theta\delta) = \emptyset$. Since $(\Theta''\sigma'')_{|\mathcal{X}} \subseteq (\Theta\delta\Delta)_{|\mathcal{X}}$, it follows that $(\Theta''\sigma'')_{|Var(Q)} \subseteq (\Theta\delta)_{|Var(Q)}$. Now treat each a_x as a variable and δ as a renaming substitution. Then we have $(\Theta''\sigma''(\delta^{-1}))_{|Var(Q)} \subseteq (\Theta\delta(\delta^{-1}))_{|Var(Q)}$. Since each a_x occurs neither in Θ nor in Θ'', for $\sigma' = (\sigma''\delta^{-1})_{|Dom(\sigma'')}$, we can derive that $(\Theta''\sigma')_{|Var(Q)} \subseteq \Theta$. Hence $(\Theta'\sigma')_{|Var(Q)} \subseteq \Theta$ and Θ' is more general than Θ.

7 Keeping Information for Computed Answers

In this section, we first modify the definition of derivation so that disjunctive substitutions in informative goals keep only necessary information without violating soundness and completeness of the calculus. We then show that informative goals can be simulated by normal goals using answer literals. We also study cases when it is possible to make computed answers more compact.

Let P be a logic program, Q a query, and $X \subseteq Var(Q)$. A *derivation restricted to* X from $P \cup Q$ is a modification of a derivation from $P \cup Q$ in which each newly derived hyper-resolvent $\varphi : \Theta$ is replaced *immediately* by $\varphi : \Theta_{|X}$. (Note that such a replacement affects the remaining part of the derivation.) A *refutation restricted to* X of $P \cup Q$ is a derivation restricted to X from $P \cup Q$ with the last goal of the form $\bot : \Theta$.

Example 2. Reconsider Example 1. Here is a refutation restricted to $\{x, y\}$ of $P \cup Q$:

(6) $\leftarrow s(x, y) : \varepsilon$ from (5)

(7) $\leftarrow p(x_2) : \{x/x_2, y/a\}$ (1),(6)
(8) $\leftarrow q(x_4) : \{x/x_4, y/b\}$ (2),(6)
(9) $\leftarrow r(x_5) : \{\{x/x_5, y/a\}, \{x/x_5, y/b\}\}$ (3),(7),(8)
(10) $\bot : \{\{x/c, y/a\}, \{x/c, y/b\}\}$ (4),(9)

Lemma 6. *Let P be a logic program, Q a query, and $X \subseteq Var(Q)$. Let $\varphi_1 : \Theta_1$, ..., $\varphi_n : \Theta_n$ be a derivation from $P \cup Q$ and $\varphi_1 : \Theta'_1$, ..., $\varphi_n : \Theta'_n$ be its version restricted to X. Then $\Theta'_i = \Theta_{i|X}$ for all $1 \leq i \leq n$.*

This lemma can be proved by induction on i in a straightforward way.

The following theorem states that we can save memory when searching for refutations by restricting kept disjunctive substitutions to the set of interested variables. The theorem immediately follows from the above lemma.

Theorem 4. *Let P be a logic program, Q a query, and $X \subseteq Var(Q)$. If $\bot : \Theta'$ is the last goal of a refutation restricted to X of $P \cup Q$, then there exists a computed answer Θ of $P \cup Q$ such that $\Theta' = \Theta_{|X}$. Conversely, for every computed answer Θ of $P \cup Q$, there exists a refutation restricted to X of $P \cup Q$ with the last goal $\bot : \Theta'$ such that $\Theta' = \Theta_{|X}$ (in particular, $\Theta' = \Theta$ when $X = Var(Q)$).*

We can simulate disjunctive substitutions by answer literals as follows.

For each variable x, let "x" be a constant symbol for keeping the name of x. We use "x"$/t$, where $/$ is an infix function symbol, to keep the binding x/t. Let *ans* be a special predicate symbol which can have different arities. Atoms of this predicate symbol will be always denoted either explicitly as $ans(\ldots)$ or using a prefix *Ans*. A literal $ans(\text{"}x_1\text{"}/t_1, \ldots, \text{"}x_n\text{"}/t_n)$ is called an *answer literal* if $x_1, \ldots, x_n$ are different variables. This answer literal can be treated as $\{x_1/t_1, \ldots, x_n/t_n\}$. By deleting from this set pairs x_i/t_i with $t_i = x_i$ we obtain a normal substitution, which is called the *substitution corresponding to the answer literal* $ans(\text{"}x_1\text{"}/t_1, \ldots, \text{"}x_n\text{"}/t_n)$. If $\varphi = Ans_1 \vee \ldots \vee Ans_m$ and θ_i is the substitution corresponding to Ans_i, for $1 \leq i \leq m$, then we call $\{\theta_1, \ldots, \theta_m\}$ the *disjunctive substitution corresponding to* φ. Assume that ε is the substitution corresponding to the empty clause.

A *goal with answer literals* is a clause of the following form, with $n, m \geq 0$:

$$Ans_1 \vee \ldots \vee Ans_n \leftarrow B_1 \wedge \ldots \wedge B_m$$

Let $\varphi = A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ be a program clause ($n > 0$), and $(\psi_1 \leftarrow \varphi_1)$, ..., $(\psi_n \leftarrow \varphi_n)$ be goals with answer literals (i.e. each ψ_i is a disjunction of answer literals). Let $\varphi_i = (\xi_i \wedge \zeta_i)$ for $1 \leq i \leq n$, where ξ_i is a non-empty set of atoms *selected* for φ_i. If there exists an mgu σ such that $A_i\sigma = A'_i\sigma$ for every $1 \leq i \leq n$ and every atom A'_i of ξ_i, then we call the goal

$$(\psi_1 \vee \ldots \vee \psi_n \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \zeta_1 \wedge \ldots \wedge \zeta_n)\sigma$$

a *hyper-resolvent (with answer literals)* of φ and $(\psi_1 \leftarrow \varphi_1)$, ..., $(\psi_n \leftarrow \varphi_n)$. Note that such a hyper-resolvent is also a goal with answer literals.

Let P be a logic program, $Q = \{\varphi_1, \ldots, \varphi_n\}$ a query, and $X \subseteq Var(Q)$. For each $1 \leq i \leq n$, let $Var(\varphi_i) \cap X = \{x_{i,1}, \ldots, x_{i,k_i}\}$, $\varphi_i = \leftarrow \xi_i$, and $\varphi'_i = ans(\text{“}x_{i,1}\text{”}/x_{i,1}, \ldots, \text{“}x_{i,k_i}\text{”}/x_{i,k_i}) \leftarrow \xi_i$ if $k_i > 0$, or $\varphi'_i = \varphi_i$ if $k_i = 0$. Let $Q' = \{\varphi'_1, \ldots, \varphi'_n\}$. A *derivation from $P \cup Q$ with answer literals for X* is a sequence $\psi_1, \ldots, \psi_m$ of goals with answer literals such that for each $1 \leq j \leq m$, either $\psi_j \in Q'$ or ψ_j is a hyper-resolvent with answer literals of a standardized variant of a program clause of P and standardized variants of some goals from $\psi_1, \ldots, \psi_{j-1}$, where a *standardized variant* is a renaming of all the variables in the original clause so that it does not contain variables of the other involved variants. Such a derivation is called a *refutation of $P \cup Q$ with answer literals for X* if the last goal ψ_m is either the empty clause or a positive clause (consisting of only answer literals).

Example 3. Reconsider Example 1. Here is a refutation of $P \cup Q$ with answer literals for $\{x, y\}$:

(6)	$ans(\text{“}x\text{”}/x, \text{“}y\text{”}/y) \leftarrow s(x, y)$	from (5)
(7)	$ans(\text{“}x\text{”}/x_2, \text{“}y\text{”}/a) \leftarrow p(x_2)$	(1),(6)
(8)	$ans(\text{“}x\text{”}/x_4, \text{“}y\text{”}/b) \leftarrow q(x_4)$	(2),(6)
(9)	$ans(\text{“}x\text{”}/x_5, \text{“}y\text{”}/a) \vee ans(\text{“}x\text{”}/x_5, \text{“}y\text{”}/b) \leftarrow r(x_5)$	(3),(7),(8)
(10)	$ans(\text{“}x\text{”}/c, \text{“}y\text{”}/a) \vee ans(\text{“}x\text{”}/c, \text{“}y\text{”}/b)$	(4),(9)

Theorem 5. *Let P be a logic program, Q a query, and $X \subseteq Var(Q)$. If ψ is the last goal of a refutation of $P \cup Q$ with answer literals for X, then there exists a computed answer Θ of $P \cup Q$ such that $\Theta_{|X}$ is the disjunctive substitution corresponding to ψ. Conversely, for every computed answer Θ of $P \cup Q$, there exists ψ as the last goal of a refutation of $P \cup Q$ with answer literals for X such that $\Theta_{|X}$ is the disjunctive substitution corresponding to ψ.*

Proof. Given a refutation of $P \cup Q$ with answer literals for X, simulate it by a refutation restricted to X of $P \cup Q$. For the converse direction, do it analogously. Let $\zeta_i \leftarrow \psi_i$ and $\leftarrow \psi_i : \Theta_i$ be the goals number i in the two corresponding refutations. By induction on i, it is easy to see that Θ_i is the disjunctive substitution corresponding to ζ_i. This together with Theorem 4 proves this theorem.

Keeping information for computed answers by using answer literals is just one of possible techniques, which is not always optimal. For example, $ans(\text{“}x\text{”}/a, \text{“}y\text{”}/y) \vee ans(\text{“}x\text{”}/a, \text{“}y\text{”}/b) \vee ans(\text{“}x\text{”}/a, \text{“}y\text{”}/c)$ can be better represented as the composition of $\{x/a\}$ and $\{\varepsilon, \{y/b\}, \{y/c\}\}$.

We say that $\Theta = \{\theta_1, \ldots, \theta_n\}$ has a *conflict w.r.t. x* if there exist bindings $x/t_1 \in \theta_i$ and $x/t_2 \in \theta_j$ for some i, j from $1..n$ such that $t_1 \neq t_2$. Suppose that Θ is a computed answer of $P \cup Q$ and Θ has no conflicts w.r.t. any variable. Then the normal substitution $\theta = \bigcup \Theta$ is also a correct answer of $P \cup Q$. Despite that θ is “tighter” than Θ, from the point of view of users, θ is more intuitive and sufficient enough.

Consider a more general case. Suppose that $\Theta = \{\theta_1, \ldots, \theta_n\}$ is a computed answer of $P \cup Q$, $x \in Dom(\Theta)$, and Θ has no conflicts w.r.t. x. Let x/t be

the binding of x that belongs to some θ_i, $1 \leq i \leq n$. Let $\theta'_j = \theta_j - \{x/t\}$ for $1 \leq j \leq n$. Then $\{x/t\}\{\theta'_1, \ldots, \theta'_n\}$ is also a correct answer of $P \cup Q$. This kind of extraction can be applied further for $\{\theta'_1, \ldots, \theta'_n\}$, and so on. The resulting composition is a correct answer "tighter" than Θ but it is more compact and still acceptable from the point of view of users.

8 Conclusions

We have proved that negative hyper-resolution is a sound and complete procedure for answering queries in disjunctive logic programs. This is a fundamental theoretical result for the intersection of theorem proving, disjunctive logic programming and AI. Our completeness proof is short and based on our reverse fixpoint semantics of disjunctive logic programs.

We have also introduced disjunctive substitutions to represent answers of queries. Our definition can be looked at as a formulation on the semantic level, while answer literals used in theorem proving systems are defined on the syntactical level. Our formulation extracts the meaning of answers from representation and in some situations allows a better encoding.

As a future work, we will study answer completeness of negative hyper-resolution under ordering refinements.

Acknowledgements: The author would like to thank Dr. Rajeev Goré and the anonymous reviewers for helpful comments.

References

1. K.R. Apt and M.H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3):841–862, 1982.
2. P. Baumgartner and U. Furbach. Calculi for disjunctive logic programming. In Jan Maluszynski, editor, *Proc. of ILPS 1997*, pages 229–243. The MIT Press, 1997.
3. S. Brass and U.W. Lipeck. Generalized bottom-up query evaluation. In G. Gottlob A. Pirotte, C. Delobel, editor, *Advances in Database Technology — EDBT'92, 3rd Int. Conf.*, volume LNCS 580, pages 88–103. Springer-Verlag, 1992.
4. C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.
5. C.C. Green. Theorem proving by resolution as basis for question-answering systems. *Machine Intelligence*, 4:183–205, 1969.
6. R.A. Kowalski. Predicate logic as a programming language. *Information Processing Letters*, 74:569–574, 1974.
7. K. Kunen. The semantics of answer literals. *Journal of Automated Reasoning*, 17(1):83–95, 1996.
8. J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.
9. J. Lobo, A. Rajasekar, and J. Minker. Semantics of Horn and disjunctive logic programs. *Theoretical Computer Science*, 86(1):93–106, 1991.
10. D. Loveland. Near-Horn Prolog. In J.-L. Lassez, editor, *Proc. of the 4th Int. Conf. on Logic Programming*, pages 456–469. The MIT Press, 1987.

11. D. Luckham and N.J. Nilsson. Extracting information from resolution proof trees. *Artificial Intelligence*, 2:27–54, 1971.
12. J.A. Robinson. Automatic deduction with hyper-resolution. *International Journal of Computer Mathematics*, 1:227–234, 1965.
13. J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.
14. M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.