CONSTRUCTIVE NEGATION UNDER THE WELL-FOUNDED SEMANTICS *

JULIE YUCHIH LIU† , LEROY ADAMS‡ AND WEIDONG CHEN

▷

Constructive negation derives constraint answers for non-ground negative literals. Its incorporation into query evaluation under the well-founded semantics introduces two problems. One is the detection of repeated subgoals and the elimination of redundant answers, which is required in order to guarantee termination. The other is the interaction between constraint answers of non-ground negative literals and recursion through negation. This paper presents SLG_{CN} for effective query evaluation with constructive negation under the well-founded semantics. It has two unique features. First, it supports reduction of constraint answers and redundant answer elimination and provides the first termination result for goal-oriented query evaluation with constructive negation for function-free programs. Second, it avoids repeated computation in a subgoal. Even if a non-ground negative literal depends upon some ground negative literals whose truth values are not completely determined when they are selected, the constraints and bindings for variables in the non-ground negative literal can still be propagated once and for all.

◁

*Supported in part by the National Science Foundation under Grant No. IRI-9314897.

†Department of Information Management, Yuan-Ze University, Taiwan. Work was done while at Southern Methodist University.

‡Department of Computer Science, College of the Ozarks, Point Lookout, MO 65726. Work was done while at Southern Methodist University.

Address correspondence to Weidong Chen, Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas 75275-0122.

THE JOURNAL OF LOGIC PROGRAMMING

655 Avenue of the Americas, New York, NY 10010 0743-1066/93/$3.50

1. INTRODUCTION

Negation as failure [8] is the dominant mechanism for processing negative literals in logic programming. Procedurally speaking, a negative literal succeeds if the corresponding positive literal fails, and it fails if the corresponding positive literal succeeds. No variable binding or constraint is generated from a negative literal. Not surprisingly negation as failure is sound for only ground negative literals. Consider a simple definition of a bachelor:

$$bachelor(X) \text{ :- } \sim married(X), man(X).$$

with respect to the following database:

$married(john)$.	$man(john)$.
$married(mary)$.	$man(jack)$.

In most Prolog systems, a query such as $bachelor(X)$ fails even though $bachelor(jack)$ succeeds. (We use $\sim$ to emphasize the non-monotonic nature of negation in logic programming.)

Most procedural semantics of logic programs guarantees completeness for only "non-floundered" queries, whose evaluation does not involve the selection of a non-ground negative literal [1, 7, 20, 26]. Similar restrictions are placed by bottom up methods of query evaluation such that negation can be implemented by set difference [19, 28].

Incompleteness or abrupt termination of query evaluation due to non-ground negative literals is not satisfactory from users' point of view. In addition, it is useful to treat negative literals as generators of constraints, especially from the constraint logic programming perspective.

In terms of the mechanisms to solve a non-ground negative literal, several distinct techniques have been developed, with varying degrees to which the corresponding positive literal is evaluated. In [5, 29, 32], a negative literal is solved using Clark's completed definitions at run time, possibly with partial evaluation. Quantified complex formulas have to be transformed into a disjunctive normal form and be dealt with explicitly. In [11, 12, 17], substitutions called *fail answers* are generated for variables in a negative literal $\sim A$ based upon a *frontier* of the positive literal A. This is a powerful technique since A does not have to be completely evaluated before an answer for $\sim A$ is derived. Since a subgoal can have many different frontiers, there is an implementation problem how to control the derivation tree of a subgoal and the choice of frontiers. In [2, 4, 10, 15, 21], constraint answers of a negative literal are derived by taking the negation of the disjunction of all the answers of its positive counterpart.

In terms of the semantics, most of the previous work on constructive negation, with notable exceptions of [10, 11, 21], uses Clark's completion as the corresponding declarative semantics. It is known, however, that Clark's completion has various drawbacks [24]. The well-founded semantics [30] has been accepted as a more natural and robust semantics for logic programs.

Przymusinski first studied constructive negation under the perfect model semantics and developed SLSC-resolution for constructive negation of stratified programs [21]. In [11], Drabent described SLSFA-resolution for constructive negation under the well-founded semantics. However, both SLSC-resolution [21] and SLSFA-resolution [11] require infinite failure and therefore are not suitable for effective query evaluation.

This paper focuses on termination and efficient query evaluation under the well-founded semantics with constructive negation — issues that have received little attention in the literature. The work that is more closely related to ours is by Warren [33] and by Damasio [10].

Warren [33] developed a Prolog meta interpreter for constructive negation that was executed using an OLDT implementation. Constraint answers of negative literals are represented using anti-subsumption constraints. In fact the use of anti-subsumption constraints in this paper is motivated by [33]. However, the implementation in [33] does not handle constraints of the form $\forall \overline{U}.\exists \overline{V}.E$ properly and thus is not sound in general. Also recursion through negation is not supported.

The work in [10] is a systematic study of constructive negation in tabled query evaluation under the well-founded semantics. It extends tabulated resolution for the well-founded semantics in [1] with constructive negation. For non-ground negative literals involved in recursion through negation, approximate constraint answers are derived. Due to iterated approximations, constructive negation may be repeatedly applied to the same non-ground negative literal using slightly different answers, causing repeated computation inside a subgoal. Although theoretical results of soundness and search space completeness are established, independently of the constraint domain used, for constraint logic programs with function symbols, pragmatic control issues such as redundant answer elimination, termination and repeated computation are not incorporated into the formalization. In particular, no termination result is given for tabled query evaluation with constructive negation.

We extend SLG resolution [7] with constructive negation for effective query evaluation under the well-founded semantics. The resulting SLG_{CN} resolution has two major contributions, distinguishing itself from [10] and other previous work on constructive negation.

First, we have developed a normal form for constraint answers and a simple algorithm for redundant answer elimination. This allows us to establish the first termination result for constructive negation of function-free logic programs.

Second, like SLG resolution [7], SLG_{CN} resolution is formalized directly in such a way that repeated computation is avoided in a subgoal. The key idea is to take advantage of the difference between ground and non-ground negative literals and to delay only ground negative literals when dealing with recursion through negation, Consequently even if a non-ground negative literal depends upon some ground negative literals whose truth values are possibly undefined, the constraints and bindings of variables in the non-ground negative literal can still be propagated once and for all. In other words, each selected non-ground negative literal is solved using constructive negation at most once. This is an important property of practical significance since constructive negation is a complex operation and repeated applications of constructive negation to the same negative literal can cause performance overhead.

The rest of this paper is organized as follows. Section 2 contains terminology and definitions used throughout the paper. Section 3 describes a normal form of constraint answers, detection and elimination of redundant constraint answers and the derivation of constraint answers through constructive negation. Section 4 illustrates through examples tabled evaluation with constructive negation and how to avoid repeated computation even if there is recursion through negation. Section 5 presents the formal details of SLG_{CN} *resolution*. Section 6 establishes correctness and termination of SLG_{CN} resolution. Finally we conclude with a discussion of

some issues for future work.

2. PRELIMINARIES

This section defines the terminology and notations used in this paper, including anti-subsumption constraints, systems in SLG resolution [7], and three-valued stable models.

2.1. Semantics of Equality and Anti-Subsumption Constraints

We assume a countable language $\mathcal{LF}$ of function symbols. $\mathcal{LF}$ contains all function symbols that occur in programs involved in the evaluation of a query, plus a unary function symbol f' and a zero-ary function symbol c' that do not occur in any of the programs or query being considered. The symbols f' and c' are needed for two purposes. One is to cope with the "universal query problem" [22], where the semantics of a program containing a single fact, $p(a)$, may imply $\forall X.p(X)$ if the Herbrand universe is $\{a\}$. But the empty answer substitution cannot be obtained for $p(X)$ by SLD resolution. The introduction of new symbols f' and c' eliminates such situations. The other is to facilitate the reduction of constraint answers.

The Herbrand universe $\mathcal{HU}$ is the set of all ground terms that can be constructed using function symbols in $\mathcal{LF}$. A *substitution* θ is of the form $\{t_1/X_1, ..., t_n/X_n\}$, where all X_i's are distinct variables and each t_i is a term different from X_i. We say that θ is *ground* if every $t_i (1 \leq i \leq n)$ is ground, i.e., in $\mathcal{HU}$. The *domain* of θ is the set $\{X_1, ..., X_n\}$ of variables. We denote sequences of variables by $\overline{X}, \overline{Y}$, etc. and sequences of terms by $\overline{s}, \overline{t}$, etc.

An *atom* is of the form $p(t_1, ..., t_n)$, where p is an n-ary predicate symbol and $t_1, ..., t_n$ are terms. If A is an atom, then A is a *positive literal* and $\sim A$ is a *negative literal*. A *literal* is either a positive or negative literal.

Constructive negation can produce constraint answers possibly containing disequations and universal quantifiers. We choose anti-subsumption constraints [14] and, with a slight abuse of notation, consider (dis)equations over atoms instead of terms. The anti-subsumption constraints provide a compact representation of counter-examples [16] or exceptions [3].

Definition 2.1. Let A and B be atoms, and ν be a ground substitution. Then $A = B$ is *true* under ν if $A\nu$ and $B\nu$ are identical ground atoms.

The standard definitions of truth, validity, and (un)satisfiability can be extended to equations over atoms.

Definition 2.2. [Anti-Subsumption Constraint] Let A and B be atoms with no variables in common. An *anti-subsumption constraint* (or simply *AS-constraint*) is of the form $\forall \overline{X}.A \neq B$, where $\overline{X}$ are all the variables occurring in B. The variables in A are called *free* variables, and those in B are called *bound* variables. We write $A \not\in B$ as an abbreviation of $\forall \overline{X}.A \neq B$.

Intuitively, an AS-constraint of the form $A \not\in B$ means that A cannot be an instance of B (if B is viewed as the set of its ground instances). As shown in [16],

constraints in general cannot be converted into a finite disjunction of equations (or substitutions).

Definition 2.3. A *constrained atom*, $\mathcal{A}$, is a pair of the form (A, ϕ), where A is an atom and ϕ is a conjunction of AS-constraints and ϕ is satisfiable.

Let $\overline{X}$ be all the free variables in ϕ excluding those in A. The *semantics of* $\mathcal{A}$, denoted by $|\mathcal{A}|$, is defined as the set of ground atoms $A\nu$, where ν is a ground substitution such that $(\exists \overline{X}.\phi)\nu$ is valid.

For convenience, a constrained atom (A, ϕ) is also written as A :- ϕ. If ϕ is empty, (A, ϕ) is simply viewed as an atom and written as A. For constrained atoms, subsumption can occur on the atom or on the constraint part.

Definition 2.4. Let $\mathcal{A}_i (i = 1, 2)$ be constrained atoms (A_i, ϕ_i) with no variables in common. $\mathcal{A}_1$ is *subsumed* (respectively, *AS-subsumed*) by $\mathcal{A}_2$ if and only if for some substitution θ whose domain is a subset of variables in A_2,

- $A_1 = A_2\theta$ (respectively, $A_1 = A_2\theta$ and θ is a renaming substitution), and
- every AS-constraint in $\phi_2\theta$ is a variant of some AS-constraint in ϕ_1 up to renaming of variables not occurring in A_1.

$\mathcal{A}_1$ and $\mathcal{A}_2$ are *variants* if and only if $\mathcal{A}_1$ is AS-subsumed by $\mathcal{A}_2$ and vice versa.

2.2. Adding AS-Constraints to Systems in SLG Resolution

SLG resolution [7] supports tabled query evaluation under the well-founded semantics with negation as failure. It maintains a global table of subgoals and their partially evaluated rules. In extending SLG resolution with constructive negation, we generalize some notions in SLG resolution with AS-constraints.

A *rule* is of the form

$$\mathcal{H} \text{ :- } L_1, ..., L_n$$

where $\mathcal{H}$ is a constrained atom (H, ϕ) and $L_i (1 \leq i \leq n, n \geq 0)$ is a literal. If ϕ is empty, the head $\mathcal{H}$ is simply written as H. A *ground instance* of the rule is

$$(H \text{ :- } L_1, ..., L_n)\theta$$

where θ is a ground substitution whose domain includes all free variables in the rule such that $\phi\theta$ is true.

A *logic program* (or simply *program*) P is a set of rules. The *Herbrand instantiation* of P is the set of all the ground instances of rules in P. The *Herbrand base of* P, denoted by $\mathcal{HB}_P$, is the set of all ground atoms that are constructed using predicates in P and terms in $\mathcal{HU}$.

Definition 2.5. A *subgoal* is a constrained atom. Two subgoals are considered identical if they are variants of each other. A *delayed literal* has one of the following forms:

- $\sim B^B$, where B is a ground atom;

- $B^{\mathcal{A}}_{\mathcal{H}}$ or $\sim B^{\mathcal{A}}_{\mathcal{H}}$, where B is an atom, and $\mathcal{A}$ and $\mathcal{H}$ are constrained atoms such that $\mathcal{H}$ is subsumed by $\mathcal{A}$ and if $\mathcal{H}$ is of the form (H, ϕ), then B is an instance of H.

If θ is a variable substitution, then $(B^{\mathcal{A}}_{\mathcal{H}})\theta$ is the delayed literal $(B\theta)^{\mathcal{A}}_{\mathcal{H}}$.

Delayed literals are used in SLG resolution to deal with recursion through negation when the truth value of a negative literal cannot be determined when it is selected. The superscript and subscript in a delayed literal provide control information for simplifying the delayed literal when its truth value is known later.

Definition 2.6. An *X-rule* G is of the form:

$$\mathcal{H} \text{ :- } L_1, ..., L_n$$

where $n \geq 0$, $\mathcal{H}$ is a constrained atom and each $L_i (1 \leq i \leq n)$ is an atom, the negation of an atom, or a delayed literal. If $n = 0$, G is called a *fact*. If every $L_i (1 \leq i \leq n, n \geq 0)$ is a delayed literal, G is called an *answer*.

A *computation rule* is an algorithm that selects from the body of an X-rule G a literal L that is not a delayed literal (if there is any).

Given a computation rule R, an *annotated X-rule* is either

- an X-rule that does not have a selected atom, or
- a pair of the form $\langle G, \Sigma(G) \rangle$, where G is an X-rule that has a selected atom and $\Sigma(G)$ is a set of constrained atoms.

In tabled query evaluation, a set of answers is maintained for each subgoal and each selected atom is solved using answers of the corresponding subgoal. The annotation of an X-rule G with a selected atom is used to indicate what answers have been returned to the selected atom.

Definition 2.7. Let P be a program, and R be a computation rule. A *system* $\mathcal{S}$ is a set of pairs of the form $(\mathcal{A} : \Gamma)$, where $\mathcal{A}$ is a subgoal and Γ is a multiset of annotated X-rules, such that no two pairs in $\mathcal{S}$ have the same subgoal. If $(\mathcal{A} : \Gamma) \in \mathcal{S}$, $\mathcal{A}$ is said to be a *subgoal in* $\mathcal{S}$, and each element in Γ is an *annotated X-rule of subgoal* $\mathcal{A}$ *in* $\mathcal{S}$, and if the element is an X-rule G or of the form $\langle G, \Sigma(G) \rangle$, then G is called an *X-rule of subgoal* $\mathcal{A}$ *in* $\mathcal{S}$. If G is an answer, then G is called an *answer of subgoal* $\mathcal{A}$.

SLG resolution [7] is essentially a process of transforming an initially empty system into a system that contains only subgoals that are encountered during the evaluation of a query and their answers. The correctness of SLG resolution is established based upon three-valued stable models [23] by relating annotated X-rules of subgoals in a system $\mathcal{S}$ to a program P.

Annotated X-rules of subgoals in a system $\mathcal{S}$ can be viewed as partial answers of subgoals. The correctness of SLG resolution is established by associating a program with $\mathcal{S}$, denoted by $P(\mathcal{S})$, and studying the relationship between three valued stable models of P and $P \cup P(\mathcal{S})$.

The program $P(\mathcal{S})$ is defined as follows. A new predicate is introduced for every subgoal $\mathcal{A}$ in $\mathcal{S}$ and every constrained atom $\mathcal{H} = (H, \phi)$ that is subsumed by $\mathcal{A}$. Atoms of the new predicate will be written as $B^{\mathcal{A}}_{\mathcal{H}}$, where B is an instance of H. Let G, of the form $\mathcal{H}$:- $L_1, ..., L_n$, be an X-rule of a subgoal in a system $\mathcal{S}$, where $\mathcal{H} = (H, \phi)$ for some atom H and some conjunction ϕ of AS-constraints. Then we denote by $G^{\mathcal{A}}$ the rule of the form, $(H^{\mathcal{A}}_{\mathcal{H}}, \phi)$:- $L'_1, ..., L'_n$, where for each $i (1 \leq i \leq n)$,

- L'_i is L_i if L_i is not a delayed literal;
- L'_i is $\sim B^B_B$ if L_i is a ground negative delayed literal of the form $\sim B^B$;
- L'_i is L_i if L_i is a delayed literal of the form $B^{\mathcal{A}'}_{\mathcal{H}'}$ or $\sim B^{\mathcal{A}'}_{\mathcal{H}'}$.

We denote by $P(\mathcal{S})$ the program that is the set of all rules $G^{\mathcal{A}}$, where G is an X-rule of a subgoal $\mathcal{A}$ in $\mathcal{S}$. In general, $P(\mathcal{S})$ depends upon P.

For every subgoal $\mathcal{A}$ in a system $\mathcal{S}$ and for every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, $\mathcal{A}$ may or may not have an X-rule with $\mathcal{H}$ in the head. Nevertheless, for technical reasons, we include in the Herbrand base of $P \cup P(\mathcal{S})$ and the Herbrand base of $P(\mathcal{S})$ all ground atoms of the form $B^{\mathcal{A}}_{\mathcal{H}}$ for every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$ and for every atom B in $|\mathcal{H}|$.

2.3. Three-Valued Stable Models

Let **f**, **u**, **t** be truth values ordered by $\mathbf{f} < \mathbf{u} < \mathbf{t}$. An *interpretation I of a program P* is a mapping from $\mathcal{HB}_P$ to $\{\mathbf{f}, \mathbf{u}, \mathbf{t}\}$. I can be represented as a partition, $Pos(I) \cup Und(I) \cup Neg(I)$, of $\mathcal{HB}_P$, where $Pos(I)$ (respectively, $Und(I)$, $Neg(I)$) is the set of ground atoms A such that $I(A) = \mathbf{t}$ (respectively, **u**, **f**). I can also be viewed as the set $Pos(I) \cup \{\sim B | B \in Neg(I)\}$ of ground literals.

Let P_1 and P_2 be programs such that $\mathcal{HB}_{P_1} \subseteq \mathcal{HB}_{P_2}$, and let I be an interpretation of P_2. Then the *restriction of I to P_1*, denoted by $I|_{P_1}$, is the interpretation of P_1 whose mapping is the restriction of I to $\mathcal{HB}_{P_1}$.

An interpretation I is a *model* of a program P if and only if for every rule in its Herbrand instantiation

$$A \text{ :- } L_1, ..., L_n$$

if all L_i's are true in I then A is true in I and if A is false in I then at least one of the L_i's is also false in I.

We assume that there is a special ground atom **u**. Atom **u** is always undefined ($\mathbf{u} \in Und(I)$). It can appear only in the body of a rule in a program. A *non-negative program* is a finite set of rules whose bodies do not contain any negative literals, but may contain atom **u**.

An interpretation I can also be determined by specifying $Pos(I)$ and $Und(I)$. Let P be a program possibly containing undefined atom **u** in the bodies of rules, and I be an interpretation of P. We define $\tau_P(I)$ such that

- $A \in Pos(\tau_P(I))$ if and only if there is a rule A :- $L_1, ..., L_n$ in the Herbrand instantiation of P and all L_i's are true in I;
- $A \in Und(\tau_P(I))$ if $A \notin Pos(\tau_P(I))$ and there is a rule A :- $L_1, ..., L_n$ in the Herbrand instantiation of P and all L_i's are true or undefined in I.

Theorem 2.1 ([7, 23]). Let P be a non-negative program. Then P has a unique least three valued model, denoted by $LPM(P)$. Furthermore, τ_P has a least fixed point, which coincides with $\tau_P \uparrow \omega$ and $LPM(P)$.

Definition 2.8. ([23]) Let P be a program and I be an interpretation of P. The *quotient of P modulo I*, denoted by $\frac{P}{I}$, is the non-negative program obtained from the Herbrand instantiation of P by

- deleting every rule with a negative literal in the body that is false in I; and
- deleting every negative literal in the body of a rule that is true in I; and
- replacing every negative literal with **u** in the body of a rule that is undefined in I.

I is a *three valued stable model* of P if I is the least three valued model $LPM(\frac{P}{I})$. The set of all three valued stable models of P is denoted by $\mathcal{ST}3(P)$.

The notion of three valued stable models is a generalization of both the well founded partial model [30] and the (two-valued) stable models [13].

Theorem 2.2 ([23]). Let P be a program, and $\mathcal{WF}(P)$ be the well founded partial model of P. Then $\mathcal{WF}(P)$ is the smallest three valued stable model of P. Stable models as defined by Gelfond and Lifschitz coincide with two valued stable models.

3. CONSTRAINT ANSWERS

This section describes algorithms for reducing constrained atoms to a normal form, detecting redundant constraint answers and deriving constraint answers of a negative literal from those of its positive counterpart.

3.1. Simple Conjunction of AS-Constraints and Reduced Constrained Atoms

We start with AS-constraints in a constrained atom.

Lemma 3.1. An AS-constraint $A \not\in B$ is valid if and only if A and B are not unifiable, and is unsatisfiable if and only if there exists a substitution θ such that $A = B\theta$.

PROOF. Let $\overline{X}$ and $\overline{Y}$ be variables occurring in A and B, respectively. By Definition 2, $A \not\in B$ is an abbreviation of $\forall \overline{Y}.(A \neq B)$. Since the domain for each variable is the Herbrand universe $\mathcal{HU}$, $A \not\in B$ is valid if and only if for all ground terms $\overline{t}$ and $\overline{s}$, $A[\overline{t}/\overline{X}] \neq B[\overline{s}/\overline{Y}]$, i.e., A and B are not unifiable.

$A \not\in B$ is unsatisfiable if and only if for all ground terms $\overline{t}$, there exist $\overline{s}$ such that $A[\overline{t}/\overline{X}] = B[\overline{s}/\overline{Y}]$. If $\overline{X}$ is $X_1, ..., X_n$, let $\overline{t}$ be $f'(c'), ..., f'^n(c')$, where f' and c' are the new function and constant symbol. Then there exist $\overline{s}$ for $\overline{Y}$ such that $A[\overline{t}/\overline{X}] = B[\overline{s}/\overline{Y}]$. Since c' and f' do not occur in A or B, by replacing terms in $\overline{t}$ with the corresponding variables in $\overline{X}$ on both sides, we derive a substitution

$\{\overline{s'}/\overline{Y}\}$ such that $A = B[\overline{s'}/\overline{Y}]$, where $\overline{s'}$ is obtained from $\overline{s}$ by replacing terms in $\overline{t}$ with the corresponding variables in $\overline{X}$. □

A conjunction of AS-constraints is *simple* if no conjunct is valid or unsatisfiable. An empty conjunction is treated as *true* and an empty disjunction is treated as *false*.

Lemma 3.2. A simple conjunction of AS-constraints is satisfiable [1].

PROOF. Let $A_1 \not\in B_1 \wedge ... \wedge A_n \not\in B_n$ be a simple conjunction of AS-constraints. By Lemma 3.1, no $A_i (1 \leq i \leq n)$ is subsumed by B_i, i.e., $A_i \neq B_i\theta$ for any substitution θ. Let $\overline{X} = X_1, ..., X_k$ be all the free variables in the conjunction, and $\overline{t} = f'(c'), ..., f'^k(c')$ be ground terms that are constructed out of the new constant symbol c' and the new function symbol f'. Then $(A_1 \not\in B_1 \wedge ... \wedge A_n \not\in B_n)[\overline{t}/\overline{X}]$ is true in the Herbrand universe $\mathcal{HU}$. □

Given an atom A as a query, an answer for A is represented by a constrained atom of the form (H, ϕ), which is also written as

$$H \text{ :- } \phi$$

where H is an instance of A and ϕ is a simple conjunction of AS-constraints. Let $\overline{Y}$ be the free variables in H and $\overline{X}$ be all the free variables in ϕ excluding those in $\overline{Y}$. The constraint part of the answer can be represented by $\exists\overline{X}.\phi$. The following lemma shows that the existential quantification of $\overline{X}$ can be pushed into each AS-constraint in ϕ, which is useful for answer reduction.

Theorem 3.1. Let $\phi = \phi_1 \wedge ... \wedge \phi_n$ be a simple conjunction of AS-constraints, $\overline{X}$ and $\overline{Y}$ be a partition of all the free variables of ϕ. Then $\exists\overline{X}.\phi \leftrightarrow (\exists\overline{X}.\phi_1 \wedge ... \wedge \exists\overline{X}.\phi_n)$ is valid.

PROOF. It is obvious that $\exists\overline{X}.\phi \rightarrow (\exists\overline{X}.\phi_1 \wedge ... \wedge \exists\overline{X}.\phi_n)$. For the other direction, suppose that for some $\overline{s}$,

$$(\exists\overline{X}.\phi_1 \wedge ... \wedge \exists\overline{X}.\phi_n)[\overline{s}/\overline{Y}]$$

holds. Let each $\phi_i (1 \leq i \leq n)$ be of the form $A_i(\overline{X}, \overline{Y}) \not\in B_i$. Then $\exists\overline{X}.A_i(\overline{X}, \overline{s}) \not\in B_i$ holds. Let l be the total number of occurrences of variables and function symbols in $\phi[\overline{s}/\overline{Y}]$. Suppose that $\overline{X}$ is of the form $X_1, ..., X_k$. Let $\overline{t}$ be $f'^{l+1}(c'), ..., f'^{l+k}(c')$. Then $A_i(\overline{t}, \overline{s})$ is not an instance of B_i. Otherwise, by replacing $\overline{t}$ with $\overline{X}$, we can derive that $A_i(\overline{X}, \overline{s})$ is an instance of B_i, contradictory with the assumption that $\exists\overline{X}.A_i(\overline{X}, \overline{s}) \not\in B_i$ holds. Therefore $\phi[\overline{t}/\overline{X}, \overline{s}/\overline{Y}]$ holds, and so does $\exists\overline{X}.\phi[\overline{s}/\overline{Y}]$. □

Using the semantic properties of AS-constraints, we show that constrained atoms can be reduced to a normal form and redundant answers can be detected using basic operations such as variant checking and set membership.

Definition 3.1. A constrained atom $\mathcal{A}$ of the form (A, ϕ) is *reduced* if

[1] It should be mentioned that Lemma 3.2 holds in the Herbrand universe $\mathcal{HU}$, but not in the domain of all ground terms constructed from constant and function symbols in a program. For instance, the simple conjunction

$$(p(X,Y) \not\in p(U,U)) \wedge (p(Y,Z) \not\in p(V,V)) \wedge (p(X,Z) \not\in p(W,W))$$

is unsatisfiable in the domain $\{a, b\}$, but is satisfiable in $\mathcal{HU}$.

- ϕ is a simple conjunction of AS-constraints; and
- every AS-constraint in ϕ contains a free variable in A; and
- no two AS-constraints in ϕ are variants of each other up to renaming of variables not occurring in A.

Lemma 3.3. For any constrained atom $\mathcal{A}$, there exists a reduced constrained atom $\mathcal{A}'$ such that $|\mathcal{A}| = |\mathcal{A}'|$.

PROOF. Let $\mathcal{A}$ be of the form (A, ϕ) and ϕ be of the form $\phi_1 \wedge ... \wedge \phi_n$. Let $\overline{X}$ be all free variables of ϕ that do not occur in A. By Theorem 3.1,

$$\exists \overline{X}.\phi \leftrightarrow \exists \overline{X}.\phi_1 \wedge ... \wedge \exists \overline{X}.\phi_n$$

$\mathcal{A}'$ is obtained from $\mathcal{A}$ according to Definition 1. □

Let $\mathcal{A}$ be a constrained atom. We call $\mathcal{A}'$ a *reduced form* of $\mathcal{A}$ if $\mathcal{A}'$ is reduced and $|\mathcal{A}'| = |\mathcal{A}|$.

Lemma 3.4. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two reduced constrained atoms. If $\mathcal{A}_1$ is subsumed (resp. AS-subsumed) by $\mathcal{A}_2$, then $|\mathcal{A}_1| \subseteq |\mathcal{A}_2|$, and if $\mathcal{A}_1$ and $\mathcal{A}_2$ are variants, then $|\mathcal{A}_1| = |\mathcal{A}_2|$.

PROOF. It follows from definitions and Theorem 3.1. □

The subsumption or AS-subsumption of constrained atoms can be used to detect redundant answers.

Example 3.1. Consider the following recursive and function-free program:

$p(X)$:- $\sim q(X, Y), p(Y)$.
$p(a)$.
$q(X, X)$.

The query $p(X)$ has an infinite number of answers represented as constrained atoms:

$p(a)$.
$p(X)$:- $q(X, a) \not\in q(V, V)$.
$p(X)$:- $\exists \overline{Y}.(q(X, Y_1) \not\in q(V_1, V_1) \wedge (\bigwedge_{1 \leq i < n} (q(Y_i, Y_{i+1}) \not\in q(V_{i+1}, V_{i+1}))) \wedge q(Y_n, a) \not\in q(V_{n+1}, V_{n+1}))$.

where $n \geq 1$. By Theorem 3.1, the existential quantification of Y_i's can be pushed into individual AS-constraint. Since only $q(X, Y_1) \not\in q(V_1, V_1)$ has a free variable X occurring in the atom $p(X)$, the last (series of) answer can be reduced to:

$p(X)$:- $\exists Y_1.q(X, Y_1) \not\in q(V_1, V_1)$.

allowing query evaluation to terminate.

3.2. Deriving Answers of Negative Literals

For constrained atoms to be sufficient for answer representation, they have to be closed under negation. Applying negation to an existentially quantified conjunction of AS-constraints can result in quantified equations. We show how to solve quantified equations and derive constraint answers of negative literals.

Let ϕ be a simple conjunction of AS-constraints of the form $(A_1 \not\in H_1 \wedge ... \wedge A_n \not\in H_n)$ and $\overline{Z}$ be some free variables in ϕ. By Theorem 3.1,

$$\exists \overline{Z}.\phi \leftrightarrow \exists \overline{Z}.(A_1 \not\in H_1) \wedge ... \wedge \exists \overline{Z}.(A_n \not\in H_n)$$

Suppose that each H_i has variables $\overline{W}_i$, where $1 \leq i \leq n$. Then

$$\sim \exists \overline{Z}.\phi \leftrightarrow \forall \overline{Z} \exists \overline{W}_1.(A_1 = H_1) \vee ... \vee \forall \overline{Z} \exists \overline{W}_n.(A_n = H_n)$$

We introduce a modification of the unification algorithm [18] for solving quantified equations, where all universal quantifiers precede existential quantifiers [2]. Each disjunct $\forall \overline{Z} \exists \overline{W}_i (A_i = H_i)$ can be transformed into a substitution.

Let E be a conjunction of equations, which can be viewed also as a finite set of equations. E is in *solved form* if E is $\{X_1 = t_1, ..., X_n = t_n\}$, where all X_i's are distinct variables and do not occur in t_i's. We solve quantified equation sets of the form $\forall \overline{U} \exists \overline{V}.E$, where $\overline{U}$ and $\overline{V}$ are disjoint sets of variables that occur in E. (Quantified variables that do not occur in E can be eliminated.) All variables in E that are not in $\overline{U}$ or $\overline{V}$ are called free variables. The algorithm proceeds nondeterministically by choosing an equation $e \in E$ to which it applies the following transformations when they become applicable:

1. For $f(t_1, \ldots, t_n) = g(s_1, \ldots, s_m)$, if f and g are identical function symbols, then $n = m$ and replace the equation by $t_1 = s_1$, ..., $t_n = s_n$; otherwise halt with failure.

2. For $X = X$, where X is a variable, delete the equation.

3. For $t = X$, where t is not a variable and X is a variable, replace the equation by $X = t$.

4. For $X = Y$, where X and Y are distinct variables, there are several cases:

   a. if X is in $\overline{V}$ and X occurs in other equations, replace X by Y wherever it occurs in other equations;
   If X is not in $\overline{V}$ but Y is in $\overline{V}$ and Y occurs in other equations, replace $X = Y$ by $Y = X$ and replace Y by X wherever it occurs in other equations;
   b. if both X and Y are in $\overline{U}$, or one of X and Y is in $\overline{U}$ and the other is a free variable, then halt with failure;
   c. otherwise, both X and Y are free variables. If X occurs in other equations, replace X by Y wherever it occurs in other equations.

5. For $X = t$, where X is a variable and t is a term that is not a variable, there are several cases:

[2] Comon and Lescanne [9] considered equational problems that involved Boolean operators, but with no existential quantifiers inside the scopes of universal quantifiers.

d. if X appears in t, or X is in $\overline{U}$, then halt with failure;
e. if X is in $\overline{V}$ and X does not occur in t and X occurs in other equations, replace X by t wherever it occurs in other equations;
f. if X is a free variable, we consider several cases for t:
   f.1. If t contains some variables in $\overline{U}$, then halt with failure;
   f.2. If t is of the form $f(t_1, ..., t_n)$, containing some variable in $\overline{V}$ but no variable in $\overline{U}$, then replace the equation with $X = f(Z_1, ..., Z_n)$ and $Z_1 = t_1, ..., Z_n = t_n$ and replace X by $f(Z_1, ..., Z_n)$ in other equations, where $Z_1, ..., Z_n$ are new distinct variables;
   f.3. Otherwise, all variables (if any) in t are free variables. If X occurs in other equations, then replace X by t wherever it occurs in other equations.

The algorithm terminates when no further transformation can be applied or when failure is reported.

Theorem 3.2. The extended unification algorithm applied to $\forall\overline{U}.\exists\overline{V}.E$*, where* E *is a finite set of equations, returns failure if and only if* $\forall\overline{U}.\exists\overline{V}.E$ *is unsatisfiable. Otherwise, it returns a finite set of equations* E^* *in solved form such that*

$$\forall\overline{U}.\exists\overline{V}.E \leftrightarrow \forall\overline{U}.\exists\overline{V}.E^*$$

The proof of Theorem 3.2 is in the appendix. The set of equations in solved form returned by the modified unification algorithm satisfies the following properties:

- free variables are bound to terms with free variables only; and
- there is no binding for any universally quantified variable.

Let E^* be $E^*_{free} \cup E^*_{exist}$, where E^*_{free} is the set of all equations in E^* for bindings of free variables and E^*_{exist} be the set of all equations in E^* for bindings of existentially quantified variables in $\overline{V}$. Then

$$\forall\overline{U}.\exists\overline{V}.E \leftrightarrow \forall\overline{U}.\exists\overline{V}.E^* \leftrightarrow E^*_{free} \wedge \forall\overline{U}.\exists\overline{V}.E^*_{exist} \leftrightarrow E^*_{free}$$

Therefore $\forall\overline{U}.\exists\overline{V}.E$ can be reduced to a substitution.

As an example, $\forall X.\exists Y.f(g(X), a) = f(Y, Z)$, where Z is a free variable, can be reduced to $\forall X.\exists Y.(Y = g(X) \wedge Z = a)$. By omitting the binding for the existentially quantified variable Y, we obtain $Z = a$.

Definition 3.2. Let $\mathcal{S}$ be a system and A' be an atom that is a subgoal in $\mathcal{S}$ such that all X-rules of A' in $\mathcal{S}$ are answers. Let $\mathcal{H}_i = (H_i, \phi_i)(1 \leq i \leq m, m \geq 0)$ be all the constrained atoms that are answers of A' and $\mathcal{H}_i = (H_i, \phi_i)(m+1 \leq i \leq n, n \geq 0)$ be all the constrained atoms that occur in the head of some answer of A' with delayed literals. Let $\overline{Z}_i (1 \leq i \leq n)$ be all free variables in ϕ_i that do not occur in H_i. Let

$$\sim \left(\bigvee_{1 \leq i \leq m} (A' = H_i \wedge \exists\overline{Z}_i.\phi_i) \vee \bigvee_{m+1 \leq i \leq n} (A' = H_i \wedge \exists\overline{Z}_i.\phi_i \wedge (H_i)^{A'}_{\mathcal{H}_i}) \right) \quad (1)$$

be converted into a disjunction of the form:

$$\bigvee_{1\leq j\leq k}(\theta_j \wedge \phi_j \wedge D_j) \tag{2}$$

where $k \geq 1$, θ_j is a variable substitution, ϕ_j is a simple conjunction of AS-constraints, and D_j is either empty or a negative delayed literal $\sim(H')^{A'}_{\mathcal{H}_i}$ for some $i(m+1 \leq i \leq n)$. Then we call

$$(\sim A'\theta_j, \phi_j) \text{ :- } D_j$$

for each $j(1 \leq j \leq k)$ an *answer of* $\sim A'$ *in* $\mathcal{S}$.

The intuition is the following equivalence:

$$A' \leftrightarrow \bigvee_{1\leq i\leq m}(A' = H_i \wedge \exists\overline{Z}_i.\phi_i) \vee \bigvee_{m+1\leq i\leq n}(A' = H_i \wedge \exists\overline{Z}_i.\phi_i \wedge (H_i)^{A'}_{\mathcal{H}_i})$$

By applying negation on both sides, we obtain:

$$\begin{aligned}\sim A' \leftrightarrow \quad & \textstyle\bigwedge_{1\leq i\leq m}(A \not\in H_i \vee (A' = H_i \wedge \sim\exists\overline{Z}_i.\phi_i)) \wedge \\ & \textstyle\bigwedge_{m+1\leq i\leq n}(A' \not\in H_i \vee (A' = H_i \wedge (\sim\exists\overline{Z}_i.\phi_i \vee (\exists\overline{Z}_i.\phi_i \wedge \sim(H_i)^{A'}_{\mathcal{H}_i}))))\end{aligned} \tag{3}$$

Formulas $\sim\exists\overline{Z}_i.\phi_i$ can be converted into a variable substitution using the extended unification algorithm. Thus the formula in (1) can be converted into a disjunction in (2).

4. CONSTRUCTIVE NEGATION AND TABLED EVALUATION

Tabled evaluation has been used successfully for effective query processing under the well-founded semantics [1, 10, 7, 28]. In particular, SLG resolution [7] not only has various desirable theoretical properties, including goal-orientedness, polynomial time data complexity, answer sharing and preservation of all three-valued stable models, but also has been implemented efficiently [6, 25, 27], delivering excellent performance for query evaluation. This section discusses informally how to extend SLG resolution [7] with constructive negation. For simplicity, we consider query evaluation for only function-free programs.

4.1. Tabled Evaluation with Constraints

Given a function-free program P and a query atom Q, SLG resolution [7] transforms rules in P that are relevant to Q into answers. An intermediate state of query evaluation, called a *system*, is represented as a set of pairs of the form $(\mathcal{A} : \Gamma)$, where $\mathcal{A}$ is a subgoal and Γ is a multiset of annotated X-rules.

In SLG resolution [7], a subgoal is an atom and two subgoals are identical if they are variants of each other. When an atom A is selected from the body of a rule, a transformation in SLG resolution called NEW SUBGOAL introduces A as a new subgoal into a system only if it is not identical to any existing subgoal. This avoids possible loops evaluating the same subgoal repeatedly. (Formal definitions of all transformations will be presented in the next section.)

With AS-constraints, we choose to represent a subgoal as a constrained atom. The AS-constraints from a calling environment restrict further the search space for answers of a subgoal. AS-subsumption is used for detecting repeated subgoals. That is, a new subgoal is created if it is not AS-subsumed by any subgoal in the current table. Other notions of subgoals and redundant subgoals are also possible, e.g., atoms and subsumption of (constrained) atoms, with different implementation tradeoffs.

The NEW SUBGOAL transformation ensures that each subgoal A is evaluated only once using rules in a program P. Every occurrence of a selected atom A in the body of a rule is solved using answers from a global table instead of using rules from a program. A transformation called POSITIVE RETURN returns each new answer of A to every rule that has a selected atom A. The annotation associated with an X-rule keeps track of what answers have been returned.

With AS-constraints, an answer can be a constrained atom in general. Reduction and AS-subsumption of constrained atoms allow us to detect and eliminate redundant answers, avoiding possible loops generating redundant answers of the same subgoal.

It is possible to have subgoals that depend upon each other even if the program P is positive. When all answers have been computed and returned to rules with corresponding selected atoms, these rules with selected atoms are disposed by a transformation called COMPLETION, leaving only answers of subgoals in a system. The three transformations, namely NEW SUBGOAL, POSITIVE RETURN and COMPLETION, are sufficient for positive programs.

Example 4.1. Consider a query $p(X,Y)$ with respect to the following program:

$p(X,Y)$:- $r(X,Y)\not\in r(X_1,X_1), q(X,Y)$.
$p(X,Y)$:- $s(X,Y)\not\in s(a,b)$.
$q(X,X)$:- $t(X)$.
$q(X,Y)$:- $p(X,Y)$.

Initially the system is empty. The first transformation to be applied is always NEW SUBGOAL that introduces a new subgoal into a system. The initial subgoal is $p(X,Y)$ from the query and its rules are obtained by resolving $p(X,Y)$:- $p(X,Y)$, on $p(X,Y)$ in the body, with rules in the program. The new system has a single subgoal:

$$p(X,Y): \quad \left[\begin{array}{ll} p(X,Y) \text{ :- } r(X,Y)\not\in r(X_1,X_1), q(X,Y) & \{\} \\ p(X,Y) \text{ :- } s(X,Y)\not\in s(a,b) & \end{array}\right]$$

(To save space, we show only rules of individual subgoals instead of the entire system every time.) The first rule of $p(X,Y)$ has a selected atom $q(X,Y)$. To keep track of answers that have been returned, each rule with a selected atom is annotated by a set of constrained atoms, where two constrained atoms are considered identical if they are variants of each other. The annotation is initially empty, represented by $\{\}$. (For convenience we write an annotated X-rule of the form $\langle G, \Sigma(G)\rangle$ as G followed by $\Sigma(G)$ in all examples.)

Even though the selected atom is $q(X,Y)$, it is also restricted by the constraints in the body of the rule for $p(X,Y)$. Thus the new subgoal is $(q(X,Y), r(X,Y)\not\in r(X_1,X_1))$:

$$(q(X,Y), r(X,Y)\not\in r(X_1,X_1)): \quad q(X,Y) \text{ :- } r(X,Y)\not\in r(X_1,X_1), p(X,Y) \quad \{\}$$

Notice that due to the propagation of the constraint $r(X,Y){\not\in}r(X_1,X_1)$ from the calling environment to $q(X,Y)$, the program rule $q(X,X)$:- $t(X)$ does not generate any rule for the subgoal, avoiding the potentially expensive evaluation of $t(X)$.

Similarly $p(X,Y)$ is selected, which is restricted by the constraint $r(X,Y){\not\in}r(X_1,X_1)$. However, no new subgoal is created since $(p(X,Y), r(X,Y){\not\in}r(X_1,X_1))$ is AS-subsumed by $p(X,Y)$. Therefore the subgoal for the selected atom $p(X,Y)$ in this case is $p(X,Y)$, not $(p(X,Y), r(X,Y){\not\in}r(X_1,X_1))$. In other words, the subgoal for a selected atom is determined by both the relevant rule and the subgoals that exist in the global table when the atom is selected.

The answer for $p(X,Y)$ can be returned to the selected $p(X,Y)$ in the rule for $q(X,Y)$:

$$(q(X,Y), r(X,Y){\not\in}r(X_1,X_1)) : \left[\begin{array}{ll} q(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), p(X,Y) & \{p(X,Y) \text{ :- } s(X,Y){\not\in}s(a,b)\} \\ q(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), s(X,Y){\not\in}s(a,b) & \end{array}\right]$$

Similarly the answer for $(q(X,Y), r(X,Y){\not\in}r(X_1,X_1))$ can be returned to the selected $q(X,Y)$ in the rule for $p(X,Y)$, generating another answer for $p(X,Y)$:

$$p(X,Y) : \left[\begin{array}{l} p(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), q(X,Y) \\ \qquad\qquad \{q(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), s(X,Y){\not\in}s(a,b)\} \\ p(X,Y) \text{ :- } s(X,Y){\not\in}s(a,b) \\ p(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), s(X,Y){\not\in}s(a,b) \end{array}\right]$$

With variant checking of repeated answers, the new answer for $p(X,Y)$ is also returned to the subgoal for $q(X,Y)$, but no new answer is generated thanks to elimination of redundant answers:

$$(q(X,Y), r(X,Y){\not\in}r(X_1,X_1)) : \left[\begin{array}{l} q(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), p(X,Y) \\ \qquad\qquad \left\{\begin{array}{l} p(X,Y) \text{ :- } s(X,Y){\not\in}s(a,b) \\ p(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), s(X,Y){\not\in}s(a,b) \end{array}\right\} \\ q(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), s(X,Y){\not\in}s(a,b) \end{array}\right]$$

At this point, the set of subgoals $\{p(X,Y), (q(X,Y), r(X,Y){\not\in}r(X_1,X_1))\}$ is completely evaluated for two reasons:

- it is self contained in the sense that subgoals in the set depend upon each other and only through selected atoms; and
- all answers have been returned as indicated by the annotations associated with rules that have a selected atom.

The COMPLETION transformation is applied to the set of subgoals, disposing all rules that have a selected atom. The final system for the evaluation of the initial query $p(X,Y)$ is as follows:

$$\left\{\begin{array}{ll} p(X,Y) : & \left[\begin{array}{l} p(X,Y) \text{ :- } s(X,Y){\not\in}s(a,b) \\ p(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), s(X,Y){\not\in}s(a,b) \end{array}\right] \\ (q(X,Y), r(X,Y){\not\in}r(X_1,X_1)) : & q(X,Y) \text{ :- } r(X,Y){\not\in}r(X_1,X_1), s(X,Y){\not\in}s(a,b) \end{array}\right\}$$

4.2. Constructive Negation and Recursion through Negation

For stratified negation [21], a transformation for negation as failure or constructive negation can be added. The real challenge is to handle recursion through negation. The reason is that given a fixed computation rule, e.g., left-most, the truth values of the instances of a negative literal and the constraints of its variables may not be completely determined when the negative literal is selected.

In SLG resolution[7], where negation as failure is used, a transformation called DELAYING is introduced that postpones the application of negation as failure to a ground negative literal. This allows query evaluation to proceed and continue to solve the remaining literals in a rule body. Answers of a subgoal may now contain delayed ground negative literals. These delayed ground negative literals may turn out to be true or false later. Query evaluation then consists of a goal-oriented phase coupled with a bottom-up phase. The goal-oriented phase solves positive literals and propagates their variable bindings and uses negation as failure whenever possible. The bottom-up phase determines and propagates the truth values of delayed ground negative literals, using two additional transformations called SIMPLIFICATION and ANSWER COMPLETION. Since all variable bindings have been propagated in the goal-oriented phase, the bottom-up phase essentially deals with a ground program. This approach avoids repeated computation inside a subgoal and is carefully formalized in SLG resolution [7].

With constructive negation, non-ground negative literals may also be involved in recursions through negation. Unlike the truth value of a ground negative literal, the constraints of a non-ground negative literal may have to be determined incrementally through several iterations. Consequently constructive negation may be applied to the same negative literal several times, causing repeated computation within a subgoal.

Our approach is motivated primarily by implementation considerations and tries to avoid repeated computation within a subgoal by delaying only ground negative literals. This means that variable bindings and constraints can be propagated through constructive negation once and for all and that only the truth values of delayed ground negative literals need to be propagated iteratively. Still constructive negation has to deal with constraint answers that are in general three-valued. We show next how constructive negation interacts with delayed ground negative literals.

Example 4.2. Let the query $m(X)$ be evaluated with respect to the following program assuming a left-most computation rule:

$m(X)$:- $\sim q(X)$.
$q(a)$:- $\sim r$.
$q(b)$:- $\sim s$.
r :- $\sim s, r$.
s :- $\sim r$.

The system is as follows after the initial subgoal $m(X)$ is created:

$$\{\ m(X):\quad m(X) \text{ :- } \sim q(X)\ \}$$

Several applications of NEW SUBGOAL lead to a new system:

$$\left\{\begin{array}{ll} m(X): & m(X) \text{ :- } \sim q(X) \\ q(X): & \left[\begin{array}{l} q(a) \text{ :- } \sim r \\ q(b) \text{ :- } \sim s \end{array}\right] \\ r: & r \text{ :- } \sim s, r \\ s: & s \text{ :- } \sim r \end{array}\right\}$$

Following SLG resolution [7], we delay ground negative literals to break cycles through negation. A delayed ground negative literal is denoted by $\sim B^B$, where B is a ground atom. Delayed literals are never selected by a computation rule, although they could be simplified away later if they turn out to be true or false. The new system is as follows:

$$\left\{\begin{array}{ll} m(X): & m(X) \text{ :- } \sim q(X) \\ q(X): & \left[\begin{array}{l} q(a) \text{ :- } \sim r^r \\ q(b) \text{ :- } \sim s^s \end{array}\right] \\ r: & r \text{ :- } \sim s^s, r \ \ \{\} \\ s: & s \text{ :- } \sim r^r \end{array}\right\}$$

Notice that the subgoal $q(X)$ is completely evaluated with two answers with delayed literals. By Definition 2, we derive three answers of $\sim q(X)$ in the system:

$$\begin{array}{l} (\sim q(X), (q(X) \not\in q(a) \wedge q(X) \not\in q(b))) \\ \sim q(a) \text{ :- } \sim q(a)^{q(X)}_{q(a)} \\ \sim q(b) \text{ :- } \sim q(b)^{q(X)}_{q(b)} \end{array}$$

Solving $\sim q(X)$ by constructive negation leads to the following rules for $m(X)$:

$$m(X): \quad \left[\begin{array}{l} m(X) \text{ :- } q(X) \not\in q(a), q(X) \not\in q(b) \\ m(a) \text{ :- } \sim q(a)^{q(X)}_{q(a)} \\ m(b) \text{ :- } \sim q(b)^{q(X)}_{q(b)} \end{array}\right]$$

The evaluation of subgoal r continues with the selection of r in its rule body. An application of COMPLETION to the singleton set $\{r\}$ of subgoals disposes the rule for r. Since r is completely evaluated with no answers, all occurrences of $\sim r^r$ can be deleted. The final system is as follows after all delayed literals have been simplified away using SIMPLIFICATION:

$$\left\{\begin{array}{ll} m(X): & \left[\begin{array}{l} m(X) \text{ :- } q(X) \not\in q(a), q(X) \not\in q(b) \\ m(b) \end{array}\right] \\ q(X): & q(a) \\ r: & \\ s: & s \end{array}\right\}$$

Our approach avoids repeated computation inside a subgoal and is different from that in [10]. In [10], two different contexts are used, one for computing *true* answers (in a T-search tree) and the other for computing *possibly true* answers (in a TU-search tree). A negative literal in a T-search tree is resolved using possible true

answers, while a negative literal in a TU-search tree is resolved using true answers. Initially nothing is definitely true and everything is possibly true. For the query $m(X)$ in Example 2, the first iteration will derive possibly true answers of $q(a)$, $q(b)$, and s, and no definitely true answers. With the refined set of true and possibly true answers from the first iteration, the second iteration will derive one true answer for $m(X)$, i.e., $m(X)$:- $q(X)\not\in q(a), q(X)\not\in q(b)$, possibly true answers $q(a)$ and $q(b)$, and s as both true and possibly true. The iterative process continues until a fixed point is reached for the set of true and possibly true answers. Constructive negation may be applied to $\sim q(X)$ multiple times whenever $q(X)$ has a different set of possibly true answers, causing repeated computation.

We are able to avoid repeated computation when loops through negation can be broken by delaying *ground* negative literals. In the most general case, however, non-ground negative literals may have to be delayed. For example, when a query $p(X)$ is evaluated with respect to the following program:

$p(X)$:- $\sim q(X)$.
$p(a)$.
$q(X)$:- $\sim p(X)$.
$q(b)$.

there is a cycle through negation between $p(X)$ and $q(X)$. Variable bindings and constraints for $\sim p(X)$ and $\sim q(X)$ have to be propagated iteratively. It remains an open problem how to support constructive negation under the well-founded semantics in general while avoiding any repeated computation within a subgoal.

5. TRANSFORMATIONS AND SLG_{CN} DERIVATIONS

SLG resolution [7] is a goal-oriented method of tabled evaluation for normal logic programs. In this paper, we consider query evaluation with constructive negation for function-free programs and extend SLG resolution with AS-constraints and constructive negation.

In Section 2 we already extended the notions of subgoals, X-rules, and systems in SLG resolution with AS-constraints. With the definition of reduced constrained atoms in Section 3.1, we assume that each subgoal is a reduced constrained atom and each constrained atom in $\Sigma(G)$ of an annotated X-rule $\langle G, \Sigma(G)\rangle$ is also reduced. This section continues with formal definitions of an extension of SLG resolution with AS-constraints.

Definition 5.1. Let P be a program, R be a computation rule, $\mathcal{S}$ be a system and G be an X-rule of some subgoal in $\mathcal{S}$, of the form (H, ϕ) :- $L_1, ..., L_n$, where $n > 0$, such that G is not an answer. Suppose that L_i is selected for some $i(1 \leq i \leq n)$ when the computation rule R is applied to G. We say that $\mathcal{A}$ is the *subgoal of the selected literal* L_i if

- L_i is a negative literal of the form $\sim B$ and $\mathcal{A}$ is B; or
- L_i is an atom B and $\mathcal{A}$ is either some subgoal in $\mathcal{S}$ that AS-subsumes (B, ϕ) or (B, ϕ) if no subgoal in $\mathcal{S}$ AS-subsumes (B, ϕ).

The notion of the subgoal of the selected literal of an X-rule embodies the idea of AS-subsumption for subgoals.

Definition 5.2. [X-resolution] Let G be an X-rule, of the form (H, ϕ) :- $L_1, ..., L_n$, and L_i be the selected atom of G for some $i(1 \leq i \leq n)$. Let C be an X-rule and C', of the form (H', ϕ') :- $L'_1, ..., L'_m$, be a variant of C with variable renamed so that G and C' have no variables in common. Then G is *X-resolvable* with C if L_i and H' have a most general unifier θ and $(\phi \wedge \phi')\theta$ is satisfiable. The X-rule:

$$((H, \phi \wedge \phi') \text{ :- } L_1, ..., L_{i-1}, L'_1, ..., L'_m, L_{i+1}, ..., L_n)\theta$$

is the *X-resolvent* of G with C if G is X-resolvable with C.

X-resolution is used for resolution with a rule in a program or with an answer of a subgoal that is a fact. If an answer has delayed literals, *X-factoring* is used to return the answer to the selected atom of an X-rule.

Definition 5.3. [X-factoring] Let $\mathcal{S}$ be a system. Let G be an X-rule of a subgoal $\mathcal{A}$ in $\mathcal{S}$, of the form (H, ϕ) :- $L_1, ..., L_n$, and let L_i be the selected atom of G for some $i(1 \leq i \leq n)$. Let $\mathcal{A}'$ be the subgoal of L_i and C be an answer of $\mathcal{A}'$ in $\mathcal{S}$, and C', of the form (H', ϕ') :- $L'_1, ..., L'_m$, be a variant of C with variables renamed so that G and C' have no variables in common, and $m > 0$. Then the X-rule:

$$(H\theta, \phi\theta \wedge \phi'\theta) \text{ :- } L_1\theta, ..., L_{i-1}\theta, (L_i\theta)^{\mathcal{A}'}_{(H', \phi')}, L_{i+1}\theta, ..., L_n\theta$$

is the *X-factor* of G with C, where θ is the most general unifier of L_i and H'.

Definition 5.4. Let $\mathcal{S}$ be a system and let $(\mathcal{A} : \Gamma) \in \mathcal{S}$, where $\mathcal{A}$ is a subgoal and Γ is a multiset of annotated X-rules. Then

- $\mathcal{A}$ *succeeds* if Γ contains a fact $\mathcal{H}$ that is a variant of $\mathcal{A}$;
- $\mathcal{A}$ *fails* if $\Gamma = \{\}$;
- $\mathcal{A}$ is *completed* if all annotated X-rules in Γ are answers;
- a positive delayed literal of the form $B^{\mathcal{A}}_{\mathcal{H}}$ is *successful* if Γ contains a fact $\mathcal{H}$; $B^{\mathcal{A}}_{\mathcal{H}}$ is *failed* if $\mathcal{A}$ is completed and does not have any answer in Γ with $\mathcal{H}$ in the head.

Let B be a ground subgoal in $\mathcal{S}$. Then $\sim B^{B}$ is *successful* if B fails, and is *failed* if B succeeds. A system $\mathcal{S}$ is *completed* if every subgoal in $\mathcal{S}$ is completed.

Definition 5.5. [CN-resolution] Let $\mathcal{S}$ be a system and A be an atom that is a completed subgoal in $\mathcal{S}$. Let G be an X-rule, of the form (H, ϕ) :- $L_1, ..., L_n$, and L_i be the selected literal of the form $\sim A$. Let C be a variant of an answer of $\sim A$ in $\mathcal{S}$, of the form $(\sim A', \phi')$:- L' where L' is either empty or a delayed literal, such that G and C have no variables in common. Then G is *CN-resolvable with*

C if L_i and $\sim A'$ have a most general unifier θ and $(\phi \wedge \phi')\theta$ is satisfiable. The X-rule:

$$((H, \phi \wedge \phi') \text{ :- } L_1, ..., L_{i-1}, L', L_{i+1}, ..., L_n)\theta$$

is the *CN-resolvent of G with C* if G is CN-resolvable with C.

Definition 5.6. Let $\mathcal{S}$ be a system and Λ be a non-empty set of subgoals in $\mathcal{S}$ that are not completed. Λ is said to be *completely evaluated* if for every subgoal $\mathcal{A} \in \Lambda$, either

- $\mathcal{A}$ succeeds; or
- let $\mathcal{A} : \Gamma \in \mathcal{S}$, where Γ is a multiset of annotated X-rules, and every annotated X-rule in Γ that is not an answer of $\mathcal{A}$ is of the form $\langle G, \Sigma(G)\rangle$ such that
  - G has a selected atom L, and
  - the subgoal $\mathcal{A}'$ for the selected atom L is completed or in Λ, and
  - for every constrained atom $\mathcal{H}$ that occurs in the head of some answer of $\mathcal{A}'$ in $\mathcal{S}$, $\mathcal{H} \in \Sigma(G)$.

The following definition will be used in the transformation ANSWER COMPLETION.

Definition 5.7. Let $\mathcal{S}$ be a system, and let $\mathcal{A}$ be a subgoal in $\mathcal{S}$ and $\mathcal{H}$ be a constrained atom that occurs in the head of some answer of $\mathcal{A}$. Then $\mathcal{H}$ is *supported* by $\mathcal{A}$ if either

- $\mathcal{A}$ is not completed; or
- there is an answer G of $\mathcal{A}$ with $\mathcal{H}$ in the head such that for every positive delayed literal of the form $B^{\mathcal{A}_1}_{\mathcal{H}_1}$ in the body of G, $\mathcal{H}_1$ is supported by $\mathcal{A}_1$.

Starting with the empty system of subgoals, each transformation transforms one system into another. Let P be a program, R be an arbitrary computation rule, and Q be a query atom. The following are all the transformations in SLG resolution that are tailored to function-free programs and extended with AS-constraints. $\mathcal{S}$ and $\mathcal{S}'$ are systems and Γ's possibly with subscripts are multisets of annotated X-rules.

- NEW SUBGOAL Let $\mathcal{A}$ be a constrained atom that is either Q or the subgoal of the selected literal of some X-rule of some subgoal in $\mathcal{S}$ such that $\mathcal{A}$ is not AS-subsumed by any subgoal in $\mathcal{S}$. Let A' be the atom constrained in $\mathcal{A}$. Then

$$\frac{\mathcal{S}}{\mathcal{S} \cup \{(\mathcal{A} : \Gamma)\}}$$

  where Γ is a multiset of annotated X-rules that contains, for each X-resolvent G of $\mathcal{A}$:- A' with a rule in P,

- G if G does not have a selected atom, or
- $\langle G, \{\}\rangle$ if G has a selected atom.

- POSITIVE RETURN Let $\mathcal{S} = \{(\mathcal{A} : \{\langle G, \Sigma(G)\rangle\} \cup \Gamma_{\mathcal{A}})\} \cup \mathcal{S}'$, where G has a selected atom, and $\mathcal{A}'$ be the subgoal of the selected atom of G and $(\mathcal{A}' : \{C\} \cup \Gamma_{\mathcal{A}'}) \in \mathcal{S}$, and C be an answer of $\mathcal{A}'$ with a constrained atom $\mathcal{H}$ in the head such that $\mathcal{H} \notin \Sigma(G)$. Let G' be the X-resolvent of G with C if C is a fact, or the X-factor of G with C if C has some delayed literals in its body. Then

$$\frac{\mathcal{S}}{\{(\mathcal{A} : \{\langle G, \Sigma(G) \cup \{\mathcal{H}\}\rangle\} \cup \Gamma_C \cup \Gamma_{\mathcal{A}})\} \cup \mathcal{S}'}$$

  where Γ_C is the singleton set $\{G'\}$ if G' does not have a selected atom or the singleton set $\{\langle G', \{\}\rangle\}$ if G' has a selected atom.

- CONSTRUCTIVE NEGATION Let $\mathcal{S} = \{(\mathcal{A} : \{G\} \cup \Gamma_{\mathcal{A}})\} \cup \mathcal{S}'$ and G have a selected negative literal $\sim A'$ such that A' is a completed subgoal in $\mathcal{S}$. Let Γ' be the multiset of annotated X-rules that contains, for every CN-resolvent G' of G with an answer of $\sim A'$, either G' itself if G' does not have a selected atom or $\langle G', \{\}\rangle$ if G' has a selected atom. Then

$$\frac{\mathcal{S}}{\{(\mathcal{A} : \Gamma' \cup \Gamma_{\mathcal{A}})\} \cup \mathcal{S}'}$$

- DELAYING Let $\mathcal{S} = \{(\mathcal{A} : \{G\} \cup \Gamma_A)\} \cup \mathcal{S}'$ and G have a selected ground negative literal $\sim B$. Then

$$\frac{\mathcal{S}}{\{(\mathcal{A} : \{G'\} \cup \Gamma_{\mathcal{A}})\} \cup \mathcal{S}'}$$

  where G' is G with the selected ground negative literal $\sim B$ replaced by $\sim B^B$.

- COMPLETION

$$\frac{\Lambda \text{ is a non-empty set of subgoals in } \mathcal{S} \text{ that is completely evaluated}}{\text{all X-rules of subgoals in } \Lambda \text{ that are not answers are deleted}}$$

- SIMPLIFICATION Let $\mathcal{S} = \{(\mathcal{A} : \{G\} \cup \Gamma_{\mathcal{A}})\} \cup \mathcal{S}'$ and G have a delayed literal L. Then

$$\frac{\mathcal{S}}{\{(\mathcal{A} : \{G'\} \cup \Gamma_{\mathcal{A}})\} \cup \mathcal{S}'} \text{ if } L \text{ is successful} \qquad \frac{\mathcal{S}}{\{(\mathcal{A} : \Gamma_{\mathcal{A}})\} \cup \mathcal{S}'} \text{ if } L \text{ is failed}$$

  where G' is G with L deleted.

- ANSWER COMPLETION Let $\mathcal{A}$ be a subgoal in $\mathcal{S}$ and $\mathcal{H}$ be a constrained atom that occurs in the head of an answer of $\mathcal{A}$ such that $\mathcal{H}$ is not supported by $\mathcal{A}$. Then

$$\frac{\mathcal{S}}{\text{delete all answers of } \mathcal{A} \text{ with } \mathcal{H} \text{ in the head}}$$

Definition 5.8. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. An SLG_{CN} *derivation* for Q is a finite sequence of systems $\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_n$ such that:

- $\mathcal{S}_0$ is the empty system {}; and
- every $\mathcal{S}_{i+1}$, where $0 < i < n$, is obtained from $\mathcal{S}_i$ by an application of one of the transformations.

The integer n is called the *length* of the SLG_{CN} derivation. If no transformation is applicable to $\mathcal{S}_n$, $\mathcal{S}_n$ is called a *final system* of Q.

SLG_{CN} *resolution* is the process of constructing an SLG_{CN} derivation for a function-free query atom Q with respect to a function-free program P under a computation rule R.

6. SOUNDNESS AND COMPLETENESS OF SLG_{CN} RESOLUTION

Given a function-free program P and a function-free query atom A, SLG_{CN} resolution transforms A :- A into a set of answers. The set of answers for A provides a more direct representation of true and false instances of A with respect to the declarative semantics of P. This section shows that SLG_{CN} resolution terminates for all function-free programs and queries, preserves all three-valued stable models, and computes the well-founded semantics.

6.1. Termination and Data Complexity

We establish the first termination result for goal-oriented query evaluation with constructive negation for function-free programs. The proof is in the appendix.

Theorem 6.1 (Termination). Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Then: (1) there exists an integer n such that the length of every SLG_{CN} derivation for Q is bounded by n; and (2) every final system for Q is either completed or involves recursion through non-ground negative literals.

For function-free programs, van Gelder *et el* [30] has shown that computing the well founded semantics has a polynomial time data complexity. The notion of *data complexity*, as defined by Vardi [31], is the complexity of evaluating a database query when the query is fixed and the database is regarded as input. In [7], we have shown that SLG resolution has a polynomial time data complexity for query evaluation with negation as failure under the well-founded semantics. With constructive negation, the number of distinct constraint answers may be exponential.

Example 6.1. Consider the following program:

$succ(a_1, a_2)$. ... $succ(a_{n-1}, a_n)$.
$max(a_n)$.
$p(X, Z)$:- $q(X, Z), succ(X, Y), p(Y, Z)$.
$p(X, Z)$:- $max(X), q(X, Z)$.

$q(X, Z)$:- $\sim r(X, W, Z)$.
$q(X, Z)$:- $\sim s(X, W, Z)$.
$r(X, X, a)$.
$s(X, X, b)$.

For subgoal $p(0, Z)$, its constraints are all the disjuncts in the disjunctive normal form of

$$\bigwedge_{1 \le i \le n} (r(a_i, W_i, Z) \notin r(X_i, X_i, a) \vee s(a_i, W_i, Z) \notin s(X_i, X_i, b))$$

The number of such constraint answers is 2^n.

It remains open whether some more compact representation of constraints can be used to preserve the polynomial time data complexity of query evaluation with constructive negation under the well-founded semantics. A similar example can be constructed without negation but with constraints. Thus the more general problem is whether polynomial data complexity can be achieved for query evaluation of function-free constraint logic programs.

6.2. Relating Partial Answers of Subgoals to a Program

Given a function-free program P and a function-free query atom Q, the X-rules of a subgoal $\mathcal{A}$ represent partial answers of $\mathcal{A}$ with respect to P. The constrained atom in the head of an X-rule captures the variable bindings and constraints; the delayed literals in the body of an X-rule are partially solved since their variable bindings and constraints have been propagated; and the remaining literals in the body of an X-rule are yet to be solved with respect to P.

To relate X-rules of subgoals in a system $\mathcal{S}$ to a program P, we defined a program $P(\mathcal{S})$ associated with $\mathcal{S}$ (in Section 2).

Definition 6.1. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}$ be a system in an SLG_{CN} derivation for Q. We associate with $\mathcal{S}$ a set of ground literals $I(\mathcal{S})$ as follows:

(a) if the Herbrand instantiation of $P(\mathcal{S})$ has a fact $B^{\mathcal{A}}_{\mathcal{H}}$, then $B \in I(\mathcal{S})$ and $B^{\mathcal{A}}_{\mathcal{H}} \in I(\mathcal{S})$;

(b) if a subgoal $\mathcal{A}$ in $\mathcal{S}$ is completed and $\mathcal{H}$ is a constrained atom that is subsumed by $\mathcal{A}$ and B is an atom in $|\mathcal{H}|$, then $\sim B^{\mathcal{A}}_{\mathcal{H}} \in I(\mathcal{S})$ if the Herbrand instantiation of $P(\mathcal{S})$ does not contain any rule with $B^{\mathcal{A}}_{\mathcal{H}}$ in the head;

(c) if a subgoal $\mathcal{A}$ in $\mathcal{S}$ is completed and B is in $|\mathcal{A}|$ and the Herbrand instantiation of $P(\mathcal{S})$ does not contain any rule with $B^{\mathcal{A}}_{\mathcal{H}}$ in the head for any $\mathcal{H}$, then $\sim B \in I(\mathcal{S})$ and $\sim B^{\mathcal{A}}_{\mathcal{H}'} \in I(\mathcal{S})$ for every constrained atom $\mathcal{H}'$ subsumed by $\mathcal{A}$ such that $B \in |\mathcal{H}'|$.

The difference between (b) and (c) in Definition 1 is as follows. In (b), even though the Herbrand instantiation of $P(\mathcal{S})$ does not contain any rule with $B^{\mathcal{A}}_{\mathcal{H}}$ in the head, it may still contain some rule (or even fact) with $B^{\mathcal{A}}_{\mathcal{H}'}$ for some constrained atom $\mathcal{H}'$ subsumed by $\mathcal{A}$. This is not possible in (c).

Lemma 6.1. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}_0, ..., \mathcal{S}_n$ be an arbitrary SLG_{CN} derivation for Q. Then $I(\mathcal{S}_0) \subseteq ... \subseteq I(\mathcal{S}_n)$.

PROOF. The lemma follows from two observations. One is that answers of a subgoal that are facts are never deleted by any transformation. The other is that when a subgoal $\mathcal{A}$ is completed, no new answers can be added although existing answers can be simplified. □

Let P be a function-free program and $\mathcal{S}$ be a system in an SLG_{CN} derivation for a function-free query atom Q. To relate the semantics of $P(\mathcal{S})$ to P, we look at the least partial model $LPM(\frac{P \cup P(\mathcal{S})}{J})$, where J is an interpretation of $P \cup P(\mathcal{S})$ and $\frac{P \cup P(\mathcal{S})}{J}$ is the quotient of $P \cup P(\mathcal{S})$ modulo J. Notice that

$$\frac{P \cup P(\mathcal{S})}{J} = \frac{P}{J} \cup \frac{P(\mathcal{S})}{J}$$

J has to satisfy certain symmetric requirements in order for the comparison between $\frac{P}{J}$ and $\frac{P(\mathcal{S})}{J}$ to be meaningful since $P(\mathcal{S})$ is essentially derived from P by solving literals in rule bodies.

Definition 6.2. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}$ be a system in an SLG_{CN} derivation for Q. Let J be an interpretation of $P \cup P(\mathcal{S})$. J is *symmetric on a subgoal* $\mathcal{A}$ in $\mathcal{S}$ if for every atom $B \in |\mathcal{A}|$,

- $J(B) = \mathbf{t}$ if and only if $J(B^{\mathcal{A}}_{\mathcal{H}}) = \mathbf{t}$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$ such that $B \in |\mathcal{H}|$; and
- $J(B) = \mathbf{f}$ if and only if $J(B^{\mathcal{A}}_{\mathcal{H}}) = \mathbf{f}$ for every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$ such that $B \in |\mathcal{H}|$.

J is a *symmetric interpretation* of $P \cup P(\mathcal{S})$ if J is symmetric on every subgoal in $\mathcal{S}$. $\mathcal{S}$ is a *symmetric system* if for every symmetric interpretation J of $P \cup P(\mathcal{S})$ such that $I(\mathcal{S}) \subseteq J$, $LPM(\frac{P \cup P(\mathcal{S})}{J})$ is symmetric.

6.3. Preservation of Three-Valued Stable Models

The following key theorems show that every system in an SLG_{CN} derivation for a function-free query atom Q is a symmetric system and that three-valued stable models are preserved. Their proofs are in the appendix.

Theorem 6.2. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_\alpha$ be an arbitrary SLG_{CN} derivation for Q, where α is an integer. Then for every $i (0 \leq i \leq \alpha)$, $I(\mathcal{S}_i) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_i))$ and $\mathcal{S}_i$ is a symmetric system.

Theorem 6.3. Let P be a function-free program, R be an arbitrary computation rule, and Q be a query atom, and $\mathcal{S}$ be a final system for Q that is completed. Then:

(a) for every $I \in \mathcal{ST}3(P)$, there exists a symmetric interpretation M of $P \cup P(\mathcal{S})$ such that $M|_P = I$ and $M|_{P(\mathcal{S})} \in \mathcal{ST}3(P(\mathcal{S}))$; and (b) for every $I \in \mathcal{ST}3(P(\mathcal{S}))$, there exists a symmetric interpretation M of $P \cup P(\mathcal{S})$ such that $M|_{P(\mathcal{S})} = I$ and $M|_P \in \mathcal{ST}3(P)$.

6.4. Computation of the Well-Founded Semantics

The primary purpose of SLG_{CN} resolution is to compute answers of a query with respect to the well-founded partial model of a function-free program. Let $\mathcal{S}$ be a final and completed system that is derived for a function-free query atom with respect to a function-free program P. We show that $\mathcal{WF}(P)$ coincides with $\mathcal{WF}(P(\mathcal{S}))$ as far as ground instances of subgoals in $\mathcal{S}$ are concerned. Moreover, for every ground atom $B \in |\mathcal{A}|$ of a subgoal A in $\mathcal{S}$, B is true in $\mathcal{WF}(P)$ if and only if $B \in |\mathcal{H}|$ for the head $\mathcal{H}$ of some answer of $\mathcal{A}$ that has an empty body, and B is false in $\mathcal{WF}(P)$ if and only if $B \notin |\mathcal{H}|$ for the head of any answer of $\mathcal{A}$. In other words, the truth values of ground instances of subgoals relevant to a query can be determined directly from the answers in $\mathcal{S}$, without any further derivation.

Theorem 6.4. Let P be a function-free program, R be an arbitrary computation rule, and Q be a query atom, and $\mathcal{S}$ be a final system for Q that is completed. Then there exists a symmetric interpretation J of $P \cup P(\mathcal{S})$ such that $J|_P = \mathcal{WF}(P)$ and $J|_{P(\mathcal{S})} = \mathcal{WF}(P(\mathcal{S}))$.

PROOF. By Theorem 2.2, $\mathcal{WF}(P) \in \mathcal{ST}3(P)$. By Theorem 6.3, there exists a symmetric interpretation M of $P \cup P(\mathcal{S})$ such that $M|_P = \mathcal{WF}(P)$ and $M|_{P(\mathcal{S})} \in \mathcal{ST}3(P(\mathcal{S}))$. By Theorem 2.2, $\mathcal{WF}(P(\mathcal{S})) \subseteq M|_{P(\mathcal{S})}$. Therefore for every subgoal $\mathcal{A}$ in $\mathcal{S}$ and for every $B \in |\mathcal{A}|$,

- if $B^{\mathcal{A}}_{\mathcal{H}} \in \mathcal{WF}(P(\mathcal{S}))$ for some constrained atom H subsumed by $\mathcal{A}$, then $B^{\mathcal{A}}_{\mathcal{H}} \in M|_{P(\mathcal{S})}$. Since M is symmetric, $B \in M|_P = \mathcal{WF}(P)$; and
- if $\sim B^{\mathcal{A}}_{\mathcal{H}} \in \mathcal{WF}(P(\mathcal{S}))$ for every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, then $\sim B^{\mathcal{A}}_{\mathcal{H}} \in M|_{P(\mathcal{S})}$ for every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$. Since M is symmetric, $\sim B \in M|_P = \mathcal{WF}(P)$.

For the other direction, $\mathcal{WF}(P(\mathcal{S})) \in \mathcal{ST}3(P(\mathcal{S}))$. By Theorem 6.3, there exists a symmetric interpretation M of $P \cup P(\mathcal{S})$ such that $M|_{P(\mathcal{S})} = \mathcal{WF}(P(\mathcal{S}))$ and $M|_P \in \mathcal{ST}3(P)$. By Theorem 2.2, $\mathcal{WF}(P) \subseteq M|_P$. Therefore for every subgoal A in $\mathcal{S}$ and for every $B \in |\mathcal{A}|$,

- if $B \in \mathcal{WF}(P)$, then $B \in M|_P$. Since M is symmetric, there exists a constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$ such that $B^{\mathcal{A}}_{\mathcal{H}} \in M|_{P(\mathcal{S})} = \mathcal{WF}(P(\mathcal{S}))$;
- if $\sim B \in \mathcal{WF}(P)$, then $\sim B \in M|_P$. Since M is symmetric, for every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, $\sim B^{\mathcal{A}}_{\mathcal{H}} \in M|_{P(\mathcal{S})} = \mathcal{WF}(P(\mathcal{S}))$.

Let J be an interpretation of $P \cup P(\mathcal{S})$ such that $J|_P = \mathcal{WF}(P)$ and $J|_{P(\mathcal{S})} = \mathcal{WF}(P(\mathcal{S}))$. Then J is a symmetric interpretation by the arguments above. □

Theorem 6.4 says only that the set of answers in a final system $\mathcal{S}$ preserves the well-founded partial model as a whole as far as instances of subgoals relevant to a query are concerned. The following theorem establishes further that the truth

values of ground instances of subgoals in the well-founded partial model of the original program can be determined by simply looking at the heads of answers in $\mathcal{S}$, without any further derivation.

Theorem 6.5. Let P be a function-free program, R be an arbitrary computation rule, and Q be a query atom, and $\mathcal{S}$ be a final system for Q that is completed. Then for every subgoal A in $\mathcal{S}$ and for every $B \in |\mathcal{A}|$, (a) $B \in \mathcal{WF}(P)$ if and only if $B \in |\mathcal{H}|$ for the head $\mathcal{H}$ of some answer of $\mathcal{A}$ in $\mathcal{S}$ that has an empty body; and (b) $\sim B \in \mathcal{WF}(P)$ if and only if $B \notin |\mathcal{H}|$ for the head $\mathcal{H}$ of any answer of $\mathcal{A}$ in $\mathcal{S}$.

PROOF. Let I be the interpretation of $P(\mathcal{S})$ such that $I = I(\mathcal{S})|_{P(\mathcal{S})}$. By Theorem 6.4, it suffices to prove that $I = \mathcal{WF}(P(\mathcal{S}))$. $I \subseteq \mathcal{WF}(P(\mathcal{S}))$ by Theorem 6.2.

For the other direction, it suffices to show that $I \in \mathcal{ST}3(P(\mathcal{S}))$, i.e., $I = LPM(\frac{P(\mathcal{S})}{I})$. Since $\mathcal{S}$ is a final system, no transformation can be applied. For every negative literal $\sim B^B_B$ that occurs in $P(\mathcal{S})$, since $\sim B^B$ cannot be simplified using SIMPLIFICATION, $I(B^B_B) = \mathbf{u}$. Since ANSWER COMPLETION cannot be applied to $\mathcal{S}$, for every subgoal $\mathcal{A}$ in $\mathcal{S}$ and for every constrained atom $\mathcal{H}$ in the head of some answer of $\mathcal{A}$, $\mathcal{H}$ is supported by $\mathcal{A}$. After every occurrence of $\sim B^B_B$ is replaced with $\mathbf{u}$, the program $P(\mathcal{S})$ is stratified and let J be the perfect model of the resulting stratified program. By a structural induction over the definition of $\mathcal{H}$ being supported by $\mathcal{A}$,

- $J(B^{\mathcal{A}}_{\mathcal{H}}) = \mathbf{t}$ for every $B \in |\mathcal{H}|$ if and only if $\mathcal{A}$ has an answer with $\mathcal{H}$ in the head and an empty body;
- $J(B^{\mathcal{A}}_{\mathcal{H}}) = \mathbf{u}$ for every $B \in |\mathcal{H}|$ if and only if $\mathcal{A}$ has some answers that have $\mathcal{H}$ in the head and all answers of $\mathcal{A}$ that have $\mathcal{H}$ in the head have some delayed literals.

Therefore $I = J$ and $I \in \mathcal{ST}3(P(\mathcal{S}))$. By Theorem 2.2, $\mathcal{WF}(P(\mathcal{S})) \subseteq I$. □

7. FUTURE WORK

We have presented SLG_{CN} resolution for effective query evaluation of function-free programs with constructive negation under the well-founded semantics. Termination is guaranteed due to the reduction of constraint answers to a normal form, redundant answer elimination and tabled evaluation.

Like SLG resolution [7], SLG_{CN} resolution is formalized directly in such a way that repeated computation is avoided in a subgoal. This is achieved by delaying only ground negative literals. It remains a challenge to extend SLG_{CN} resolution to general cases where non-ground negative literals may have to be delayed, while avoiding any repetition of computation in a subgoal. Delaying non-ground negative literals directly means that constructive negation can be applied even if the constraints and bindings of variables in a negative literal are not completely determined. This achieves the same effect that is realized by the notion of frontiers in [11, 12] and the TU-forests in [10]. The open problem is how to control the delaying of non-ground negative literals and the iterated propagation of constraints so that repetition of computation is avoided.

Several minor aspects of SLG_{CN} resolution can be refined. In all anti-subsumption constraints, we have $A \not\in B$, where A and B are atoms of the same predicate. Clearly the predicate in A and B is irrelevant and may be even confusing for answer reduction and redundant answer elimination. For example, $p(X,Y) \not\in p(a,b)$ is equivalent to $q(X,Y) \not\in q(a,b)$. A simple solution is to replace the predicate with a standard tuple constructor of the same arity.

For simplicity of formalization, we have considered only variant checking for redundant answers. Obviously just like repeated subgoals, one may also use AS-subsumption for redundant answers, although the resulting formalization may be more complicated due to the way X-rules in a system are related to a program in the correctness for SLG_{CN} resolution.

ACKNOWLEDGMENT

The authors thank David S. Warren, Luis M. Pereira, Terrance Swift and Carlos Damásio for discussions on constructive negation and thank the anonymous referees for their careful reading of the paper and comments.

REFERENCES

1. R. Bol and L. Degerstedt. Tabulated resolution for well founded semantics. In *Intl. Logic Programming Symposium*, October 1993.

2. G. Bossu and P. Siegel. Saturation, nonmonotonic reasoning and the closed world assumption. *Artificial Intelligence*, 25(1):13–63, 1985.

3. W.L. Buntine and H.-J. Bürckert. On solving equations and disequations. *Journal of ACM*, 41(4):591–629, July 1994.

4. D. Chan. Constructive negation based on the completed database. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proc. 5th Int. Conf. and Symp. on Logic Programming*, pages 111–125, 1988.

5. D. Chan. An extension of constructive negation and its application in coroutining. In *Proc. North American Conference on Logic Programming*, October 1989.

6. W. Chen, T. Swift, and D.S. Warren. Efficient top-down computation of queries under the well-founded semantics. *Journal of Logic Programming*, 24(3):161–199, September 1995.

7. W. Chen and D.S. Warren. Tabled evaluation with delaying for general logic programs. *Journal of ACM*, 43(1):20–74, January 1996.

8. K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum, New York, 1978.

9. H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.

10. C. Damásio. *Paraconsistent Extended Logic Programming with Constraints*. PhD thesis, Dept. de Informática, Universidade Nova de Lisboa, 1996.

11. W. Drabent. What is failure? an approach to constructive negation. *Acta Informatica*, 32(1):27–59, 1995.

12. F. Fages. Constructive negation by pruning. *Journal of Logic Programming*, to appear.

13. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R.A. Kowalski and K.A. Bowen, editors, *Joint Intl. Conference and Symposium on Logic Programming*, pages 1070–1080, 1988.

14. M. Johnson. A negation meta interpreter using anti-subsumption constraints. posted to comp.lang.prolog, 1992.

15. T. Khabaza. Negation as failure and parallelism. In *IEEE Symposium on Logic Programming*, pages 70–75, March 1984.

16. J.-L. Lassez and K.G. Marriott. Explicit representation of terms defined by counter examples. *Journal of Automated Reasoning*, 3(3):1–17, September 1987.

17. J. Małuszyński and T. Näslund. Fail substitutions for negation as failure. In E.L. Lusk and R. A. Overbeek, editors, *North American Conference on Logic Programming*, pages 461–476, 1989.

18. A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.

19. J.F. Naughton and R. Ramakrishnan. Bottom-up evaluation of logic programs. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic: Essays in honor of Alan Robinson*, pages 640–700. MIT Press, 1991.

20. T.C. Przymusinski. Every logic program has a natural stratification and an iterated least fixed point model. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 11–21, 1989.

21. T.C. Przymusinski. On constructive negation in logic programming. In *North American Conference on Logic Programming*, October 1989.

22. T.C. Przymusinski. On the declarative and procedural semantics of logic programs. *Journal of Automated Reasoning*, 5:167–205, 1989.

23. T.C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13:445–463, 1990.

24. T.C. Przymusinski and D.S. Warren. Well founded semantics: Theory and implementation. Draft, 1990.

25. I. V. Ramakrishnan, Prasad Rao, Konstantinos Sagonas, Terrance Swift, and David S. Warren. Efficient tabling mechanisms for logic programs. In Leon Sterling, editor, *Intl. Conference on Logic Programming*, pages 697–711. MIT Press, 1995.

26. K.A. Ross. A procedural semantics for well founded negation in logic programs. *Journal of Logic Programming*, 13(1):1–22, 1992.

27. K. Sagonas, T. Swift, and D.S. Warren. XSB as an efficient deductive database engine. In *ACM SIGMOD Conference on Management of Data*, pages 442–453, 1994.

28. P. Stuckey and S. Sudarshan. Well-founded ordered search. In *Proceedings of the 13th Conference on Foundations of Software Technology and Theoretical Computer Science*, 1993. LNCS 761.

29. P.J. Stuckey. Constructive negation in constraint logic programming. In *Proceedings of the 6th IEEE Annual Symposium on Logic in Computer Science*, pages 328–339, 1991.

30. A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620–650, July 1991.

31. M. Vardi. The complexity of relational query languages. In *ACM Symposium on Theory of Computing*, pages 137–146, May 1982.

32. M. Wallace. Negation by constraints: A sound and efficient implementation of negation in deductive databases. In *IEEE Symposium on Logic Programming*, pages 253–263, 1987.

33. D.S. Warren. *The XOLDT System.* SUNY at Stony Brook, 1992.

A. PROOFS OF THEOREMS 3.2, 6.1, 6.2 AND 6.3

Proof of Theorem 3.2: The theorem claims that the extended unification algorithm applied to $\forall\overline{U}.\exists\overline{V}.E$, where E is a finite set of equations, returns failure if and only if $\forall\overline{U}.\exists\overline{V}.E$ is unsatisfiable. Otherwise, it returns a finite set of equations E^* in solved form such that

$$\forall\overline{U}.\exists\overline{V}.E \leftrightarrow \forall\overline{U}.\exists\overline{V}.E^*$$

The proof is similar to that in [18]. We consider those cases that are specific to the modified unification algorithm.

Let $X = Y$ be an equation that is selected, where X and Y are distinct variables. In (b), either both X and Y are universally quantified, or one of X and Y is universally quantified and the other is a free variable. In both cases, the quantified set of equations is unsatisfiable.

Consider the case of $X = t$, where X is a variable and t is not a variable. In (d), if X is universally quantified, the quantified set of equations is unsatisfiable. In (f.1), X is a free variable and t contains a universally quantified variable and the quantified set of equations is not satisfiable. In (f.2), X is a free variable and t contains some variables in $\overline{V}$ that are existentially quantified. To avoid binding X to any term containing variables in $\overline{V}$, we introduce new distinct variables $Z_1, ..., Z_n$ if t is of the form $f(t_1, ..., t_n)$. The equation is replaced by $X = f(Z_1, ..., Z_n)$ and $Z_1 = t_1, ..., Z_n = t_n$.

Each transformation preserves all solutions. The process of transformations terminates for any finite set of equations. In addition, when it terminates without failure, the resulting set of equations is in solved form. □

Proof of Theorem 6.1: Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Theorem 6.1 claims that

(1) there exists an integer n such that the length of every SLG_{CN} derivation for Q is bounded by n; and

(2) every final system for Q is either completed or involves recursion through non-ground negative literals.

PROOF. (1). Since both P and Q are function-free, all atoms that occur in a system in an SLG_{CN} derivation are function-free. The number of atoms that are not variants of each other is finite. The number of reduced constrained atoms that are not variants of each other, denoted by $\mathcal{N}$, is finite, even though it may be exponential in the number of constant symbols in P and Q.

The number of subgoals in a system $\mathcal{S}$ is bounded by $\mathcal{N}$. For each subgoal, the number of initial X-rules introduced by NEW SUBGOAL is bounded by the number of rules in P, which is finite. The number of X-rules that can be generated directly from an X-rule G is

- bounded by some number dependent upon $\mathcal{N}$ in the case of CONSTRUCTIVE NEGATION,
- bounded by $\mathcal{N}$ in the case of POSITIVE RETURN, and
- 1 in the cases of DELAYING and SIMPLIFICATION.

Each of the resulting X-rules that is generated directly from G either

- has the same number of delayed literals as G and has one literal less than G that is not delayed; or
- has the same number of literals that are not delayed as G and has one delayed literal less than G.

The number of literals in the body of each X-rule is bounded by the maximum number of literals in a rule body in P. This is because delayed literals in the body of an answer are never propagated by POSITIVE RETURN or CONSTRUCTIVE NEGATION.

Finally COMPLETION and ANSWER COMPLETION only delete X-rules. Thus there exists some integer n such that the length of each SLG_{CN} derivation for Q is bounded by n.

(2). Let $\mathcal{S}$ be a final system for Q. By definition, no transformation is applicable to $\mathcal{S}$. Suppose that $\mathcal{S}$ is not completed. Then there must be at least one subgoal $\mathcal{A}$ that has an X-rule with a selected negative literal $\sim B$ such that the subgoal B is not ground and is not completed. Otherwise, either DELAYING or CONSTRUCTIVE NEGATION is applicable. We say that there is a *non-ground negative edge from $\mathcal{A}$ to B* in $\mathcal{S}$. Consider the graph that is composed of all the non-ground negative edges in $\mathcal{S}$. If the graph is acyclic, then some transformation must be applicable to subgoals (that are not completed) in the graph with no out-going non-ground edges, a contradiction with the assumption that $\mathcal{S}$ is a final system. If the graph is cyclic, then the cycles through negation involve only non-ground negative literals, which cannot be broken by DELAYING. □

Before proving Theorem 6.2 and Theorem 6.3, we establish several lemmas that relate the program associated with a system $\mathcal{S}$ to an original program P.

Lemma A.1. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}$ be a system in an arbitrary SLG_{CN} derivation for Q. Then for every symmetric interpretation J of $P \cup P(\mathcal{S})$ such

that $I(\mathcal{S}) \subseteq J$, (a) J satisfies Equivalence (3) in CONSTRUCTIVE NEGATION*; and (b) if a new system $\mathcal{S}'$ is derived from $\mathcal{S}$ by replacing an X-rule G of a subgoal $\mathcal{A}$ with X-rules $G_1, ..., G_k$ using* CONSTRUCTIVE NEGATION*, then for every rule in $\frac{P(\mathcal{S})}{J}$ of the form*

$$B^{\mathcal{A}}_{\mathcal{H}} \text{ :- } \varsigma$$

where $\mathcal{H}$ is the constrained atom in the head of G and $B \in |\mathcal{H}|$, there exists a rule in $\frac{P(\mathcal{S}')}{J}$ of the form

$$B^{\mathcal{A}}_{\mathcal{H}_i} \text{ :- } \varsigma$$

for some $i (1 \leq i \leq k)$ where $\mathcal{H}_i$ is the constrained atom in the head of G_i and $B \in |\mathcal{H}_i|$, and vice versa.

PROOF. Since (b) follows from (a), it is sufficient to prove (a). Let θ be an arbitrary ground substitution that is applied to both sides of the equivalence (1). For the direction of $\rightarrow$, suppose that $\sim A'\theta \in J$. Then $A'\theta \notin |\mathcal{H}_i|$ for every $i (1 \leq i \leq m)$ since J is an interpretation and $I(\mathcal{S}) \subseteq J$. Thus the first conjunct on the right side of (3) holds. If $A'\theta \notin |\mathcal{H}_i|$ for every $i (m+1 \leq i \leq n)$, then the second conjunct is also true. Otherwise, for every $i (m+1 \leq i \leq n)$ such that $A'\theta \in |\mathcal{H}_i|$, $\sim (A'\theta)^{A'}_{\mathcal{H}_i} \in J$ by the symmetry of J and so the second conjunct still holds.

Suppose that the right side of (3) is false. Then either the first conjunct is false or the second conjunct is false. If the first conjunct is false, then $A'\theta \in |\mathcal{H}_i|$ for some $i (1 \leq i \leq m)$. Thus $A'\theta \in I(\mathcal{S}) \subseteq J$ and so $\sim A'\theta$ is false in J. If the second conjunct is false, then $A'\theta \in |\mathcal{H}_i|$ for some $i (m+1 \leq i \leq n)$ such that $(A'\theta)^{A'}_{\mathcal{H}_i}$ is true. The symmetry of J implies $A'\theta \in J$ and so $\sim A'\theta$ is false in J.

The other direction, $\leftarrow$, is similar and the details are omitted. □

Lemma A.2. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_\alpha$ be an arbitrary SLG_{CN} derivation for Q, where α is an integer. Then for every $i (0 \leq i \leq \alpha)$ and for every symmetric interpretation J of $P \cup P(\mathcal{S}_i)$ such that $\cup_{0 \leq j < i} I(\mathcal{S}_j) \subseteq J$, and for every subgoal $\mathcal{A}$ in $\mathcal{S}_i$ that is not completed and for every $B \in |\mathcal{A}|$, if B :- ξ is in $\frac{P}{J}$, then $B^{\mathcal{A}}_{\mathcal{H}}$:- ξ is in $\frac{P(\mathcal{S}_i)}{J}$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$ such that $B \in |\mathcal{H}|$.

PROOF. The proof is based upon an induction on i. The basis case, $i = 0$, is trivial since $\mathcal{S}_0$ is empty.

Let i be an integer $\beta + 1$. Then $\mathcal{S}_i$ is obtained from $\mathcal{S}_\beta$ by one of the transformations. The cases of COMPLETION, SIMPLIFICATION, and ANSWER COMPLETION are trivial since they affect only subgoals that are completed in $\mathcal{S}_i$. The case of NEW SUBGOAL follows from the use of most general unifier in X-resolution in deriving the initial X-rules of a new subgoal. The case of POSITIVE RETURN follows from the inductive hypothesis since it only adds another X-rule to a subgoal that is not completed. The case of DELAYING follows directly from the symmetry of J. The case of CONSTRUCTIVE NEGATION follows directly from Lemma A.1 and the inductive hypothesis. This concludes the inductive proof of the lemma. □

Lemma A.3. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_\alpha$ be an arbitrary SLG_{CN} derivation for Q, where α is an integer. Then for every $i (0 \leq i \leq \alpha)$ and for

every symmetric interpretation J of $P \cup P(\mathcal{S}_i)$ such that $\cup_{0 \le j < i} I(\mathcal{S}_j) \subseteq J$, and for every subgoal $\mathcal{A}$ in $\mathcal{S}_i$ that is not completed and for every $B \in |\mathcal{A}|$, if $B^{\mathcal{A}}_{\mathcal{H}}$:- ς is in $\frac{P(\mathcal{S}_i)}{J}$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, then B :- ξ is in $\frac{P}{J}$ such that

$$Undelay(\varsigma) \subseteq \xi \text{ and } \xi - Undelay(\varsigma) \subseteq \cup_{0 \le j < i} I(\mathcal{S}_j)$$

where $Undelay(\varsigma) = \{B' \mid B' \text{ or } (B')^{\mathcal{A}'}_{\mathcal{H}'} \text{ occurs in } \varsigma \text{ for some subgoal } \mathcal{A}' \text{ and some constrained atom } \mathcal{H}' \text{ subsumed by } \mathcal{A}'\}$, and ξ is viewed as a set of atoms.

PROOF. Intuitively SLG_{CN} derivation tries to solve literals in the body of an X-rule. If a literal cannot be solved completely when it is selected, it may be replaced with some delayed literal. This lemma essentially says that literals that are delayed or not solved at all come from some rule in the original program P, i.e., $Undelay(\varsigma) \subseteq \xi$, and the other literals, i.e., $\xi - Undelay(\varsigma)$, have been solved based upon answers of subgoals that have been computed, i.e., $\cup_{0 \le j < i} I(\mathcal{S}_j)$.

The proof is based upon an induction on i. The basis case, $i = 0$, is trivial since $\mathcal{S}_0$ is empty.

Let i be an integer $\beta + 1$. Then $\mathcal{S}_i$ is obtained from $\mathcal{S}_\beta$ by one of the transformations. The cases of COMPLETION, SIMPLIFICATION, and ANSWER COMPLETION are trivial since they affect only subgoals that are completed in $\mathcal{S}_i$. The case of NEW SUBGOAL follows from the use of the most general unifier in X-resolution in deriving the initial X-rules of a new subgoal. The case of DELAYING follows directly from the symmetry of J, and the case of CONSTRUCTIVE NEGATION follows from Lemma A.1.

Let $\mathcal{A}$ be a subgoal in $\mathcal{S}_\beta$ and G be an X-rule with a constrained atom $\mathcal{H}$ in the head and a selected atom in the body. Suppose that a new X-rule G' is added by POSITIVE RETURN with a constrained atom $\mathcal{H}'$ in the head when $\mathcal{S}_i$ is derived from $\mathcal{S}_\beta$. Suppose that $B^{\mathcal{A}}_{\mathcal{H}'}$:- ς' is a rule in $\frac{P(\mathcal{S}_i)}{J}$ that is obtained from a ground instance of $(G')^{\mathcal{A}}$. Then by the definition of POSITIVE RETURN, there exists a rule, $B^{\mathcal{A}}_{\mathcal{H}}$:- ς in $\frac{P(\mathcal{S}_\beta)}{J}$ that is obtained from a ground instance of $G^{\mathcal{A}}$ such that $Undelay(\varsigma') \subseteq Undelay(\varsigma)$ and $Undelay(\varsigma) - Undelay(\varsigma') \subseteq I(\mathcal{S}_\beta)$. By inductive hypothesis, there is a rule B :- ξ in $\frac{P}{J}$ such that $Undelay(\varsigma) \subseteq \xi$ and $\xi - Undelay(\varsigma) \subseteq \cup_{0 \le j < \beta} I(\mathcal{S}_j)$. Therefore $Undelay(\varsigma') \subseteq \xi$ and $\xi - Undelay(\varsigma') \subseteq \cup_{0 \le j < i} I(\mathcal{S}_j)$ and the lemma holds. □

Lemma A.4. Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_\alpha$ be an arbitrary SLG_{CN} derivation of Q, where α is an integer. Then for every $i (0 \le i \le \alpha)$ and for every symmetric interpretation J of $P \cup P(\mathcal{S}_i)$ such that $\cup_{0 \le j < i} I(\mathcal{S}_j) \subseteq J$, and for every subgoal $\mathcal{A}$ in $\mathcal{S}_i$ that is not completed and for every X-rule G of $\mathcal{A}$ in $\mathcal{S}_i$, of the form

$$\mathcal{H} \leftarrow Left, A_1, Right$$

where A_1 is the selected atom of G and for every constrained atom $\mathcal{H}_1$ in $\Sigma(G)$, there exists some X-rule G^* of $\mathcal{A}$ in $\mathcal{S}_i$ such that for every rule in $\frac{P(\mathcal{S}_i)}{J}$ of the form

$$B^{\mathcal{A}}_{\mathcal{H}} \text{ :- } \varsigma_{left}, B_1, \varsigma_{right}$$

obtained from a ground instance of $G^{\mathcal{A}}$ such that $B_1 \in |\mathcal{H}_1|$, there exists a rule

in $\frac{P(\mathcal{S}_i)}{J}$, *obtained from a ground instance of* $(G^*)^{\mathcal{A}}$, *of the form*

$$B^{\mathcal{A}}_{\mathcal{H}^*} \leftarrow \varsigma_{left}, \varsigma_{right}$$

or

$$B^{\mathcal{A}}_{\mathcal{H}^*} \leftarrow \varsigma_{left}, (B_1)^{\mathcal{A}_1}_{\mathcal{H}_1}, \varsigma_{right}$$

where $\mathcal{H}^*$ *is the constrained atom in the head of* G^* *and* $\mathcal{A}_1$ *is the subgoal for the selected atom of* G.

PROOF. The proof is based upon an induction on i. The basis case, $i = 0$, is trivial.

Let i be a successor ordinal $\beta + 1$. Then $\mathcal{S}_i$ is obtained from $\mathcal{S}_\beta$ by one of the transformations. For the case of NEW SUBGOAL, the annotation $\Sigma(G)$ of any initial X-rule G of a new subgoal is always empty and the lemma holds by inductive hypothesis. The cases of COMPLETION, SIMPLIFICATION, and ANSWER COMPLETION hold by inductive hypothesis, because they affect only X-rules of subgoals that are completed in $\mathcal{S}_i$. For DELAYING, the symmetry of J implies that $\frac{P(\mathcal{S}_i)}{J} = \frac{P(\mathcal{S}_\beta)}{J}$ and so the lemma holds by inductive hypothesis. The case of CONSTRUCTIVE NEGATION follows from Lemma A.1 and the inductive hypothesis.

For POSITIVE RETURN, let G be an active X-rule of $\mathcal{A}$ in $\mathcal{S}_\beta$, of the form

$$\mathcal{H} \leftarrow Left, A_1, Right$$

where A_1 is the selected atom. Let $\mathcal{A}_1$ be the subgoal of the selected atom A_1. For every $\mathcal{H}_1 \in \Sigma(G)$, there exists an answer C of $\mathcal{A}_1$ with $\mathcal{H}_1$ in the head in $\mathcal{S}_\beta$ such that POSITIVE RETURN is applied to G by using C when $\mathcal{S}_i$ is derived from $\mathcal{S}_\beta$. Let G' be the X-resolvent of G with C if C has an empty body, and be the X-factor of G with C if C has some delayed literals in its body. Then G' satisfies the properties of G^* as specified in the lemma, and the lemma holds by inductive hypothesis. □

Proof of Theorem 6.2: Let P be a function-free program, R be an arbitrary computation rule, and Q be a function-free query atom. Let $\mathcal{S}_0, \mathcal{S}_1, ..., \mathcal{S}_\alpha$ be an arbitrary SLG_{CN} derivation for Q, where α is an integer. Theorem 6.2 claims that for every $i (0 \leq i \leq \alpha)$, $I(\mathcal{S}_i) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_i))$ and $\mathcal{S}_i$ is a symmetric system.

The proof is based upon an induction on i. For the basis case, $i = 0$, $\mathcal{S}_0$ is the empty system and $I(\mathcal{S}_0)$ is the empty set and $P(\mathcal{S}_0)$ is the empty program. The lemma holds trivially.

For the inductive case, we prove the following:

(a) $LPM(\frac{P \cup P(\mathcal{S}_i)}{J})$ is symmetric for every symmetric interpretation J of $P \cup P(\mathcal{S}_i)$ such that $\cup_{0 \leq j < i} I(\mathcal{S}_j) \subseteq J$;

(b) $I(\mathcal{S}_i) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_i))$.

Let J be an arbitrary symmetric interpretation of $P \cup P(\mathcal{S}_i)$ such that $I(\mathcal{S}_i) \subseteq J$. Then $\cup_{0 \leq j < i} I(\mathcal{S}_j) \subseteq J$ by Lemma 6.1. Thus (a) implies that $\mathcal{S}_i$ is a symmetric system. We show that (a) implies (b) and then prove (a).

(a) $\Rightarrow$ (b). By inductive hypothesis, $I(\mathcal{S}_j) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_j))$ for every $j (0 \leq j < i)$. Since P is independent of $P(\mathcal{S}_j)$ for every $j (0 \leq j \leq i)$, $\cup_{0 \leq j < i} I(\mathcal{S}_j)|_P \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_i))$. By Lemma 6.1, $I(\mathcal{S}_j) \subseteq I(\mathcal{S}_i)$ for every $j (0 \leq j \leq i)$. By the definition of $I(\mathcal{S}_i)$, $I(\mathcal{S}_i)|_{P(\mathcal{S}_i)} \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_i))$. Thus

$$\cup_{0 \leq j < i} I(\mathcal{S}_j) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_i))$$

We construct an interpretation J of $P \cup P(\mathcal{S}_i)$ as follows:

- $J|_P = \mathcal{WF}(P)$;
- for every subgoal $\mathcal{A}$ in $\mathcal{S}_i$ and every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$ and every $B \in |\mathcal{H}|$,
  - if $B^{\mathcal{A}}_{\mathcal{H}} \in \cup_{0 \leq j < i} I(\mathcal{S}_j)$, then $B^{\mathcal{A}}_{\mathcal{H}} \in J$;
  - if $\sim B^{\mathcal{A}}_{\mathcal{H}} \in \cup_{0 \leq j < i} I(\mathcal{S}_j)$, then $\sim B^{\mathcal{A}}_{\mathcal{H}} \in J$;
  - otherwise, $J(B^{\mathcal{A}}_{\mathcal{H}}) = J(B)$

Clearly $\cup_{0 \leq j < i} I(\mathcal{S}_j) \subseteq J$ since $\cup_{0 \leq j < i} I(\mathcal{S}_j) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_i))$. It can also be verified that J is a symmetric interpretation of $P \cup P(\mathcal{S}_i)$.

Let $M = LPM(\frac{P \cup P(\mathcal{S}_i)}{J})$. Since $J|_P = \mathcal{WF}(P)$ and P is independent of $P(\mathcal{S}_i)$, $M|_P = \mathcal{WF}(P)$ as $\mathcal{WF}(P)$ is a three-valued stable model of P. By (a), M is symmetric. By the definition of $I(\mathcal{S}_i)$, every literal of the form $B^{\mathcal{A}}_{\mathcal{H}}$ or $\sim B^{\mathcal{A}}_{\mathcal{H}}$ in $I(\mathcal{S}_i)$ is in M and in $\mathcal{WF}(P \cup P(\mathcal{S}_i))$, where $\mathcal{A}$ is a subgoal in $\mathcal{S}_i$, $\mathcal{H}$ is a constrained atom subsumed by $\mathcal{A}$, and $B \in |\mathcal{H}|$. Since M is symmetric, $I(\mathcal{S}_i)|_P \subseteq M|_P = \mathcal{WF}(P)$. Thus $I(\mathcal{S}_i) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}_i))$.

Now that we have established that (a) implies (b), we prove (a). Let i be an integer β+1. Then $\mathcal{S}_i$ is obtained from $\mathcal{S}_\beta$ by one of the transformations. By Lemma 6.1, $I(\mathcal{S}_\beta) = \cup_{0 \leq j < i} I(\mathcal{S}_j)$. Let $M_\beta = LPM(\frac{P \cup P(\mathcal{S}_\beta)}{J})$ and $M_i = LPM(\frac{P \cup P(\mathcal{S}_i)}{J})$. By inductive hypothesis, M_β is a symmetric interpretation of $P \cup P(\mathcal{S}_\beta)$. We prove that M_i is symmetric by a case analysis of the transformations.

NEW SUBGOAL: Suppose that $\mathcal{A}$ is a new subgoal that is introduced and B is an arbitrary ground atom in $|\mathcal{A}|$. Then the Herbrand instantiation of P contains a rule of the form $B \leftarrow \phi$ if and only if the Herbrand instantiation of $P(\mathcal{S}_i)$ contains a rule of the form $B^{\mathcal{A}}_{\mathcal{H}} \leftarrow \phi$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$. Therefore M_i is symmetric on $\mathcal{A}$. Subgoals in $\mathcal{S}_\beta$ are not affected, and (a) holds by inductive hypothesis.

DELAYING: Since J is a symmetric interpretation of $P \cup P(\mathcal{S}_i)$ and $\cup_{0 \leq j < i} I(\mathcal{S}_j) \subseteq J$, it can be verified that $\frac{P(\mathcal{S}_\beta)}{J} = \frac{P(\mathcal{S}_i)}{J}$. Thus $M_i = M_\beta$ and (a) holds by inductive hypothesis.

SIMPLIFICATION: Let $\mathcal{A}$ be a subgoal that is completed in $\mathcal{S}_\beta$ and G be an answer of $\mathcal{A}$. Let L be a delayed literal in the body of G. If L is a negative delayed literal, it can be verified that $\frac{P(\mathcal{S}_\beta)}{J} = \frac{P(\mathcal{S}_i)}{J}$ based upon the assumption on J. Thus $M_i = M_\beta$ and (a) holds by inductive hypothesis.

If L is a positive delayed literal of the form $B^{\mathcal{A}_1}_{\mathcal{H}_1}$, where $\mathcal{A}_1$ is a subgoal in $\mathcal{S}_\beta$ and $\mathcal{H}_1$ is subsumed by $\mathcal{A}_1$, there are two cases. If L is successful, then $\mathcal{A}_1$ has an answer C in $\mathcal{S}_\beta$ that has $\mathcal{H}_1$ in the head and an empty body. Then L is deleted from the body of G. Clearly for every ground instance h of B, $h^{\mathcal{A}_1}_{\mathcal{H}_1}$ can always be derived using $C^{\mathcal{A}_1}$ in $P(\mathcal{S}_i)$. Thus $M_i = M_\beta$ and (a) holds by inductive hypothesis. The case that L is failed is similar.

ANSWER COMPLETION: Let U be the set of all pairs $(\mathcal{A}, \mathcal{H})$ in $\mathcal{S}_\beta$ such that $\mathcal{A}$ is a subgoal and $\mathcal{H}$ be the head of some answer of $\mathcal{A}$ such that $\mathcal{H}$ is not supported by $\mathcal{A}$. Then $\mathcal{S}_i$ is obtained from $\mathcal{S}_\beta$ by deleting all the answers of $\mathcal{A}$ that have $\mathcal{H}$ in the head, for some $(\mathcal{A}, \mathcal{H}) \in U$.

By definition, for every pair $(\mathcal{A}, \mathcal{H}) \in U$, $\mathcal{A}$ is completed and for every answer G of $\mathcal{A}$ that has $\mathcal{H}$ in the head, there exists a positive delayed literal in the body of G, of the form $(B_1)^{\mathcal{A}_1}_{\mathcal{H}_1}$, where $\mathcal{H}_1$ is not supported by $\mathcal{A}_1$, i.e., $(\mathcal{A}_1, \mathcal{H}_1) \in U$. Thus

for every pair $(\mathcal{A},\mathcal{H}) \in U$ and every atom B in $\mathcal{H}|$, $\sim B^{\mathcal{A}}_{\mathcal{H}} \in M_\beta$ and $\sim B^{\mathcal{A}}_{\mathcal{H}} \in M_i$. Hence $M_i = M_\beta$ and (a) holds by inductive hypothesis.

POSITIVE RETURN: First, POSITIVE RETURN does not affect subgoals that are completed in $\mathcal{S}_\beta$. In particular, for every completed subgoal $\mathcal{A}$ in $\mathcal{S}_\beta$ and every answer C of $\mathcal{A}$ that is not disposed in $\mathcal{S}_\beta$, and for every positive delayed literal in the body of C, of the form $B^{\mathcal{A}_1}_{\mathcal{H}_1}$, $\mathcal{A}_1$ is also completed. The reason is that a positive delayed literal is created by POSITIVE RETURN from an X-rule with a selected atom, and an X-rule of a subgoal with a selected atom is disposed only by COMPLETION. By inductive hypothesis, M_i remains symmetric on all completed subgoals in $\mathcal{S}_i$, which are precisely completed subgoals in $\mathcal{S}_\beta$.

Second, let $\mathcal{A}$ be an arbitrary subgoal in $\mathcal{S}_i$ that is not completed, and let B be an arbitrary atom in $|\mathcal{A}|$. By Lemma A.2, $B \in M_i$ implies $B^{\mathcal{A}}_{\mathcal{H}} \in M_i$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, and if $\sim B^{\mathcal{A}}_{\mathcal{H}} \in M_i$ for every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, then $\sim B \in M_i$.

For the other direction, let $P' = \frac{P \cup P(\mathcal{S}_i)}{J}$. Recall that $M_i = \tau^{\uparrow\omega}_{P'}$. We show by induction on k that for every $k \geq 0$, and for every subgoal $\mathcal{A}$ in $\mathcal{S}_i$ and every atom B in $|\mathcal{A}|$, if $B^{\mathcal{A}}_{\mathcal{H}} \in Und(\tau^{\uparrow k}_{P'})$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, then $B \in Und(M_i) \cup Pos(M_i)$, and if $B^{\mathcal{A}}_{\mathcal{H}} \in Pos(\tau^{\uparrow k}_{P'})$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, then $B \in Pos(M_i)$.

The basis case, $k = 0$, is trivial since $\tau^{\uparrow 0}_{P'} = \emptyset$, in which every ground atom is false. For the inductive case, $k+1$, consider any rule of the form $B^{\mathcal{A}}_{\mathcal{H}} \leftarrow \varsigma$ in P'. By Lemma A.3, there is a rule $B \leftarrow \xi$ in P' such that $Undelay(\varsigma) \subseteq \xi$ and $\xi - Undelay(\varsigma) \subseteq \cup_{0 \leq j < i} I(\mathcal{S}_j) = I(\mathcal{S}_\beta)$. The definition of $I(\mathcal{S}_\beta)$ implies that $I(\mathcal{S}_\beta)|_{P(\mathcal{S}_\beta)} \subseteq M_\beta$. Since M_β is symmetric by inductive hypothesis, every ground atom in $\xi - Undelay(\varsigma)$ is in M_β. As $M_\beta|_P = M_i|_P$, every ground atom in $\xi - Undelay(\varsigma)$ is in M_i too. If $B^{\mathcal{A}}_{\mathcal{H}} \in Und(\tau^{\uparrow k+1}_{P'})$ due to a rule $B^{\mathcal{A}}_{\mathcal{H}} \leftarrow \varsigma$, then B is in $Und(M_\beta) \cup Pos(M_i)$ due to the rule $B \leftarrow \xi$ by inductive hypothesis. Similarly, if $B^{\mathcal{A}}_{\mathcal{H}} \in Pos(\tau^{\uparrow k+1}_{P'})$, then $B \in Pos(M_i)$.

This concludes the induction on k. Thus for every subgoal $\mathcal{A}$ that is not completed in $\mathcal{S}_i$ and for every ground atom B in $|\mathcal{A}|$, if $B^{\mathcal{A}}_{\mathcal{H}} \in M_i$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$, then $B \in M_i$, and if $\sim B \in M_i$, then $\sim B^{\mathcal{A}}_{\mathcal{H}} \in M_i$ for every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$.

This concludes the proof that M_i is symmetric, and so (a) holds.

CONSTRUCTIVE NEGATION: Like POSITIVE RETURN, CONSTRUCTIVE NEGATION does not affect subgoals that are completed in $\mathcal{S}_\beta$. By inductive hypothesis, M_i remains symmetric on subgoals that are completed in $\mathcal{S}_\beta$, which remain completed in $\mathcal{S}_i$.

Notice that CONSTRUCTIVE NEGATION does not introduce any new subgoals. Consider a subgoal $\mathcal{A}$ in $\mathcal{S}_\beta$ that is not completed in $\mathcal{S}_\beta$ and an arbitrary ground atom $B \in |\mathcal{A}|$. There are two possibilities for $\mathcal{A}$. One is that $\mathcal{A}$ is not completed in $\mathcal{S}_i$ and the other is that $\mathcal{A}$ becomes completed in $\mathcal{S}_i$ after CONSTRUCTIVE NEGATION is applied to its only X-rule that has a selected (negative) literal. In the former case, the rest of the argument is the same as in the case for POSITIVE RETURN based upon Lemma A.2 and Lemma A.3. In the latter case, Lemma A.2 and Lemma A.3 are not applicable to $\mathcal{A}$ in $\mathcal{S}_i$ since it is completed in $\mathcal{S}_i$, but are applicable to $\mathcal{A}$ in $\mathcal{S}_\beta$ in which it is not completed. Together with Lemma A.1, the same argument in the case for POSITIVE RETURN goes through.

COMPLETION: Following the same argument as in POSITIVE RETURN, COMPLE-

TION does not affect subgoals that are completed in $\mathcal{S}_\beta$. In particular, M_i and M_β coincide on all literals of the form $B^{\mathcal{A}}_{\mathcal{H}}$ or $\sim B^{\mathcal{A}}_{\mathcal{H}}$, where $\mathcal{A}$ is a subgoal that is completed in $\mathcal{S}_\beta$. In addition, $M_\beta|_P = M_i|_P$. By inductive hypothesis, M_i remains symmetric on all subgoals that are completed in $\mathcal{S}_\beta$.

By definition, COMPLETION disposes all X-rules of some subgoals that are not answers, and so $P(\mathcal{S}_i)$ can be obtained from $P(\mathcal{S}_\beta)$ by deleting some rules. Therefore $M_i \preceq M_\beta$ (with respect to the truth ordering). Lemma A.2 together with $M_i \preceq M_\beta$ implies by inductive hypothesis that M_i is symmetric on all subgoals that are not completed in $\mathcal{S}_i$.

Let Λ be a non-empty set of subgoals that are completely evaluated in $\mathcal{S}_\beta$ such that all X-rules of subgoals in Λ that are not answers are disposed by COMPLETION. It remains to show that M_i is symmetric on subgoals in Λ. Let $P' = \frac{P \cup P(\mathcal{S}_i)}{J}$. Since $M_i \preceq M_\beta$ and $M_\beta|_P = M_i|_P$, it suffices to prove that for every $k \geq 0$, and for every subgoal $\mathcal{A} \in \Lambda$ and every ground atom B in $|\mathcal{A}|$,

(1) if $B \in Und(\tau^{\uparrow k}_{P'})$, then $B^{\mathcal{A}}_{\mathcal{H}} \in Und(\tau^{\uparrow k}_{P'}) \cup Pos(\tau^{\uparrow k}_{P'})$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$; and

(2) if $B \in Pos(\tau^{\uparrow k}_{P'})$, then $B^{\mathcal{A}}_{\mathcal{H}} \in Pos(\tau^{\uparrow k}_{P'})$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$.

The basis case, $k = 0$, is trivial since $\tau^{\uparrow 0}_{P'} = \emptyset$. For the inductive case, suppose that $B \in Und(\tau^{\uparrow k+1}_{P'}) \cup Pos(\tau^{\uparrow k+1}_{P'})$, and the derivation of B uses a rule of the form $B \leftarrow \xi \in P'$. However, $B \leftarrow \xi$ is also a rule in $\frac{P \cup P(\mathcal{S}_\beta)}{J}$. By Lemma A.2, $B^{\mathcal{A}}_{\mathcal{H}} \leftarrow \xi$ is a rule in $\frac{P \cup P(\mathcal{S}_\beta)}{J}$ for some constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$.

If $B^{\mathcal{A}}_{\mathcal{H}} \leftarrow \xi$ is a rule in P', then (1) and (2) hold by inductive hypothesis. Otherwise, since $\mathcal{A}$ is in Λ, either $\mathcal{A}$ succeeds, in which case (1) and (2) hold, or $\mathcal{A}$ has an X-rule G in $\mathcal{S}_\beta$ of the form

$$\mathcal{H} \leftarrow Left, A_1, Right$$

with a selected atom A_1, and $B^{\mathcal{A}}_{\mathcal{H}} \leftarrow \xi$ is obtained from a ground instance of $G^{\mathcal{A}}$ and is of the form

$$B^{\mathcal{A}}_{\mathcal{H}} \leftarrow \xi_{left}, B_1, \xi_{right}$$

where B_1 is a ground instance of A_1. By assumption on B, $B_1 \in Und(\tau^{\uparrow k}_{P'}) \cup Pos(\tau^{\uparrow k}_{P'})$. By inductive hypothesis on k, (1) and (2) hold for B_1 and $(B_1)^{\mathcal{A}_1}_{\mathcal{H}_1}$ for the subgoal $\mathcal{A}_1$ of the selected atom A_1 and some constrained atom $\mathcal{H}_1$ subsumed by $\mathcal{A}_1$. Since $\mathcal{A}$ is completed in $\mathcal{S}_i$, $\mathcal{H}_1$ must be in $\Sigma(G)$. By Lemma A.4, there exists some constrained atom $\mathcal{H}'$ subsumed by $\mathcal{A}$ such that

$$B^{\mathcal{A}}_{\mathcal{H}'} \leftarrow \xi_{left}, \xi_{right}$$

or

$$B^{\mathcal{A}}_{\mathcal{H}'} \leftarrow \xi_{left}, (B_1)^{\mathcal{A}_1}_{\mathcal{H}_1}, \xi_{right}$$

is a rule in $\frac{P(\mathcal{S}_\beta)}{J}$.

The number of positive literals in the body of an X-rule is bounded by the maximum number of literals in the body of a rule in P, which is finite. By repeatedly applying the same argument for $B^{\mathcal{A}}_{\mathcal{H}} \leftarrow \xi$ to $B^{\mathcal{A}}_{\mathcal{H}'} \leftarrow \xi_{left}, (B_1)^{\mathcal{A}_1}_{\mathcal{H}_1}, \xi_{right}$, we will

eventually obtain some $B^{\mathcal{A}}_{\mathcal{H}^*}$ for some constrained atom $\mathcal{H}^*$ subsumed by $\mathcal{A}$ such that (1) and (2) hold for B and $B^{\mathcal{A}}_{\mathcal{H}^*}$. This concludes the induction on k.

This concludes the proof for the case of COMPLETION. □

Proof of Theorem 6.3: Let P be a function-free program, R be an arbitrary computation rule, and Q be a query atom, and $\mathcal{S}$ be a final system for Q that is completed. Theorem 6.3 claims the following:

(a) for every $I \in \mathcal{ST}3(P)$, there exists a symmetric interpretation M of $P \cup P(\mathcal{S})$ such that $M|_P = I$ and $M|_{P(\mathcal{S})} \in \mathcal{ST}3(P(\mathcal{S}))$;

(b) for every $I \in \mathcal{ST}3(P(\mathcal{S}))$, there exists a symmetric interpretation M of $P \cup P(\mathcal{S})$ such that $M|_{P(\mathcal{S})} = I$ and $M|_P \in \mathcal{ST}3(P)$.

Since $\mathcal{S}$ is a final system that is completed, all X-rules of subgoals in $\mathcal{S}$ are answers. Thus $P(\mathcal{S})$ and P are independent of each other. By Theorem 6.2, $\mathcal{S}$ is a symmetric system.

For (a), let $I \in \mathcal{ST}3(P)$. By Theorem 2.2, $\mathcal{WF}(P) \subseteq I$. By Theorem 6.2, $I(\mathcal{S}) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}))$. Since P and $P(\mathcal{S})$ are independent of each other, $\mathcal{WF}(P \cup P(\mathcal{S})) = \mathcal{WF}(P) \cup \mathcal{WF}(P(\mathcal{S}))$. Thus $I(\mathcal{S})|_P \subseteq \mathcal{WF}(P) \subseteq I$. We construct an interpretation J_0 of $P \cup P(\mathcal{S})$ as follows:

- $J_0|_P = I$;
- for every subgoal $\mathcal{A}$ in $\mathcal{S}$ and every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$ and every $B \in |\mathcal{H}|$,
  - if $B^{\mathcal{A}}_{\mathcal{H}} \in I(\mathcal{S})$, then $B^{\mathcal{A}}_{\mathcal{H}} \in J_0$;
  - if $\sim B^{\mathcal{A}}_{\mathcal{H}} \in I(\mathcal{S})$, then $\sim B^{\mathcal{A}}_{\mathcal{H}} \in J_0$;
  - otherwise, $J(B^{\mathcal{A}}_{\mathcal{H}}) = J_0(B)$

The existence of J_0 is ensured by the fact that $I(\mathcal{S}) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}))$ and $\mathcal{WF}(P) \subseteq I$. It can also be verified that J is a symmetric interpretation.

Notice that $I(\mathcal{S}) \subseteq J_0$. Let $M_0 = LPM(\frac{P \cup P(\mathcal{S})}{J_0})$. By Theorem 6.2, $\mathcal{S}$ is a symmetric system and so M_0 is symmetric. Since P and $P(\mathcal{S})$ are independent of each other, $M_0|_P = LPM(\frac{P}{J_0|_P})$ and $M_0|_{P(\mathcal{S})} = LPM(\frac{P(\mathcal{S})}{J_0|_{P(\mathcal{S})}})$. As $J_0|_P = I \in \mathcal{ST}3(P)$, $M_0|_P = I$.

Both M_0 and J_0 are symmetric and $M_0|_P = J_0|_P = I$. Thus for every ground subgoal B in $\mathcal{S}$, $M_0(B^B_B) = J_0(B^B_B) = I(B)$. Since $\mathcal{S}$ is completed, all negative literals occurring in $P(\mathcal{S})$ are of the form $\sim B^B_B$, where B is a ground atom, or $\sim B^{\mathcal{A}}_{\mathcal{H}}$, where $\mathcal{A}$ is not a ground atom. The latter form of negative literals in $P(\mathcal{S})$ is derived from CONSTRUCTIVE NEGATION. Note that CONSTRUCTIVE NEGATION can be applied to a selected negative literal only if the positive counterpart is completed as a subgoal. Thus after $P(\mathcal{S})$ is simplified by replacing each occurrence of $\sim B^B_B$ with its truth value in J_0 (or M_0), $P(\mathcal{S})$ must be stratified, say with a stratification $P_1 \cup P_2 \cup ... \cup P_k$. Under such stratification, M_0 and $LPM(\frac{P(\mathcal{S})}{M|_{P(\mathcal{S})}})$ coincide on all ground atoms in the lowest stratum since $M_0(B^B_B) = J_0(B^B_B) = I(B)$ for every ground subgoal B in $\mathcal{S}$.

For each $i (1 \leq i \leq k)$, we construct interpretations J_i and M_i of $P \cup P(\mathcal{S})$ as follows:

- $J_i|_P = I$;

- for every subgoal $\mathcal{A}$ in $\mathcal{S}$ and every constrained atom $\mathcal{H}$ subsumed by $\mathcal{A}$ and every $B \in |\mathcal{H}|$,
  - if $B^{\mathcal{A}}_{\mathcal{H}} \in I(\mathcal{S})$, then $B^{\mathcal{A}}_{\mathcal{H}} \in J_i$;
  - if $\sim B^{\mathcal{A}}_{\mathcal{H}} \in I(\mathcal{S})$, then $\sim B^{\mathcal{A}}_{\mathcal{H}} \in J_i$;
  - if $B^{\mathcal{A}}_{\mathcal{H}}$ belongs to a stratum less than or equal to i, then $J_i(B^{\mathcal{A}}_{\mathcal{H}}) = M_{i-1}(B^{\mathcal{A}}_{\mathcal{H}})$
  - otherwise, $J_i(B^{\mathcal{A}}_{\mathcal{H}}) = J_i(B)$
- $M_i = LPM(\frac{P \cup P(\mathcal{S})}{J_i})$

The same arguments for J_0 and M_0 can be used to verify that both J_i and M_i are symmetric interpretations such that $J_i|_P = M_i|_P = I$ for every $i(1 \leq i \leq k)$. In addition, based upon the aforementioned stratification of $P(\mathcal{S})$, $M_k = LPM(\frac{P \cup P(\mathcal{S})}{M_k})$ and $M_k|_P = I$. Since P and $P(\mathcal{S})$ are independent of each other, $M_k|_{P(\mathcal{S})} \in \mathcal{ST}3(P(\mathcal{S}))$.

This concludes the proof for Part (a) of the lemma, and we now show that Part (b) of the lemma holds. Let $I \in \mathcal{ST}3(P(\mathcal{S}))$.

By Theorem 2.2, $\mathcal{WF}(P(\mathcal{S})) \subseteq I$. By Theorem 6.2, $I(\mathcal{S}) \subseteq \mathcal{WF}(P \cup P(\mathcal{S}))$. Since P and $P(\mathcal{S})$ are independent of each other, $\mathcal{WF}(P \cup P(\mathcal{S})) = \mathcal{WF}(P) \cup \mathcal{WF}(P(\mathcal{S}))$. Thus $I(\mathcal{S})|_{P(\mathcal{S})} \subseteq \mathcal{WF}(P(\mathcal{S})) \subseteq I$. As shown in Part (a), $\mathcal{WF}(P \cup P(\mathcal{S}))$ is a subset of some symmetric interpretation of $P \cup P(\mathcal{S})$. Thus there exists some symmetric interpretation J_0 of $P \cup P(\mathcal{S})$ such that

- $I(\mathcal{S}) \subseteq J_0$; and
- $J_0(B^B_B) = J_0(B) = I(B^B_B)$ for every ground subgoal B in $\mathcal{S}$.

Let $M_0 = LPM(\frac{P \cup P(\mathcal{S})}{J_0})$. By Theorem 6.2, M_0 is symmetric.

Following the same argument in Part (a), the program $P(\mathcal{S})$ is stratified after every occurrence of $\sim B^B_B$, where B is a ground subgoal in $\mathcal{S}$, is replaced with its truth value in J_0 (or equivalently in I), with a stratification, say $P_1 \cup ... \cup P_k$. Since $I \in \mathcal{ST}3(P(\mathcal{S}))$, I coincides with the unique perfect model of $P_1 \cup ... \cup P_k$. In addition, I and M_0 coincides on ground atoms in the lowest stratum P_1.

For each $i(1 \leq i \leq k)$, we construct interpretations J_i and M_i of $P \cup P(\mathcal{S})$:

- J_i is some symmetric interpretation such that
  - $I(\mathcal{S}) \subseteq J_i$; and
  - $J_i(B^B_B) = J_i(B) = I(B^B_B)$ for every ground subgoal B in $\mathcal{S}$; and
  - J_i and M_{i-1} coincide on all ground atoms in $\mathcal{HB}_{P(\mathcal{S})}$ whose stratum in the aforementioned stratification is less than or equal to i;

  J_i exists since M_{i-1} is a symmetric interpretation that contains $I(\mathcal{S})$.
- $M_i = LPM(\frac{P \cup P(\mathcal{S})}{J_i})$.

By the same arguments for J_0 and M_0, M_i is a symmetric interpretation for every $i(1 \leq i \leq k)$, and $M_k|_{P(\mathcal{S})} = I$.

We partition the Herbrand base $\mathcal{HB}_P$ into $\mathcal{HB}_1 \cup \mathcal{HB}_2$, where $\mathcal{HB}_1$ is the set of all atoms B such that $B \in |\mathcal{A}|$ for some subgoal $\mathcal{A}$ in $\mathcal{S}$, and $\mathcal{HB}_2 = \mathcal{HB}_P - \mathcal{HB}_1$. We construct a symmetric interpretation M of $P \cup P(\mathcal{S})$ as follows:

- $M|_{P(\mathcal{S})} = M_k|_{P(\mathcal{S})} = I$; and
- $M(B) = M_k(B)$ for every $B \in \mathcal{HB}_1$, where $B \in |\mathcal{A}|$ for some subgoal $\mathcal{A}$ in $\mathcal{S}$.

For atoms in $\mathcal{HB}_2$, their truth values in M are chosen as follows. We construct a program P_{simpl} from the Herbrand instantiation of P by

- deleting every rule whose head is an atom in $\mathcal{HB}_1$;
- deleting every rule whose body contains a positive literal B such that $B \in \mathcal{HB}_1$ and $M(B) = \mathbf{f}$;
- deleting every rule whose body contains a negative literal $\sim B$ such that $B \in \mathcal{HB}_1$ and $M(B) = \mathbf{t}$;
- replacing each positive literal B in the body of a rule with $\mathbf{u}$ if $B \in \mathcal{HB}_1$ and $M(B) = \mathbf{u}$;
- replacing each negative literal $\sim B$ in the body of a rule with $\mathbf{u}$ if $B \in \mathcal{HB}_1$ and $M(B) = \mathbf{u}$.

Consider each ground atom in P_{simpl} as a new propositional symbol, and let M_β be an arbitrary three-valued stable model of P_{simpl} viewed as a propositional program. Then for every atom $B \in \mathcal{HB}_2$,

- if B occurs in P_{simpl}, $M(B) = M_\beta(B)$;
- otherwise, $M(B) = \mathbf{f}$.

Notice that M is a symmetric interpretation of $P \cup P(\mathcal{S})$ such that $M|_{P(\mathcal{S})} = I$ and $I(\mathcal{S}) \subseteq M$. Let $M' = LPM(\frac{P \cup P(\mathcal{S})}{M})$. By Theorem 6.2, M' is symmetric. We show that $M = M'$. First, since $M|_{P(\mathcal{S})} = I$ and $I \in \mathcal{ST}3(P(\mathcal{S}))$ and P and $P(\mathcal{S})$ are independent of each other, $M'|_{P(\mathcal{S})} = I = M|_{P(\mathcal{S})}$. Second, for every atom $B \in \mathcal{HB}_1$, $M'(B) = M(B)$ as both M and M' are symmetric and $M'|_{P(\mathcal{S})} = M|_{P(\mathcal{S})}$. Third, for every atom $B \in \mathcal{HB}_2$, $M'(B) = M(B)$, which can be verified by the construction of P_{simpl} and the usage of a three-valued stable model M_β of P_{simpl} in the definition of M. Thus $M = M'$.

Since $M = M' = LPM(\frac{P \cup P(\mathcal{S})}{M})$ and P and $P(\mathcal{S})$ are independent of each other, $M|_P = LPM(\frac{P}{M|_P})$, which implies that $M|_P \in \mathcal{ST}3(P)$. Furthermore, $M|_{P(\mathcal{S})} = I$ and M is symmetric, and so (b) holds. □