Fixpoint Semantics for a Fragment of First-Order Linear Logic

Marco Bozzano *

May 22, 2001
CMU-CS-01-129

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**DISI, Università di Genova, Italy*

Abstract

In this paper we investigate the theoretical foundation of a bottom-up, fixpoint semantics for a subset of Girard's linear logic [Gir87]. More precisely, we consider a first-order formulation of a fragment of LinLog [And92] including multiplicative disjunction and universal quantification over goals. The semantics is defined by means of a fixpoint operator which is monotonic and continuous over an extended notion of interpretation lattice. We prove soundness and completeness of this semantics with respect to the usual operational semantics. We discuss some applications and related work.

This research was sponsored by the Office of Naval Research (ONR) under grant no. N00014-01-1-0432.

Keywords: linear logic, fixpoint semantics

1 Introduction

In recent years a number of fragments of linear logic [Gir87] have been proposed as a logical foundation for extensions of logic programming [Mil95]. Languages like LO [AP91], LinLog [And92], ACL [KY95], Lolli [HM94], and Lygon [HP94] have been proposed with the aim of enriching traditional logic programming languages like Prolog with a well-founded notion of state and with aspects of concurrency. The operational semantics of this class of languages is given via a sequent-calculi presentation of the corresponding fragment of linear logic. Special classes of proofs like the *focusing* proofs of [And92] and the *uniform* proofs of [Mil96] allow us to restrict our attention to *cut-free*, *goal-driven* proof systems that are complete with respect to provability in linear logic. These presentations of linear logic are the natural counterpart of the traditional *top-down* operational semantics of logic programs.

In this paper we discuss an alternative operational semantics for a first-order formulation of a fragment of linear logic which is a subset of LinLog [And92], a presentation of full linear logic. The fragment under consideration comprises the multiplicative disjunction $⅋$, additive truth $\top$, linear implication $\multimap$, and also universal quantification over goals. The $⅋$ connective can be thought of as a multiset constructor, and it can be used for instance to specify (first-order) multiset rewriting, while the universal quantifier can be used to reason about specifications which require introducing new values (e.g. names). The operational semantics we propose extends our previous works [BDM01a, BDM01b], and consists of a *goal-independent bottom-up* evaluation of programs. Specifically, given a program P our aim is to compute a finite representation of the set of goals that are provable from P. There are several reasons to look at this problem. For instance, the *provability* relation of our logic can be used to naturally express verification problems for Petri Net-like models of concurrent systems [BDM01b, CDL+99]. Also, a formal definition of the bottom-up semantics can be useful for studying equivalence, compositionality and abstract interpretation, as for traditional logic programs [BGLM94, GDL95].

Technically, the semantics is based on the definition of an extended notion of interpretation comprising multisets of non ground atoms, and of a fixpoint operator which turns out to be monotonic and continuous over the complete lattice corresponding to the interpretation domain. The fixpoint semantics is proved to be sound and complete with respect to the operational semantics. As in [BDM01a, BDM01b], the semantic definition is presented in two steps. More precisely, we first present a simple, non-effective notion of (concrete) interpretation and the corresponding definition of fixpoint operator, which we call T_P. We then present an extended notion of (abstract) interpretation, and we define a symbolic and effective version of the fixpoint operator, which we call S_P. The purpose of this double definition is to ease the proof of soundness and completeness. Namely, this latter proof is carried out by proving that the fixpoint of the T_P operator is equivalent to the operational semantics and that S_P is a symbolic version of T_P. The main contribution with respect to [BDM01a, BDM01b] is the management of universal quantification over goals. Universal quantification can be seen as a means for introducing new values or names (*eigenvariables*) during the computation. The semantic definition must then take into account the fact that program signatures can dynamically grow.

Plan of the paper. After introducing some preliminaries in Section 2, we present the logical fragment under consideration and its operational semantics in Section 3. In Section 4 and 5 we discuss the definition of our bottom-up, fixpoint semantics and we prove it is sound and complete with respect to the operational semantics. More precisely, in Section 4 we present the definition of the non effective fixpoint operator T_P, while in Section 5 we present its symbolic version S_P. Finally, in Section 6 we discuss conclusions and future work.

2 Preliminaries

Signatures, terms and atoms. Given a program P, we denote by Σ_P the signature comprising the set of constants, function and predicate symbols in P. We also assume to have an infinite set V of variable symbols (usually noted X, Y, Z, ...), and an infinite set E of new constants (*eigenvariables*). We denote by Sig_P the set of signatures which comprise at least the symbols in Σ_P (and possibly some eigenvariables). T_Σ denotes the set of *non ground* terms over Σ and A_Σ the set of *non ground* atoms over Σ.

Multisets. We will extensively use operations on multisets. Multisets of atoms over A_Σ will be hereafter called *facts*, and symbolically noted as $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, ... A multiset with (possibly duplicated) elements $a_1, \ldots, a_n \in A_\Sigma$ will be simply indicated as $\{a_1, \ldots, a_n\}$, overloading the usual notation for sets. A multiset $\mathcal{A}$ is uniquely determined by a map $Occ : A_\Sigma \rightarrow \mathbf{N}$ such that $Occ_\mathcal{A}(A)$ is the number of occurrences of A in $\mathcal{A}$. Multisets are ordered according to the *multiset inclusion* relation $\preccurlyeq$ defined as follows: $\mathcal{A} \preccurlyeq \mathcal{B}$ if and only if $Occ_\mathcal{A}(A) \leq Occ_\mathcal{B}(A)$ for every $A \in A_\Sigma$. We will also write $\mathcal{B} \succcurlyeq \mathcal{A}$ for $\mathcal{A} \preccurlyeq \mathcal{B}$. The *empty* multiset is denoted ϵ and is such that $Occ_\epsilon(A) = 0$ for every $A \in A_\Sigma$, and $\epsilon \preccurlyeq \mathcal{A}$ for any $\mathcal{A}$. *Multiset union* $\mathcal{A}, \mathcal{B}$ (alternatively written $\mathcal{A} + \mathcal{B}$ when ',' is ambiguous) is such that $Occ_{\mathcal{A},\mathcal{B}}(A) = Occ_\mathcal{A}(A) + Occ_\mathcal{B}(A)$ for every $A \in A_\Sigma$. *Multiset difference* $\mathcal{A} \setminus \mathcal{B}$ is such that $Occ_{\mathcal{A}\setminus\mathcal{B}}(A) = max(0, Occ_\mathcal{A}(A) - Occ_\mathcal{B}(A))$ for every $A \in A_\Sigma$. We also define a special operation $\bullet$ to compute the *least upper bound* of two multisets with respect to $\preccurlyeq$. Namely, $\mathcal{A} \bullet \mathcal{B}$ is such that $Occ_{\mathcal{A}\bullet\mathcal{B}}(A) = max(Occ_\mathcal{A}(A), Occ_\mathcal{B}(A))$ for every $A \in A_\Sigma$.

In the rest of the paper we will use Δ, Δ', ... to denote multisets of possibly compound formulas. Given two multisets Δ and Δ', $\Delta \preccurlyeq \Delta'$ indicates multiset inclusion and Δ, Δ' multiset union, as before, and $\Delta, \{G\}$ is written simply Δ, G. In the following, a *context* will denote a multiset of goal-formulas (a *fact* is a context in which every formula is atomic). Given a linear disjunction of atomic formulas $H = a_1 \parr \ldots \parr a_n$, we introduce the notation $\widehat{H}$ to denote the multiset $\{a_1, \ldots, a_n\}$.

Finally, let $T : \mathcal{I} \rightarrow \mathcal{I}$ be an operator defined over a complete lattice $\langle \mathcal{I}, \sqsubseteq \rangle$. We define $T\!\uparrow_0 = \emptyset$, where $\emptyset$ is the bottom element, $T\!\uparrow_{k+1} = T(T\!\uparrow_k)$ for all $k \geq 0$, and $T\!\uparrow_\omega = \bigsqcup_{k=1}^{\infty} T\!\uparrow_k$, where $\bigsqcup$ is the least upper bound w.r.t. $\sqsubseteq$. Furthermore, we use $lfp(T)$ to denote the *least fixpoint* of T.

Substitutions and Multiset Unifiers. We will use the usual notion of substitution as in traditional logic programming. Substitutions will be denoted by θ, σ, τ, ... , and $F\theta$ (where F is a fact, or a context, formula, ...) will denote the application of the substitution θ to F. Composition of two substitutions θ and σ will be denoted $\theta \circ \sigma$, e.g $F(\theta \circ \sigma)$ stands for $(F\theta)\sigma$. We will indicate the domain of a substitution θ by $Dom(\theta)$, and will say "θ defined on a signature Σ" meaning that θ can only map variables in $Dom(\theta)$ to terms in T_Σ. Finally, $FV(F)$ (where F is a fact, or a context, formula, ...), will denote the set of *free* variables of F, and $\theta_{|W}$, where W is a set of variables, will denote the restriction of θ to $Dom(\theta) \cap W$.

We need to lift the definition of *most general unifier* from expressions to multisets of expressions. Namely, given two multisets (with the same number of elements) $\mathcal{A} = \{a_1, \ldots, a_n\}$ and $\mathcal{B} = \{b_1, \ldots, b_n\}$, we define a most general unifier of $\mathcal{A}$ and $\mathcal{B}$, written $m.g.u.(\mathcal{A}, \mathcal{B})$, to be the most general unifier (defined in the usual way) of the two lists of expressions $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$. Note that in the notation $\{a_1, \ldots, a_n\}$ we don't care about the order of the elements, therefore there might be more than one way to unify two given multisets. We use the notation $\theta = m.g.u.(\mathcal{A}, \mathcal{B})$ to denote any unifier which is *non deterministically* picked from the set of most general unifiers of $\mathcal{A}$ and $\mathcal{B}$.

3 A Proof Theoretical Presentation

In this section we give a proof-theoretic presentation of the fragment of linear logic we are interested in. Basically it comprises the multiplicative disjunction of linear logic ($\parr$), additive truth ($\top$), linear implication $\multimap$ (we use the reversed notation $H \multimapinv G$ for $G \multimap H$), and the universal quantifier $\forall$. In this section we will define a proof system, based on sequent calculus, for this logic, and its operational semantics.

The logic programming language we discuss is based on linear logic, and can be described by the following grammar:

$$\mathbf{D} ::= \mathbf{A}_1 \parr \ldots \parr \mathbf{A}_n \multimapinv \mathbf{G} \mid \mathbf{D} \,\&\, \mathbf{D}$$
$$\mathbf{G} ::= \mathbf{G} \parr \mathbf{G} \mid \forall x.\mathbf{G} \mid \mathbf{A} \mid \top$$

where $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and $\mathbf{A}$ are atomic formulas. $\mathbf{G}$-formulas correspond to

$$\frac{}{P \vdash_\Sigma \top, \Delta}\ \top_r \qquad \frac{P \vdash_\Sigma G_1, G_2, \Delta}{P \vdash_\Sigma G_1 \mathbin{⅋} G_2, \Delta}\ ⅋_r$$

$$\frac{P \vdash_{\Sigma,c} G[c/x], \Delta}{P \vdash_\Sigma \forall x. G, \Delta}\ \forall_r\ \ (c \notin \Sigma) \qquad \frac{P \vdash_\Sigma G\theta, \mathcal{A}}{P \vdash_\Sigma \widehat{H}\theta, \mathcal{A}}\ bc\ \ (H \multimapinv G\ \in P\ (variant))$$

Figure 1: A proof theoretical presentation for the logic

goals to be evaluated in a given program, while **D**-formulas correspond to multiple-headed *program clauses*. A program P is a **D**-formula. Let P be the program $C_1 \,\&\, \ldots \,\&\, C_n$. The execution of a multiset of **G**-formulas $G_1, \ldots, G_k$ in Σ and P corresponds to a goal-driven proof for the two-sided sequent

$$P \vdash_\Sigma G_1, \ldots, G_k,$$

which is an abbreviation for the following two-sided linear logic sequent:

$$!C_1, \ldots, !C_n \Rightarrow_\Sigma G_1, \ldots, G_k.$$

The formula $!F$ on the left-hand side of a sequent indicates that F can be used in a proof an arbitrary number of times. This implies that a program can be viewed also as a *set of reusable clauses*. According to this view, the operational semantics is given via the *uniform* (goal-driven) proof system defined in Figure 1, where P is a set of implicational clauses, Σ is a signature in Sig_P, $\mathcal{A}$ denotes a multiset of atomic formulas, and Δ denotes a multiset of **G**-formulas. Note that all the formulas (and also substitutions) on the right hand side are implicitly assumed to range over the term language T_Σ, i.e. only the constants declared in Σ can appear in formulas. A sequent is provable if all branches of its proof tree terminate with instances of the $\top_r$ axiom. The proof system of Figure 1 is a specialization of more general uniform proof systems for linear logic like Andreoli's focusing proofs [And92] and Forum [Mil96]. The proof system is intended to define the set of *non ground* goals which are provable from a given program P (the so-called *C-semantics* of [FLMP93]), in other words free variables in sequents must be implicitly considered *universally* quantified. Rule *bc* denotes a backchaining (resolution) step (where $\widehat{H}$ is the multiset consisting of the literals in the disjunction H). We can assume $Dom(\theta) \subseteq FV(H) \cup FV(G)$ (note that $H \multimapinv G$ is assumed to be a variant, therefore it has no variables in common with $\mathcal{A}$). Note that *bc* can be executed only if the right-hand side of the current sequent consists of atomic formulas. Rule $\forall_r$ is responsible for signature augmentation. Clauses having the following form

$$a_1 \mathbin{⅋} \ldots \mathbin{⅋} a_n \multimapinv \top$$

play the same role as the unit clauses of Horn programs. In fact, a backchaining step over such a clause leads to *success* independently of the current context $\mathcal{A}$, as shown in the following scheme:

$$\dfrac{\dfrac{}{P \vdash_\Sigma \top, \mathcal{A}}\ \top_r}{P \vdash_\Sigma a_1, \ldots, a_n, \mathcal{A}}\ bc$$

$$provided \quad a_1 \mathbin{⅋} \ldots \mathbin{⅋} a_n \multimapinv \top \in P$$

This observation leads us to the following property (we recall that $\preccurlyeq$ is the sub-multiset relation).

Proposition 1 *Given a program P, a signature $\Sigma \in Sig_P$, and two contexts Δ, Δ' such that $\Delta \preccurlyeq \Delta'$, if $P \vdash_\Sigma \Delta$ then $P \vdash_\Sigma \Delta'$.*

Proof. By simple induction on the structure of proofs. □

We conclude this section with the definition of the (non ground) operational semantics.

Definition 3.1 (Operational Semantics) *Given a program P, its operational semantics, denoted $O(P)$, is given by*

$$O(P) = \{\mathcal{A} \mid \mathcal{A} \textit{ is a multiset of (non ground) atoms in } A_{\Sigma_P} \textit{ and } P \vdash_{\Sigma_P} \mathcal{A}\}.$$

Note that, according to [And92], the information on provable *facts* from a given program P is all we need to decide whether a general goal (with possible nesting of connectives) is provable from P or not. In fact, provability of a compound goal can always be reduced to provability of a finite set of atomic multisets.

4 Bottom-Up Semantics

We will now discuss the definition of a *bottom-up* semantics for the logic language of Section 3. The semantics should enjoy the usual properties of classical bottom-up semantics, in particular its definition should be based on an *effective* fixpoint operator (i.e. at least every single step must be finitely computable), and it should be *goal-independent*. In this section we present the first version of our fixpoint operator, called T_P, while in Section 5 we will present its symbolic and effective version, called S_P.

In presence of universal quantification and therefore signature augmentation, we need to extend the definition of Herbrand base and (concrete) interpretations as follows. Namely, the definition of Herbrand base now depends explicitly on the signature, and interpretations can be thought of as infinite tuples, with one element for every signature $\Sigma \in Sig_P$.

Definition 4.1 (Herbrand Base) *Given a program P and a signature $\Sigma \in Sig_P$, the Herbrand base of P over Σ, denoted $HB_\Sigma(P)$, is given by*

$$HB_\Sigma(P) = \{\mathcal{A} \mid \mathcal{A} \textit{ is a multiset of (non ground) atoms in } A_\Sigma\}.$$

Definition 4.2 (Interpretations) *Given a program P, a (concrete) interpretation is a family of sets $(I_\Sigma)_{\Sigma \in Sig_P}$, where $I_\Sigma \in \mathcal{P}(HB_\Sigma(P))$ for every $\Sigma \in Sig_P$.*

In the following we often use the notation I for an interpretation to denote the family $(I_\Sigma)_{\Sigma \in Sig_P}$.

Interpretations form a complete lattice where inclusion and least upper bound are defined like (component-wise) set inclusion and union. In the following definition we therefore overload the symbols $\subseteq$ and $\cup$ for sets.

Definition 4.3 (Interpretation Domain) *Interpretations form a complete lattice $\langle \mathcal{D}, \subseteq \rangle$, where:*

- *$\mathcal{D} = \{I \mid I$ is an interpretation$\}$;*
- *$I \subseteq J$ iff $I_\Sigma \subseteq J_\Sigma$ for every $\Sigma \in Sig_P$;*
- *the least upper bound of I and J is $(I_\Sigma \cup J_\Sigma)_{\Sigma \in Sig_P}$;*
- *the bottom and top elements are $\emptyset = (\emptyset_\Sigma)_{\Sigma \in Sig_P}$ and $(HB_\Sigma(P))_{\Sigma \in Sig_P}$, respectively.*

Before introducing the definition of fixpoint operator, we need to define a notion of satisfiability of a context Δ in a given interpretation I. For this purpose, we introduce the judgment $I \models_\Sigma \Delta \rhd \mathcal{C}$, where I is an interpretation, Δ is a context, and $\mathcal{C}$ is an output fact. The judgment is also parametric with respect to a given signature Σ. The parameter $\mathcal{C}$ must be thought of as an *output* fact such that $\mathcal{C} + \Delta$ is valid in I. The notion of output fact will simplify the presentation of the algorithmic version of the judgment which we will present in Section 5. This notion of *satisfiability* is modeled according to the right-introduction (decomposition) rules of the proof system. In other words, the computation performed by the satisfiability judgment corresponds to *top-down* steps inside our *bottom-up* semantics. In what follows we always make the implicit assumption that Δ is a context defined over Σ, and, as a result, also the output fact $\mathcal{C}$ must be defined over Σ.

Definition 4.4 (Satisfiability Judgment) *Let P be a program, $\Sigma \in Sig_P$, and $I = (I_\Sigma)_{\Sigma \in Sig_P}$ an interpretation. The satisfiability judgment $\models_\Sigma$ is defined as follows:*

$$I \models_\Sigma \top, \Delta \rhd \mathcal{C} \textit{ for any fact } \mathcal{C} \textit{ in } A_\Sigma;$$

$$I \models_\Sigma \mathcal{A} \rhd \mathcal{C} \textit{ if } \mathcal{A} + \mathcal{C} \in I_\Sigma;$$

$$I \models_\Sigma \forall x. G, \Delta \rhd \mathcal{C} \textit{ if } I \models_{\Sigma,c} G[c/x], \Delta \rhd \mathcal{C}, \textit{ with } c \notin \Sigma;$$

$$I \models_\Sigma G_1 \parr G_2, \Delta \rhd \mathcal{C} \textit{ if } I \models_\Sigma G_1, G_2, \Delta \rhd \mathcal{C}.$$

The satisfiability judgment $\models_\Sigma$ satisfies the following properties.

Lemma 1 *For every interpretation $I = (I_\Sigma)_{\Sigma \in Sig_P}$, context Δ, and fact $\mathcal{C}$, $I \models_\Sigma \Delta \rhd \mathcal{C}$ iff $I \models_\Sigma \Delta, \mathcal{C} \rhd \epsilon$.*

Proof. "If part". By induction on the derivation of $I \models_\Sigma \Delta, \mathcal{C} \rhd \epsilon$.

- If $\Delta = \top, \Delta'$, obvious;
- if $\Delta = \mathcal{A}$ and $\mathcal{A} + \mathcal{C} \in I_\Sigma$, then also $I \models_\Sigma \mathcal{A} \rhd \mathcal{C}$ holds;
- if $\Delta = \forall x. G, \Delta'$ and $I \models_{\Sigma,c} G[c/x], \Delta', \mathcal{C} \rhd \epsilon$, with $c \notin \Sigma$, then by inductive hypothesis $I \models_{\Sigma,c} G[c/x], \Delta' \rhd \mathcal{C}$, which implies $I \models_\Sigma \forall x. G, \Delta' \rhd \mathcal{C}$;
- if $\Delta = G_1 \invamp G_2, \Delta'$, the conclusion follows by a straightforward application of the inductive hypothesis.

"Only if" part. By induction on the derivation of $I \models_\Sigma \Delta \rhd \mathcal{C}$.

- $\Delta = \top, \Delta'$, obvious;
- if $\Delta = \mathcal{A}$ and $\mathcal{A} + \mathcal{C} \in I_\Sigma$, then also $I \models_\Sigma \mathcal{A}, \mathcal{C} \rhd \epsilon$ holds;
- if $\Delta = \forall x. G, \Delta'$ and $I \models_{\Sigma,c} G[c/x], \Delta' \rhd \mathcal{C}$, with $c \notin \Sigma$, then by inductive hypothesis $I \models_{\Sigma,c} G[c/x], \Delta', \mathcal{C} \rhd \epsilon$, which implies $I \models_\Sigma \forall x. G, \Delta', \mathcal{C} \rhd \epsilon$;
- if $\Delta = G_1 \invamp G_2, \Delta'$ the conclusion follows by a straightforward application of the inductive hypothesis.

□

Lemma 2 *For any interpretations $I_1 = ((I_1)_\Sigma)_{\Sigma \in Sig_P}$, $I_2 = ((I_2)_\Sigma)_{\Sigma \in Sig_P}$, ..., context Δ, and fact $\mathcal{C}$,*

i) if $I_1 \subseteq I_2$ and $I_1 \models_\Sigma \Delta \rhd \mathcal{C}$ then $I_2 \models_\Sigma \Delta \rhd \mathcal{C}$;

ii) if $I_1 \subseteq I_2 \subseteq \ldots$ and $\bigcup_{i=1}^{\infty} I_i \models_\Sigma \Delta \rhd \mathcal{C}$ then there exists $k \in \mathbf{N}$ s.t. $I_k \models_\Sigma \Delta \rhd \mathcal{C}$.

Proof.

i) By induction on the derivation of $I_1 \models_\Sigma \Delta \rhd \mathcal{C}$.

- If $\Delta = \top, \Delta'$, obvious;
- if $\Delta = \mathcal{A}$ and $\mathcal{A} + \mathcal{C} \in (I_1)_\Sigma$, then $\mathcal{A} + \mathcal{C} \in (I_2)_\Sigma$, because $I_1 \subseteq I_2$, therefore $I_2 \models_\Sigma \mathcal{A} \rhd \mathcal{C}$;

- if $\Delta = \forall x.G, \Delta'$ and $I_1 \models_{\Sigma,c} G[c/x], \Delta' \rhd \mathcal{C}$, with $c \notin \Sigma$, then by inductive hypothesis $I_2 \models_{\Sigma,c} G[c/x], \Delta' \rhd \mathcal{C}$, which implies $I_2 \models_{\Sigma} \forall x.G, \Delta' \rhd \mathcal{C}$;
- if $\Delta = G_1 \parr G_2, \Delta'$, the conclusion follows by a straightforward application of the inductive hypothesis.

ii) By induction on the derivation of $\bigcup_{i=1}^{\infty} I_i \models_{\Sigma} \Delta \rhd \mathcal{C}$.

- If $\Delta = \top, \Delta'$, then for every $k \in \mathbf{N}$, $I_k \models_{\Sigma} \Delta \rhd \mathcal{C}$;
- if $\Delta = \mathcal{A}$ and $\mathcal{A} + \mathcal{C} \in (\bigcup_{i=1}^{\infty} I_i)_{\Sigma}$, there exists $k \in \mathbf{N}$ s.t. $\mathcal{A} + \mathcal{C} \in (I_k)_{\Sigma}$, i.e. $I_k \models_{\Sigma} \mathcal{A} \rhd \mathcal{C}$;
- if $\Delta = \forall x.G, \Delta'$ and $\bigcup_{i=1}^{\infty} I_i \models_{\Sigma,c} G[c/x], \Delta' \rhd \mathcal{C}$, with $c \notin \Sigma$, then by inductive hypothesis there exists $k \in \mathbf{N}$ s.t. $I_k \models_{\Sigma,c} G[c/x], \Delta' \rhd \mathcal{C}$, therefore $I_k \models_{\Sigma} \forall x.G, \Delta' \rhd \mathcal{C}$;
- if $\Delta = G_1 \parr G_2, \Delta'$, the conclusion follows by a straightforward application of the inductive hypothesis.

$\square$

We are now ready to define the fixpoint operator T_P.

Definition 4.5 (Fixpoint Operator T_P) *Given a program P and an interpretation $I = (I_\Sigma)_{\Sigma \in Sig_P}$, the fixpoint operator T_P is defined as follows:*

$$T_P(I) = ((T_P(I))_\Sigma)_{\Sigma \in Sig_P}$$

$$(T_P(I))_\Sigma = \{\widehat{H}\theta + \mathcal{C} \mid \textit{there exist } (H \multimapinv G) \in P \textit{ (variant) and a substitution } \theta \textit{ s.t. } I \models_\Sigma G\theta \rhd \mathcal{C}\}.$$

In the previous definition, θ is implicitly assumed to be defined over Σ. The following property holds.

Proposition 2 (Monotonicity and continuity) *For every program P, the fixpoint operator T_P is monotonic and continuous over the lattice $\langle \mathcal{D}, \subseteq \rangle$.*

Proof. Monotonicity follows immediately from T_P definition and Lemma 2 i). To prove continuity, it is sufficient to prove that T_P is finitary, i.e., for any increasing chain of interpretations $I_1 \subseteq I_2 \subseteq \ldots$ we have that $T_P(\bigcup_{i=1}^{\infty} I_i) \subseteq \bigcup_{i=1}^{\infty} T_P(I_i)$. We will prove that for every signature $\Sigma \in Sig_P$ and for every context Δ, if $T_P(\bigcup_{i=1}^{\infty} I_i) \models_\Sigma \Delta \rhd \epsilon$ then there exist $k \in \mathbf{N}$ s.t. $T_P(I_k) \models_\Sigma \Delta \rhd \epsilon$, from which the conclusion follows immediately. The proof is by induction on the derivation of $T_P(\bigcup_{i=1}^{\infty} I_i) \models_\Sigma \Delta \rhd \epsilon$.

- If $\Delta = \top, \Delta'$, then for every $k \in \mathbf{N}$, $T_P(I_k) \models_\Sigma \Delta \rhd \epsilon$;

- if $\Delta = \mathcal{A}$ and $\mathcal{A} \in (T_P(\bigcup_{i=1}^{\infty} I_i))_\Sigma$, then there exist a variant $H \multimapinv G$ of a clause in P and a substitution θ s.t. $\bigcup_{i=1}^{\infty} I_i \models_\Sigma G\theta \rhd \mathcal{C}$ and $\mathcal{A} = \widehat{H}\theta + \mathcal{C}$. By Lemma 2 ii), we have that there exist $k \in \mathbf{N}$ s.t. $I_k \models_\Sigma G\theta \rhd \mathcal{C}$, therefore $\widehat{H}\theta + \mathcal{C} = \mathcal{A} \in (T_P(I_k)_\Sigma)$, which implies $T_P(I_k) \models_\Sigma \mathcal{A} \rhd \epsilon$;
- if $\Delta = \forall x. G, \Delta'$ and $T_P(\bigcup_{i=1}^{\infty} I_i) \models_{\Sigma,c} G[c/x], \Delta' \rhd \epsilon$, with $c \notin \Sigma$, then by inductive hypothesis there exist $k \in \mathbf{N}$ s.t. $T_P(I_k) \models_{\Sigma,c} G[c/x], \Delta' \rhd \epsilon$, therefore we can conclude that $T_P(I_k) \models_\Sigma \forall x. G, \Delta' \rhd \epsilon$;
- if $\Delta = G_1 \parr G_2, \Delta'$ the conclusion follows by a straightforward application of the inductive hypothesis.

$\square$

Monotonicity and continuity of the T_P operator imply, by Tarski's Theorem, that $lfp(T_P) = T_P{\uparrow}_\omega$. The fixpoint semantics of a program P is then defined as follows.

Definition 4.6 (Fixpoint Semantics) *Given a program P, its fixpoint semantics, denoted $F(P)$, is defined as follows:*

$$F(P) = (lfp(T_P))_{\Sigma_P} = (T_P{\uparrow}_\omega((\emptyset_\Sigma)_{\Sigma \in Sig_P}))_{\Sigma_P}.$$

We conclude this section by proving the following fundamental result, which states that the fixpoint semantics is sound and complete with respect to the operational semantics.

Theorem 1 *For every program P, $F(P) = O(P)$.*

Proof. $F(P) \subseteq O(P)$. We prove that for every $k \in \mathbf{N}$, for every signature $\Sigma \in Sig_P$, and for every context Δ, $T_P{\uparrow}_k \models_\Sigma \Delta \rhd \epsilon$ implies $P \vdash_\Sigma \Delta$. The proof is by lexicographic induction on (k, h), where h is the length of the derivation of $T_P{\uparrow}_k \models_\Sigma \Delta \rhd \epsilon$.

- If $\Delta = \top, \Delta'$, obvious;
- if $\Delta = \mathcal{A}$ and $\mathcal{A} \in (T_P{\uparrow}_k)_\Sigma$, then there exist a variant $H \multimapinv G$ of a clause in P, a fact $\mathcal{C}$ and a substitution θ s.t. $\mathcal{A} = \widehat{H}\theta + \mathcal{C}$ and $T_P{\uparrow}_{k-1} \models_\Sigma G\theta \rhd \mathcal{C}$. By Lemma 1, this implies $T_P{\uparrow}_{k-1} \models_\Sigma G\theta, \mathcal{C} \rhd \epsilon$. Then by inductive hypothesis we have $P \vdash_\Sigma G\theta, \mathcal{C}$, from which $P \vdash_\Sigma \widehat{H}\theta, \mathcal{C}$, i.e. $P \vdash_\Sigma \mathcal{A}$ follows by bc rule;
- if $\Delta = \forall x. G, \Delta'$ and $T_P{\uparrow}_k \models_{\Sigma,c} G[c/x], \Delta' \rhd \epsilon$, with $c \notin \Sigma$, then by inductive hypothesis we have $P \vdash_{\Sigma,c} G[c/x], \Delta'$ from which $P \vdash_\Sigma \forall x. G, \Delta'$ follows by $\forall_r$ rule;
- if $\Delta = G_1 \parr G_2, \Delta'$ and $T_P{\uparrow}_k \models_\Sigma G_1, G_2, \Delta' \rhd \epsilon$, then by inductive hypothesis we have $P \vdash_\Sigma G_1, G_2, \Delta'$, from which $P \vdash_\Sigma G_1 \parr G_2, \Delta'$ follows by $\parr_r$ rule.

$O(P) \subseteq F(P)$. We prove that for every signature $\Sigma \in Sig_P$ and for every context Δ, if $P \vdash_\Sigma \Delta$ then there exists $k \in \mathbf{N}$ s.t. $T_P{\uparrow}_k \models_\Sigma \Delta \rhd \epsilon$. The proof is by induction on the derivation of $P \vdash_\Sigma \Delta$.

- If $\Delta = \top, \Delta'$, then for every $k \in \mathbf{N}$, $T_P{\uparrow}_k \models_\Sigma \Delta \rhd \epsilon$;
- if $\Delta = \widehat{H}\theta, \mathcal{A}$, with $H \multimapinv G$ variant of a clause in P, θ substitution, and $P \vdash_\Sigma G\theta, \mathcal{A}$, then by inductive hypothesis we have that there exists $k \in \mathbf{N}$ s.t. $T_P{\uparrow}_k \models_\Sigma G\theta, \mathcal{A} \rhd \epsilon$. Then, by Lemma 1, $T_P{\uparrow}_k \models_\Sigma G\theta \rhd \mathcal{A}$. By T_P definition, $\widehat{H}\theta + \mathcal{A} \in (T_P{\uparrow}_{k+1})_\Sigma$, which implies $T_P{\uparrow}_{k+1} \models_\Sigma \widehat{H}\theta + \mathcal{A} \rhd \epsilon$;
- if $\Delta = \forall x. G, \Delta'$ and $P \vdash_{\Sigma,c} G[c/x], \Delta'$, with $c \notin \Sigma$, then by inductive hypothesis we have that there exist $k \in \mathbf{N}$ s.t. $T_P{\uparrow}_k \models_{\Sigma,c} G[c/x], \Delta' \rhd \epsilon$, from which $T_P{\uparrow}_k \models_\Sigma \forall x. G, \Delta' \rhd \epsilon$ follows;
- if $\Delta = G_1 \parr G_2, \Delta'$ the conclusion follows by a straightforward application of the inductive hypothesis.

$\square$

5 Effective Fixpoint Semantics

The fixpoint operator T_P defined in the previous section does not enjoy one of the crucial properties we required for our bottom-up semantics, namely its definition is *not* effective. This is a result of both the definition of the satisfiability judgment (whose clause for $\top$ is clearly not effective) and the definition of interpretations as infinite tuples. In order to solve these problems, we first define the (abstract) Herbrand base and (abstract) interpretations as follows.

Definition 5.1 (Abstract Herbrand Base) *Given a program P, the Herbrand base of P, denoted $HB(P)$, is given by*

$$HB(P) = HB_{\Sigma_P}(P).$$

Definition 5.2 (Abstract Interpretations) *Given a program P, an interpretation I is any subset of $HB(P)$, i.e. $I \in \mathcal{P}(HB(P))$.*

In order to define the abstract domain of interpretations, we need the following definitions.

Definition 5.3 (Instance Operator) *Given an interpretation I and a signature $\Sigma \in Sig_P$, we define the operator $Inst_\Sigma$ as follows:*

$$Inst_\Sigma(I) = \{\mathcal{A}\theta \mid \mathcal{A} \in I,\ \theta \text{ substitution}\}.$$

Definition 5.4 (Upward-closure Operator) *Given an interpretation I and a signature $\Sigma \in Sig_P$, we define the operator Up_Σ as follows:*

$$Up_\Sigma(I) = \{\mathcal{A} + \mathcal{C} \mid \mathcal{A} \in I\}.$$

As usual, in the previous definitions we assume θ and the fact $\mathcal{C}$ to be defined over Σ. The following definition provides the connection between the (abstract) interpretations defined in Definition 5.2 and the (concrete) interpretations of Definition 4.2. The idea behind the definition is that an interpretation implicitly *denotes* the set of elements which can be obtained by either instantiating or closing upward elements in the interpretation itself (where the concepts of instantiation and upward-closure are made precise by the above definitions). The operation of instantiation is justified by the fact that free variables are implicitly universally quantified, while the operation of upward-closure is justified by Proposition 1. Note that the instantiation and upward-closure operations are performed for every possible signature $\Sigma \in Sig_P$.

Definition 5.5 (Denotation of an Interpretation) *Given an (abstract) interpretation I, its denotation $[\![I]\!]$ is the (concrete) interpretation $([\![I]\!]_\Sigma)_{\Sigma \in Sig_P}$ defined as follows:*

$$[\![I]\!]_\Sigma = Inst_\Sigma(Up_\Sigma(I)) \quad (\textit{or, equivalently, } [\![I]\!]_\Sigma = Up_\Sigma(Inst_\Sigma(I))).$$

Two interpretations I and J are said to be equivalent, written $I \simeq J$, iff $[\![I]\!] = [\![J]\!]$.

The equivalence of the two different equations in Definition 5.5 is stated in the following proposition.

Proposition 3 *For every interpretation I, and signature $\Sigma \in Sig_P$,*

$$Inst_\Sigma(Up_\Sigma(I)) = Up_\Sigma(Inst_\Sigma(I)).$$

Proof. Let $(\mathcal{A} + \mathcal{C})\theta \in Inst_\Sigma(Up_\Sigma(I))$, with $\mathcal{A} \in I$. Then $(\mathcal{A} + \mathcal{C})\theta = (\mathcal{A}\theta) + \mathcal{C}\theta \in Up_\Sigma(Inst_\Sigma(I))$. Vice versa, let $\mathcal{A}\theta + \mathcal{C} \in Up_\Sigma(Inst_\Sigma(I))$, with $\mathcal{A} \in I$. Let $\mathcal{B}$ be a variant of $\mathcal{C}$ with new variables (not appearing in $\mathcal{A}$, θ, and $\mathcal{C}$) and θ' be the substitution with domain $Dom(\theta) \cup FV(\mathcal{B})$ and s.t. $\theta'_{|Dom(\theta)} = \theta$ and θ' maps $\mathcal{B}$ to $\mathcal{C}$. Then $\mathcal{A}\theta + \mathcal{C} = \mathcal{A}\theta' + \mathcal{B}\theta' = (\mathcal{A} + \mathcal{B})\theta' \in Inst_\Sigma(Up_\Sigma(I))$. □

We are now ready to define the abstract interpretation domain. As we do not need to distinguish between interpretations having the same denotation, we simply identify them using equivalence classes with respect to the corresponding equivalence relation $\simeq$.

Definition 5.6 (Abstract Interpretation Domain) *Abstract interpretations form a complete lattice $\langle \mathcal{I}, \sqsubseteq \rangle$, where*

- $\mathcal{I} = \{[I]_{\simeq} \mid I$ *is an interpretation*$\}$;
- $[I]_{\simeq} \sqsubseteq [J]_{\simeq}$ *iff* $[\![I]\!] \subseteq [\![J]\!]$*, i.e. iff for every* $\mathcal{A} \in I$*, there exist* $\mathcal{B} \in J$*, a substitution* θ *and a fact* $\mathcal{C}$ *(defined over* Σ_P*) s.t.* $\mathcal{A} = \mathcal{B}\theta + \mathcal{C}$;
- *the least upper bound of* $[I]_{\simeq}$ *and* $[J]_{\simeq}$*, written* $[I]_{\simeq} \bigsqcup [J]_{\simeq}$*, is* $[I \cup J]_{\simeq}$;
- *the bottom and top elements are* $[\emptyset]_{\simeq}$ *and* $[\epsilon]_{\simeq}$*, respectively.*

The condition $\mathcal{A} = \mathcal{B}\theta + \mathcal{C}$ provides for an effective method for testing the $\sqsubseteq$ relation over interpretations. Equivalence with the non effective condition $[\![I]\!] \subseteq [\![J]\!]$ is stated in the following proposition. We will need this result later on.

Proposition 4 *Given two interpretations* I *and* J*,* $[\![I]\!] \subseteq [\![J]\!]$ *iff for every* $\mathcal{A} \in I$*, there exist* $\mathcal{B} \in J$*, a substitution* θ *and a fact* $\mathcal{C}$ *(defined over* Σ_P*) s.t.* $\mathcal{A} = \mathcal{B}\theta + \mathcal{C}$*.*

Proof. "If part". We prove that for every $\Sigma \in Sig_P$, $[\![I]\!]_\Sigma \subseteq [\![J]\!]_\Sigma$. Let $\mathcal{A}' = \mathcal{A}\theta' + \mathcal{C}' \in Up_\Sigma(Inst_\Sigma(I)) = [\![I]\!]_\Sigma$, with $\mathcal{A} \in I$ and θ', $\mathcal{C}'$ defined over Σ. By hypothesis, there exist $\mathcal{B} \in J$, a substitution θ and a fact $\mathcal{C}$ (defined over Σ_P) s.t. $\mathcal{A} = \mathcal{B}\theta + \mathcal{C}$. Therefore, $\mathcal{A}' = \mathcal{A}\theta' + \mathcal{C}' = (\mathcal{B}\theta + \mathcal{C})\theta' + \mathcal{C}' = \mathcal{B}\theta\theta' + (\mathcal{C}\theta' + \mathcal{C}') \in Up_\Sigma(Inst_\Sigma(J)) = [\![J]\!]_\Sigma$ (note that $\theta\theta'$ and $\mathcal{C}\theta' + \mathcal{C}'$ are both defined over Σ because $\Sigma_P \subseteq \Sigma$).
"Only if" part. Let $\mathcal{A} \in I$, then $\mathcal{A} \in [\![I]\!]_{\Sigma_P}$ (note that $\mathcal{A}$ is defined over Σ_P by definition of interpretation). Then, by hypothesis we have that $\mathcal{A} \in [\![J]\!]_{\Sigma_P} = Up_{\Sigma_P}(Inst_{\Sigma_P}(J))$, i.e. there exist $\mathcal{B} \in J$, a substitution θ and a fact $\mathcal{C}$ (defined over Σ_P) s.t. $\mathcal{A} = \mathcal{B}\theta + \mathcal{C}$. □

We now define the abstract satisfiability judgment $I \Vdash_\Sigma \Delta \rhd \mathcal{C}\ ;\ \theta$, where I is an interpretation, Δ is a context, $\mathcal{C}$ is an output fact, and θ is an output substitution. As usual, we make the implicit assumption that Δ is a context defined over Σ, and, as a result, also the output fact $\mathcal{C}$ and substitution θ must be defined over Σ. The judgment $\Vdash_\Sigma$ can be thought of as an abstract version of the judgment $\models_\Sigma$. We now need one more parameter, namely an output substitution. The idea behind the definition is that the output fact $\mathcal{C}$ and the output substitution θ are *minimal* (in a sense to be clarified) so that they can be computed effectively given a program P, an interpretation I, and a signature Σ. The output substitution θ is needed in order to cope with clause instantiation, and its minimality is ensured by using most general unifiers in the definition.

Definition 5.7 (Abstract Satisfiability Judgment) *Let* P *be a program,* I *an interpretation, and* $\Sigma \in Sig_P$*. The abstract satisfiability judgment* $\Vdash_\Sigma$ *is* *defined as follows:*

$I \Vdash_\Sigma \top, \Delta \rhd \epsilon \; ; \; \epsilon$;

$I \Vdash_\Sigma \mathcal{A} \rhd \mathcal{C} \; ; \; \theta$ *if there exist* $\mathcal{B} \in I$ *(variant),* $\mathcal{B}' \preccurlyeq \mathcal{B}, \mathcal{A}' \preccurlyeq \mathcal{A}$, $\mathcal{C} = \mathcal{B} \setminus \mathcal{B}'$, *and* $\theta = m.g.u.(\mathcal{B}', \mathcal{A}')_{|FV(\mathcal{A},\mathcal{C})}$;

$I \Vdash_\Sigma \forall x. G, \Delta \rhd \mathcal{C} \; ; \; \theta$ *if* $I \Vdash_{\Sigma,c} G[c/x], \Delta \rhd \mathcal{C} \; ; \; \theta$, *with* $c \notin \Sigma$;

$I \Vdash_\Sigma G_1 \parr G_2, \Delta \rhd \mathcal{C} \; ; \; \theta$ *if* $I \Vdash_\Sigma G_1, G_2, \Delta \rhd \mathcal{C} \; ; \; \theta$.

Example 1 *Let I be the interpretation with the only multiset $\{p(X), q(X)\}$ (for simplicity, hereafter we omit parenthesis in multiset notation), and P the program*

$$\begin{array}{ll} 1. & r(Y) \multimapinv q(f(Y)) \\ 2. & s(Z) \multimapinv \forall x. p(X) \parr t(Z) \end{array}$$

Let's consider (a renaming of) the body of the first clause, $q(f(Y'))$, and (a renaming of) the element in I, $p(X'), q(X')$. Using the second clause for the judgment $\Vdash_{\Sigma_P}$, with $\mathcal{A} = \mathcal{A}' = q(f(Y'))$, $\mathcal{B} = p(X'), q(X')$, $\mathcal{B}' = q(X')$, we get

$$I \Vdash_{\Sigma_P} q(f(Y')) \rhd p(X') \; ; \; [X' \rightsquigarrow f(Y')].$$

Let's consider now (a renaming of) the body of the second clause, $\forall x. p(X) \parr t(Z')$, and the same element $p(X'), q(X')$. From the last clause for $\Vdash_{\Sigma_P}$, we have that $I \Vdash_{\Sigma_P} \forall x. p(X) \parr t(Z') \rhd \mathcal{C} \; ; \; \theta$ if $I \Vdash_{\Sigma_P} \forall x. p(X), t(Z') \rhd \mathcal{C} \; ; \; \theta$. From the third clause for $\Vdash_{\Sigma_P}$, this holds if $I \Vdash_{\Sigma_P,c} p(c), t(Z') \rhd \mathcal{C} \; ; \; \theta$, with $c \notin \Sigma_P$. Now, we can apply the second rule for $\Vdash_{\Sigma_P,c}$. Unfortunately, we can't choose $\mathcal{A}'$ to be $p(c)$ and $\mathcal{B}'$ to be $p(X')$. In fact, by unifying $p(c)$ with $p(X')$, we should get the substitution $\theta = [X' \rightsquigarrow c]$ and the output fact $q(X')$ (note that X' is a free variable in the output fact) and this is not allowed because the substitution θ must be defined on Σ_P, in order for $I \Vdash_{\Sigma_P} \forall x. p(X), t(Z') \rhd \mathcal{C} \; ; \; \theta$ to be meaningful. It turns out that the only way to use the second clause for $\Vdash_{\Sigma_P,c}$ is to choose $\mathcal{A}' = \mathcal{B}' = \epsilon$, which is useless in the fixpoint computation (see Example 2). □

The following lemma states a simple property of the substitution domain, which we will need in the following.

Lemma 3 *For every interpretation I, context Δ, fact $\mathcal{C}$, and substitution θ, if $I \Vdash_\Sigma \Delta \rhd \mathcal{C} \; ; \; \theta$ then $Dom(\theta) \subseteq FV(\Delta) \cup FV(\mathcal{C})$.*

Proof. Immediate by induction on the definition of the judgment. □

The connection between the satisfiability judgments $\models_\Sigma$ and $\Vdash_\Sigma$ is clarified by the following lemma.

Lemma 4 *For every interpretation I, context Δ, fact $\mathcal{C}$, and substitution θ,* *i) if $I \Vdash_\Sigma \Delta \rhd \mathcal{C} \; ; \; \theta$ then $[\![I]\!] \models_\Sigma \Delta\theta\theta' \rhd \mathcal{C}'\theta'$ for every substitution θ' and fact $\mathcal{C}' \succcurlyeq \mathcal{C}\theta$;*

ii) if $[\![I]\!] \models_\Sigma \Delta\theta \rhd \mathcal{C}$ then there exist a fact $\mathcal{C}'$, and substitutions θ' and σ s.t. $I \Vdash_\Sigma \Delta \rhd \mathcal{C}' \; ; \; \theta'$, $\theta_{|FV(\Delta)} = (\theta' \circ \sigma)_{|FV(\Delta)}$, $\mathcal{C}'\theta'\sigma \preccurlyeq \mathcal{C}$.

Proof.

i) By induction on the derivation of $I \Vdash_\Sigma \Delta \rhd \mathcal{C} \; ; \; \theta$.

- If $\Delta = \top, \Delta'$, obvious;
- suppose $\Delta = \mathcal{A}$, with $\mathcal{B} \in I$ (variant), $\mathcal{B}' \preccurlyeq \mathcal{B}$, $\mathcal{A}' \preccurlyeq \mathcal{A}$, $\mathcal{C} = \mathcal{B} \backslash \mathcal{B}'$, and $\theta = m.g.u.(\mathcal{B}', \mathcal{A}')_{|FV(\mathcal{A},\mathcal{C})}$. We want to prove that $[\![I]\!] \models_\Sigma \mathcal{A}\theta\theta' \rhd \mathcal{C}'\theta'$ for every substitution θ' and fact $\mathcal{C}' \succcurlyeq \mathcal{C}\theta$, i.e. $\mathcal{A}\theta\theta' + \mathcal{C}\theta\theta' + \mathcal{D}\theta' \in [\![I]\!]_\Sigma$ for every substitution θ' and fact $\mathcal{D}$. Now, $\mathcal{A}\theta\theta' + \mathcal{C}\theta\theta' + \mathcal{D}\theta' = (\mathcal{A}\theta + \mathcal{C}\theta + \mathcal{D})\theta' = (\mathcal{A}'\theta + (\mathcal{A} \setminus \mathcal{A}')\theta + (\mathcal{B} \setminus \mathcal{B}')\theta + \mathcal{D})\theta' = (\mathcal{A}'\theta + (\mathcal{A} \setminus \mathcal{A}')\theta + (\mathcal{B}\theta \setminus \mathcal{B}'\theta) + \mathcal{D})\theta' = \mathcal{B}\theta\theta' + ((\mathcal{A} \setminus \mathcal{A}')\theta\theta' + \mathcal{D}\theta') \in [\![I]\!]_\Sigma$;
- if $\Delta = \forall x. G, \Delta'$ and $I \Vdash_{\Sigma,c} G[c/x], \Delta' \rhd \mathcal{C} \; ; \; \theta$, with $c \notin \Sigma$, then by inductive hypothesis we have that $[\![I]\!] \models_{\Sigma,c} G[c/x]\theta\theta', \Delta'\theta\theta' \rhd \mathcal{C}'\theta'$ for every substitution θ' and fact $\mathcal{C}' \succcurlyeq \mathcal{C}\theta$ (where θ' and $\mathcal{C}'$ are defined over Σ, c). Assuming that the variable x is not in the domain of $\theta\theta'$ (it is always possible to rename the universally quantified variable x in $\forall x. G$), we have that $[\![I]\!] \models_{\Sigma,c} G\theta\theta'[c/x], \Delta'\theta\theta' \rhd \mathcal{C}'\theta'$, and, by definition of the judgment, we get $[\![I]\!] \models_\Sigma \forall x.(G\theta\theta'), \Delta'\theta\theta' \rhd \mathcal{C}'\theta'$, i.e. $[\![I]\!] \models_\Sigma (\forall x. G, \Delta')\theta\theta' \rhd \mathcal{C}'\theta'$, where we can restrict θ' and $\mathcal{C}'$ to be defined over Σ, with $\mathcal{C}' \succcurlyeq \mathcal{C}\theta$;
- if $\Delta = G_1 \parr G_2, \Delta'$ and $I \Vdash_\Sigma G_1, G_2, \Delta' \rhd \mathcal{C} \; ; \; \theta$, then by inductive hypothesis we have that $[\![I]\!] \models_\Sigma (G_1, G_2, \Delta')\theta\theta' \rhd \mathcal{C}'\theta'$, for every substitution θ' and fact $\mathcal{C}' \succcurlyeq \mathcal{C}\theta$. Therefore, $[\![I]\!] \models_\Sigma G_1\theta\theta', G_2\theta\theta', \Delta'\theta\theta') \rhd \mathcal{C}'\theta'$, and, by definition of the judgment, we get $[\![I]\!] \models_\Sigma (G_1 \parr G_2, \Delta)\theta\theta' \rhd \mathcal{C}'\theta'$.

ii) By induction on the derivation of $[\![I]\!] \models_\Sigma \Delta\theta \rhd \mathcal{C}$.

- If $\Delta = \top, \Delta'$, take $\mathcal{C}' = \epsilon$, $\theta' = \epsilon$, and $\sigma = \theta$;
- suppose $[\![I]\!] \models_\Sigma \mathcal{A}\theta \rhd \mathcal{C}$ and $\mathcal{A}\theta + \mathcal{C} \in [\![I]\!]_\Sigma = Up_\Sigma(Inst_\Sigma(I))$. Then there exist $\mathcal{B} \in I$, a fact $\mathcal{D}$ and a substitution τ (defined on Σ) s.t. $\mathcal{A}\theta + \mathcal{C} = \mathcal{B}\tau + \mathcal{D}$. We can safely assume, thanks to the substitution τ, that $\mathcal{B}$ is a *variant* of an element in I. Also, we can assume that $Dom(\tau) \subseteq FV(\mathcal{B})$ and $Dom(\theta) \cap Dom(\tau) = \emptyset$. Now, take the substitution θ'' s.t. $Dom(\theta'') = (Dom(\theta) \cap FV(\mathcal{A})) \cup Dom(\tau)$, $\theta''_{|Dom(\theta) \cap FV(\mathcal{A})} = \theta_{|Dom(\theta) \cap FV(\mathcal{A})}$, and $\theta''_{|Dom(\tau)} = \tau$. We have that $\mathcal{A}\theta'' + \mathcal{C} = \mathcal{B}\theta'' + \mathcal{D}$. Let $\mathcal{A}' \preccurlyeq \mathcal{A}$ and $\mathcal{B}' \preccurlyeq \mathcal{B}$ be two *maximal* submultisets s.t. $\mathcal{A}'\theta'' = \mathcal{B}'\theta''$, $\rho = m.g.u.(\mathcal{A}', \mathcal{B}')$, and

$\theta' = \rho_{|FV(\mathcal{A}) \cup FV(\mathcal{B}\setminus\mathcal{B}')}$. By definition of the $\Vdash_\Sigma$ judgment, I have that $I \Vdash_\Sigma \mathcal{A} \rhd \mathcal{C}' ; \theta'$, where $\mathcal{C}' = \mathcal{B} \setminus \mathcal{B}'$. As θ'' is a unifier for $\mathcal{A}'$,$\mathcal{B}'$, while $\rho = m.g.u.(\mathcal{A}', \mathcal{B}')$, I have that there exists a substitution σ s.t. $\theta'' = \rho \circ \sigma$. Therefore, $\theta_{|FV(\mathcal{A})} = \theta''_{|FV(\mathcal{A})} = (\rho \circ \sigma)_{|FV(\mathcal{A})} = (\rho_{|(FV(\mathcal{A}) \cup FV(\mathcal{B}\setminus\mathcal{B}'))} \circ \sigma)_{|FV(\mathcal{A})} = (\theta' \circ \sigma)_{|FV(\mathcal{A})}$, as required. We also have that $\mathcal{A}\theta'' + \mathcal{C} = \mathcal{B}\theta'' + \mathcal{D}$, i.e., $\mathcal{A}'\theta'' + (\mathcal{A}\setminus\mathcal{A}')\theta'' + \mathcal{C} = \mathcal{B}'\theta'' + (\mathcal{B}\setminus\mathcal{B}')\theta'' + \mathcal{D}$, i.e. $(\mathcal{A}\setminus\mathcal{A}')\theta'' + \mathcal{C} = (\mathcal{B}\setminus\mathcal{B}')\theta'' + \mathcal{D}$. By this equality and maximality of $\mathcal{A}'$ and $\mathcal{B}'$, we get that necessarily $(\mathcal{B}\setminus\mathcal{B}')\theta'' \preccurlyeq \mathcal{C}$. Therefore, $\mathcal{C}'\theta'\sigma = (\mathcal{B} \setminus \mathcal{B}')\theta'\sigma = (\mathcal{B} \setminus \mathcal{B}')\rho\sigma = (\mathcal{B} \setminus \mathcal{B}')\theta'' \preccurlyeq \mathcal{C}$, as required;

- if $\Delta = \forall x. G, \Delta'$ and $[\![I]\!] \models_{\Sigma,c} (G[c/x], \Delta')\theta \rhd \mathcal{C}$, with $c \notin \Sigma$, then by inductive hypothesis there exist a fact $\mathcal{C}'$, and substitutions θ' and σ (defined over Σ, c) s.t. $I \Vdash_{\Sigma,c} G[c/x], \Delta' \rhd \mathcal{C}' ; \theta'$, $\theta_{|FV(G[c/x],\Delta')} = (\theta' \circ \sigma)_{|FV(G[c/x],\Delta')}$, and $\mathcal{C}'\theta'\sigma \preccurlyeq \mathcal{C}$. By definition of the $\Vdash_\Sigma$ judgment, I get that $I \Vdash_\Sigma \forall x. G, \Delta' \rhd \mathcal{C}' ; \theta'$. The conclusion follows (remember that we require $\mathcal{C}'$, θ' and σ to be defined over Σ) by the following crucial observations: $Dom(\theta') \subseteq (FV(G[c/x], \Delta') \cup FV(\mathcal{C}'))$ by Lemma 3; θ' does not map variables in $G[c/x], \Delta'$ to the eigenvariable c (otherwise also θ would); θ' does not map variables in $\mathcal{C}'$ to c and $\mathcal{C}'$ itself does not contain c (otherwise $\mathcal{C}'\theta'\sigma \preccurlyeq \mathcal{C}$ would not hold); I can safely assume that $Dom(\sigma)$ does not contain variables mapped to c, because such mappings do not affect the properties that σ must satisfy;
- if $\Delta = G_1 \mathbin{⅋} G_2, \Delta'$ the conclusion follows by a straightforward application of the inductive hypothesis.

$\square$

The satisfiability judgment $\Vdash_\Sigma$ also satisfies the following properties.

Lemma 5 *For any interpretations I_1, I_2, ..., context Δ, fact $\mathcal{C}$, and substitution θ,*

i) if $I_1 \sqsubseteq I_2$ and $I_1 \Vdash_\Sigma \Delta \rhd \mathcal{C} ; \theta$ then there exist a fact $\mathcal{C}'$, and substitutions θ' and σ s.t. $I_2 \Vdash_\Sigma \Delta \rhd \mathcal{C}' ; \theta'$, $\theta_{|FV(\Delta)} = (\theta' \circ \sigma)_{|FV(\Delta)}$, $\mathcal{C}'\theta'\sigma \preccurlyeq \mathcal{C}\theta$;

ii) if $I_1 \sqsubseteq I_2 \sqsubseteq \ldots$ and $\bigsqcup_{i=1}^{\infty} I_i \Vdash_\Sigma \Delta \rhd \mathcal{C} ; \theta$ then there exist $k \in \mathbf{N}$, a fact $\mathcal{C}'$, and substitutions θ' and σ s.t. $I_k \Vdash_\Sigma \Delta \rhd \mathcal{C}' ; \theta'$, $\theta_{|FV(\Delta)} = (\theta' \circ \sigma)_{|FV(\Delta)}$, $\mathcal{C}'\theta'\sigma \preccurlyeq \mathcal{C}\theta$.

Proof.

i) Suppose $I \Vdash_\Sigma \Delta \rhd \mathcal{C} ; \theta$ and $I \sqsubseteq J$. By Lemma 4 i), $[\![I]\!] \models_\Sigma \Delta\theta \rhd \mathcal{C}\theta$. By Lemma 2 i), $[\![J]\!] \models_\Sigma \Delta\theta \rhd \mathcal{C}\theta$. The conclusion then follows from Lemma 5 ii);

ii) Suppose $\bigsqcup_{i=1}^{\infty} I_i \Vdash_\Sigma \Delta \ \rhd \ \mathcal{C} \ ; \ \theta$ and $I_1 \sqsubseteq I_2 \sqsubseteq \ldots$. By Lemma 4 i), $[\![\bigsqcup_{i=1}^{\infty} I_i]\!] \models_\Sigma \Delta\theta \ \rhd \ \mathcal{C}\theta$, i.e. $\bigsqcup_{i=1}^{\infty}[\![I_i]\!] \models_\Sigma \Delta\theta \ \rhd \ \mathcal{C}\theta$. By Lemma 2 ii), there exists $k \in \mathbf{N}$ s.t. $[\![I_k]\!] \models_\Sigma \Delta\theta \ \rhd \ \mathcal{C}\theta$. The conclusion then follows from Lemma 5 ii).

□

We are now ready to define the abstract fixpoint operator $S_P : \mathcal{I} \to \mathcal{I}$. We will proceed in two steps. We will first define an operator working over interpretations (i.e. elements of $\mathcal{P}(HB(P))$). With a little bit of overloading, we will call the operator with the same name, i.e., S_P. This operator should satisfy the equation $[\![S_P(I)]\!] = T_P([\![I]\!])$ for every interpretation I. This property ensures soundness and completeness of the *symbolic* representation.

After defining the operator over $\mathcal{P}(HB(P))$, we will lift it to our abstract domain $\mathcal{I}$ consisting of the equivalence classes of elements of $\mathcal{P}(HB(P))$ w.r.t. the relation $\simeq$ defined in Definition 5.5. Formally, we first introduce the following definition.

Definition 5.8 (Symbolic Fixpoint Operator S_P) *Given a program P and an interpretation I, the symbolic fixpoint operator S_P is defined as follows:*

$$S_P(I) = \{(\widehat{H} + \mathcal{C})\theta \mid \textit{there exists } (H \multimap G) \in P \textit{ (variant)} \\ \textit{s.t. } I \Vdash_{\Sigma_P} G \ \rhd \ \mathcal{C} \ ; \ \theta\}.$$

Note that the S_P operator is defined using the judgment $\Vdash_{\Sigma_P}$. The following property shows that S_P is sound and complete w.r.t T_P.

Proposition 5 *For every program P and interpretation I, $[\![S_P(I)]\!] = T_P([\![I]\!])$.*

Proof. "$[\![S_P(I)]\!] \subseteq T_P([\![I]\!])$". We prove that for every $\Sigma \in Sig_P$, $[\![S_P(I)]\!]_\Sigma \subseteq (T_P([\![I]\!]))_\Sigma$. Suppose $(\widehat{H}+\mathcal{C})\theta \in S_P(I)$, with $H \multimap G$ variant of a clause in P and $I \Vdash_\Sigma G \ \rhd \ \mathcal{C} \ ; \ \theta$. Suppose also that $\mathcal{A} = ((\widehat{H} + \mathcal{C})\theta + \mathcal{D})\theta' \in Inst_\Sigma(Up_\Sigma(I)) = [\![Sp(I)]\!]$. By Lemma 4 i), $[\![I]\!] \models_\Sigma G\theta\theta' \ \rhd \ \mathcal{C}'\theta'$ for every fact $\mathcal{C}' \succcurlyeq \mathcal{C}\theta$, i.e. $[\![I]\!] \models_\Sigma G\theta\theta' \ \rhd \ \mathcal{C}\theta\theta' + \mathcal{D}\theta'$ for every fact $\mathcal{D}$. Therefore, by T_P definition, I have $\widehat{H}\theta\theta' + \mathcal{C}\theta\theta' + \mathcal{D}\theta' \in T_P([\![I]\!])$, i.e. $\mathcal{A} \in T_P([\![I]\!])$.
"$T_P([\![I]\!]) \subseteq [\![S_P(I)]\!]$". We prove that for every $\Sigma \in Sig_P$, $(T_P([\![I]\!]))_\Sigma \subseteq [\![S_P(I)_\Sigma]\!]$. Suppose $\mathcal{A} \in (T_P([\![I]\!]))_\Sigma$. By definition of T_P, there exist a variant of a clause $H \multimap G$ in P, a fact $\mathcal{C}$ and a substitution θ (defined over Σ) s.t. $\mathcal{A} = \widehat{H}\theta + \mathcal{C}$ and $[\![I]\!] \models_\Sigma G\theta \ \rhd \ \mathcal{C}$. By Lemma 4 ii), there exist a fact $\mathcal{C}'$, and substitutions θ' and σ (defined over Σ) s.t. $I \Vdash_\Sigma G \ \rhd \ \mathcal{C}' \ ; \ \theta'$, $\theta_{|FV(G)} = (\theta' \circ \sigma)_{|FV(G)}$, and $\mathcal{C}'\theta'\sigma \preccurlyeq \mathcal{C}$. Therefore, by S_P definition, $(\widehat{H} + \mathcal{C}')\theta' \in S_P(I)$. Now, $\mathcal{A} = \widehat{H}\theta + \mathcal{C} = \widehat{H}\theta'\sigma + \mathcal{C}$ (note in fact that by hypothesis $\theta'\sigma$ and θ coincide for variables in G, and are not defined on variables in H which do not appear in G because $H \multimap G$ is a variant). Therefore, we have that $\widehat{H}\theta'\sigma + \mathcal{C} \succcurlyeq \widehat{H}\theta'\sigma + \mathcal{C}'\theta'\sigma = ((\widehat{H} + \mathcal{C}')\theta')\sigma \in [\![(\widehat{H} + \mathcal{C}')\theta']\!] \subseteq [\![S_P(I)]\!]$. □

Furthermore, the following corollary holds.

Corollary 1 *For every program P and interpretations I and J, if $I \simeq J$ then $S_P(I) \simeq S_P(J)$.*

Proof. If $I \simeq J$, i.e. $[\![I]\!] = [\![J]\!]$, we have that $T_P([\![I]\!]) = T_P([\![J]\!])$, and, by Proposition 5, $[\![S_P(I)]\!] = [\![S_P(J)]\!]$, i.e. $S_P(I) \simeq S_P(J)$. $\square$

The previous Corollary allows us to safely lift the definition of S_P from the lattice $\langle \mathcal{P}(HB(P)), \subseteq \rangle$ to $\langle \mathcal{I}, \sqsubseteq \rangle$. Formally, we define the abstract fixpoint operator as follows:

Definition 5.9 (Abstract fixpoint operator S_P) *Given a program P and an equivalence class $[I]_\simeq$ of $\mathcal{I}$, the abstract fixpoint operator S_P is defined as follows:*

$$S_P([I]_\simeq) = [S_P(I)]_\simeq$$

where $S_P(I)$ is defined in Definition 5.8.

For the sake of simplicity, in the following we will often use I to denote its class $[I]_\simeq$, and we will simply use the term *(abstract) interpretation* to refer to an equivalence class, i.e. an element of $\mathcal{I}$. The abstract fixpoint operator S_P satisfies the following property.

Proposition 6 *For every program P, the abstract fixpoint operator S_P is monotonic and continuous over the lattice $\langle \mathcal{I}, \sqsubseteq \rangle$.*

Proof. "Monotonicity". We prove that if $I \sqsubseteq J$, then $S_P(I) \sqsubseteq S_P(J)$, i.e. $[\![S_P(I)]\!] \subseteq [\![S_P(J)]\!]$. To prove this latter condition, we will use the characterization given by Proposition 4. Suppose $\mathcal{A} = (\widehat{H} + \mathcal{C})\theta \in S_P(I)$, with $H \multimapinv G$ variant of a clause in P and $I \Vdash_{\Sigma_P} G \rhd \mathcal{C}\ ;\ \theta$. By Lemma 5 i), there exist a fact $\mathcal{C}'$, and substitutions θ' and σ (note that they are defined over Σ_P) s.t. $J \Vdash_{\Sigma_P} G \rhd \mathcal{C}'\ ;\ \theta'$, $\theta_{|FV(G)} = (\theta' \circ \sigma)_{|FV(G)}$, $\mathcal{C}'\theta'\sigma \preccurlyeq \mathcal{C}\theta$. Let $\mathcal{C}\theta = \mathcal{C}'\theta'\sigma + \mathcal{D}$, with $\mathcal{D}$ a fact defined over Σ_P. By S_P definition, $\mathcal{B} = (\widehat{H} + \mathcal{C}')\theta' \in S_P(J)$. Now, $\mathcal{A} = (\widehat{H} + \mathcal{C})\theta = \widehat{H}\theta + \mathcal{C}\theta = \widehat{H}\theta'\sigma + \mathcal{C}'\theta'\sigma + \mathcal{D}$ (note in fact that by hypothesis $\theta'\sigma$ and θ coincide for variables in G, and are not defined on variables in H which do not appear in G because $H \multimapinv G$ is a variant). Therefore, we have that $\mathcal{A} = \widehat{H}\theta'\sigma + \mathcal{C}'\theta'\sigma + \mathcal{D} = \mathcal{B}\sigma + \mathcal{D}$.
"Continuity". We show that S_P is finitary, i.e. if $I_1 \sqsubseteq I_2 \sqsubseteq \ldots$, then $S_P(\bigsqcup_{i=1}^{\infty} I_i) \sqsubseteq \bigsqcup_{i=1}^{\infty} S_P(I_i)$, i.e. $[\![S_P(\bigsqcup_{i=1}^{\infty} I_i)]\!] \subseteq [\![\bigsqcup_{i=1}^{\infty} S_P(I_i)]\!]$. Again, we will use the characterization given by Proposition 4. Suppose $\mathcal{A} = (\widehat{H} + \mathcal{C})\theta \in S_P(\bigsqcup_{i=1}^{\infty} I_i)$, with $H \multimapinv G$ variant of a clause in P and $\bigsqcup_{i=1}^{\infty} I_1 \Vdash_{\Sigma_P} G \rhd \mathcal{C}\ ;\ \theta$. By Lemma 5 ii), there exist $k \in \mathbf{N}$, a fact $\mathcal{C}'$, and substitutions θ' and σ (note that they are defined over Σ_P) s.t. $I_k \Vdash_{\Sigma_P} G \rhd \mathcal{C}'\ ;\ \theta'$, $\theta_{|FV(G)} = (\theta' \circ \sigma)_{|FV(G)}$, $\mathcal{C}'\theta'\sigma \preccurlyeq \mathcal{C}\theta$. Let $\mathcal{C}\theta = \mathcal{C}'\theta'\sigma + \mathcal{D}$, with $\mathcal{D}$ a fact defined over Σ_P. By S_P definition, $\mathcal{B} = (\widehat{H} + \mathcal{C}')\theta' \in S_P(I_k)$. Exactly as above, we prove that $\mathcal{A} = (\widehat{H} + \mathcal{C})\theta = \widehat{H}\theta'\sigma + \mathcal{C}'\theta'\sigma + \mathcal{D} = \mathcal{B}\sigma + \mathcal{D}$. $\square$

Corollary 2 *For every program P, $[\![lfp(S_P)]\!] = lfp(T_P)$.*

Let $SymbF(P) = lfp(S_P)$, then we have the following main Theorem.

Theorem 2 *For every program P, $O(P) = F(P) = [\![SymbF(P)]\!]_{\Sigma_P}$.*

The previous results give us an algorithm to compute the operational and fixpoint semantics of a program P via the fixpoint operator S_P.

Example 2 *Let's consider the program P given by*

$$\begin{array}{ll} 1\cdot & r(Y) \multimapinv q(f(Y)) \\ 2\cdot & s(Z) \multimapinv \forall x.p(X) \parr t(Z) \\ 3\cdot & p(X) \parr q(X) \multimapinv \top \end{array}$$

From the third clause, and using the first rule for $\Vdash_{\Sigma_P}$, we get that

$$S_P{\uparrow_1} = S_P(\emptyset) = \{[p(X), q(X)]_{\simeq}\}\cdot$$

We can now apply clauses 2. and 3. to the interpretation $I = \{[p(X), q(X)]_{\simeq}\}$ (remember that $S_P([I]_{\simeq}) = [S_P(I)]_{\simeq}$). From the first clause (see Example 1) we have $I \Vdash_{\Sigma_P} q(f(Y')) \rhd p(X')\ ;\ [X' \leadsto f(Y')]$. Therefore we have that $[(r(Y'), p(X'))[X' \leadsto f(Y')]]_{\simeq} = [r(Y'), p(f(Y'))]_{\simeq} \in S_P{\uparrow_2}$. As the reader can verify (see discussion in Example 1), clause 2. does not yield any further element, therefore we can assume

$$S_P{\uparrow_2} = \{[p(X), q(X)]_{\simeq}, [r(Y'), p(f(Y'))]_{\simeq}\},$$

which turns out to be the fixpoint of S_P. Note that $F(P)$ is defined to be $[\![lfp(S_P)]\!]_{\Sigma_P}$, therefore it includes for instance the elements $p(f(X'')), q(f(X''))$ and $p(f(X'')), q(f(X'')), s(Z)$. $\square$

6 Conclusions

In this paper we have defined a fixpoint semantics for a first-order formulation of a subset of LinLog [And92]. Possible applications of this framework include for instance studying verification problems for Petri Net-like models of concurrent systems, as discussed in [BDM01b], and studying observable properties of programs [BGLM94, GDL95]. This work extends our previous work [BDM01b], which was limited to the propositional case, along the directions explored in [BDM01a]. With respect to [BDM01a], the current work deals with first-order variables using a traditional approach based on substitutions and most general unifiers, rather than using (unification) constraints. The main novelty with respect to [BDM01a] is the treatment of the universal quantifier $\forall$. Among the applications of this extended framework, we are currently studying verification of security protocols, which, as shown in [CDL+99], can be specified in a fragment of linear logic corresponding to multiset rewriting plus existential quantification. This latter fragment directly corresponds, modulo a direct translation between *dual* connectives of linear logic, to the logical language presented in this paper.

Verification of protocols is performed by means of a bottom-up algorithm which follows a strategy related to the so-called *backward reachability* relation in model checking, as explained in [BDM01b]. The results presented in this paper are intended as a formal justification of the correctness of the algorithm in presence of universal quantification. Future work also includes extending the logic underlying the current framework. On the one hand, it is possible to allow other connectives in the right hand side of sequents, as shown for instance in [BDM01a] in the case of additive conjunction $\&$. On the other hand, a more involved extension would require considering dynamically modifiable *programs* as, e.g., in Lolli [HM94] or Lygon [HP94]. Also, by extending the current management of substitutions, it would be possible to reason about observables like *computed answer substitutions*, as in the so-called *S-semantics* [FLMP93, BGLM94]. We also plan, along the lines of [BDM01a], to allow constraints in program definitions. For instance, constraints over real numbers are commonly used in security protocols specifications to deal with *timestamps*.

Acknowledgments

The author would like to thank Frank Pfenning, Iliano Cervesato and Giorgio Delzanno for fruitful discussions.

References

[And92] J.-M. Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[AP91] J.-M. Andreoli and R. Pareschi. Linear Objects: Logical Processes with Built-In Inheritance. *New Generation Computing*, 9(3-4):445–473, 1991.

[BDM01a] M. Bozzano, G. Delzanno, and M. Martelli. An Effective Bottom-Up Semantics for First Order Linear Logic Programs. In *Proceedings of Fifth International Symposium on Functional and Logic Programming (FLOPS'01)*, 2001.

[BDM01b] M. Bozzano, G. Delzanno, and M. Martelli. An Effective Fixpoint Semantics for Linear Logic Programs, 2001. To appear in *Theory and Practice of Logic Programming*.

[BGLM94] A. Bossi, M. Gabbrielli, G. Levi, and M. Martelli. The s-Semantics Approach: Theory and Applications. *Journal of Logic Programming*, 19-20:149–197, 1994.

[CDL+99] I. Cervesato, N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. A Meta-notation for Protocol Analysis. In R. Gorrieri, editor, *12th IEEE Computer Security Foundations Workshop (CSFW'99)*, pages 55–69, 1999.

[FLMP93] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. A Model-Theoretic Reconstruction of the Operational Semantics of Logic Programs. *Information and Computation*, 103(1):86–113, 1993.

[GDL95] M. Gabbrielli, M. G. Dore, and G. Levi. Observable semantics for Constraint Logic Programs. *Journal of Logic and Computation*, 5(2):133–171, 1995.

[Gir87] J.Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1:1–102, 1987.

[HM94] J. Hodas and D. Miller. Logic Programming in a Fragment of Intuitionistic Linear Logic. *Information and Computation*, 110(2):327–365, 1994.

[HP94] J. Harland and D. J. Pym. A Uniform Proof-Theoretic Investigation of Linear Logic Programming. *Journal of Logic and Computation*, 4(2):175–207, 1994.

[KY95] N. Kobayashi and A. Yonezawa. Asynchronous Communication Model based on Linear Logic. *Formal Aspects of Computing*, 7(2):113–149, 1995.

[Mil95] D. Miller. A Survey of Linear Logic Programming. *Computational Logic: The Newsletter of the European Network of Excellence in Computational Logic*, 2(2):63–67, 1995.

[Mil96] D. Miller. Forum: A Multiple-Conclusion Specification Logic. *Theoretical Computer Science*, 165(1):201–232, 1996.