A Strong Logic Programming View for Static Embedded Implications *

R. Arruabarrena, P. Lucio, and M. Navarro

Dpto de L.S.I., Facultad de Informática, Paseo Manuel de Lardizabal, 1, Apdo 649, 20080-San Sebastián, SPAIN. Tel: +34 (9)43 448000, Fax: +34 (9)43 219306, e-mail: marisa@si.ehu.es.

Abstract. A *strong ($\mathcal{L}$) logic programming language* ([14, 15]) is given by two subclasses of formulas (*programs* and *goals*) of the underlying logic $\mathcal{L}$, provided that: firstly, any program P (viewed as a $\mathcal{L}$-theory) has a canonical model M_P which is initial in the category of all its $\mathcal{L}$-models; secondly, the $\mathcal{L}$-satisfaction of a goal G in M_P is equivalent to the $\mathcal{L}$-derivability of G from P, and finally, there exists an effective (computable) proof-subcalculus of the $\mathcal{L}$-calculus which works out for derivation of goals from programs. In this sense, Horn clauses constitute a strong (first-order) logic programming language. Following the methodology suggested in [15] for designing logic programming languages, an extension of Horn clauses should be made by extending its underlying first-order logic to a richer logic which supports a strong axiomatization of the extended logic programming language. A well-known approach for extending Horn clauses with *embedded implications* is the static scope programming language presented in [8]. In this paper we show that such language can be seen as a strong $\mathcal{FO}^{\supset}$ logic programming language, where $\mathcal{FO}^{\supset}$ is a very natural extension of first-order logic with intuitionistic implication. That is, we present a new characterization of the language in [8] which shows that Horn clauses extended with embedded implications, viewed as $\mathcal{FO}^{\supset}$-theories, preserves all the attractive mathematical and computational properties that Horn clauses satisfy as first-order-theories.

1 Introduction

Horn clause programs are theories in first-order logic (namely $\mathcal{FO}$) whose computation relation (between programs and goals) is equivalent to the following relations of $\mathcal{FO}$: logical consequence, derivability and satisfaction in the least Herbrand model of the program. Moreover, the least Herbrand model of a program is initial in the category of all first-order models of the program and it exactly satisfies the goals which are satisfied in every model in this category. In other words, Horn clauses can be seen as a $\mathcal{FO}$ logic programming language, in the strong sense of [14, 15], because its underlying logic $\mathcal{FO}$ has attractive (model-theoretic) mathematical and (proof-theoretic) computational properties (for programs and goals). This idea was formalized in [14, 15] where the notion of a *strong logic programming language* is defined as a restriction of an underlying logic satisfying good properties. This means, once fixed an underlying logic, setting which subclasses of its formulas correspond to the classes of *programs* and *queries or goals*, respectively. The underlying logic, for these subclasses, must satisfy three properties: mathematical semantics, goal completeness and operational semantics. The mathematical semantics property requires that any program has a canonical

* This work has been partially supported by the CICYT-project TIC95-1016-C02-02.

W. Thomas (Ed.): FOSSACS'99, LNCS 1578, pp. 56-72, 1999.

model, which is initial in the class of all models of the program (seen as a theory in the underlying logic). Goal completeness means that logical satisfaction of goals in the initial model is equivalent to the derivability relation (of the logic) restricted to programs and goals. The operational semantics property means the existence of an effective (computable) proof-subcalculus of the calculus (of the logic) for deriving goals from programs. We believe that this view of axiomatizing a logic programming language inside an underlying logic has many advantages. On one hand, it allows one to separate general logical features from programming language features. On the other hand, a useful way to analyse, compare and integrate different programming features is to axiomatize them into a common underlying logic.

Attempts to extend Horn clause logic programming (e.g. with modules, higher-order, data abstraction, etc.) should be done by preserving (as much as possible) the above-mentioned mathematical and computational properties. Since Horn clause logic is the greatest fragment of $\mathcal{FO}$ admitting initial models, concrete extensions could require to change (by restricting or enriching) the underlying logic $\mathcal{FO}$. Many approaches are concerned with extending Horn clauses with some features for program structuring that can be seen as a form of modularity in logic programming (see for instance [2] for a survey). Some of them consider the extension of Horn clauses with implication goals of the form $D \supset G$, called *blocks*, where D can be seen as a *local set of clauses* (or *module*) for proving the goal G. This approach yields to different extensions of Horn clause programming depending on the given semantics to such blocks. A first basic distinction is between *closed blocks*: G can be proved only using local clauses from D, and *open blocks*: G can be proved using D and also the external environment. Therefore, open blocks require *scope rules* to fix the interplay between the predicate definitions inside a module D and those in the environment. In general, dealing with open blocks, a module can extend the definition of a predicate already defined in the enviroment. Hence, different definitions of the same predicate could have to be considered, depending on the collection of modules corresponding to different goals. There are mainly two scope rules, named *static* and *dynamic*, allowing this kind of extension of predicate definitions. In the dynamic approach the set of modules taking part in the resolution of a goal G can only be determined from the sequence of goals generated until G. However, in the static case this set of modules can be determined (for each goal) statically from the block structure of the program. Different proposals of logic programming languages for open blocks with dynamic scope have been presented and studied in several papers (e.g.[4–6, 16–18]). The static scope approach has been mainly studied in [8, 7]. In [2, 7] both different approaches are compared. Some other works (e.g. [19, 20]) treat open blocks with different scope rules avoiding this kind of predicate extension.

In [16] Miller proves that the proof-theoretic semantics for its dynamic scope programming language is based on intuitionistic logic, and in [2] it is shown that the Miller's canonical model for a program is indeed an intuitionistic model of this program. However, for the static scope programming language introduced in [8], neither first-order logic nor intuitionistic logic can be used for this purpose. Following the methodology suggested in [15] for designing logic programming languages, the extension of Horn clauses with intuitionistic implication should be strongly axiomatized in a logic which integrates $\mathcal{FO}$ and intuitionistic implication. In this paper we introduce a complete logic called $\mathcal{FO}^{\supset}$, which is a very natural extension of $\mathcal{FO}$ with intuitionistic implication. We give a new characterization of the well-known semantics for the static scope programming language presented in [8]. This characterization strongly axiomatizes the

logic programming language inside $\mathcal{FO}^{\supset}$ logic, showing that it satisfies all the desirable properties.
The paper is organized as follows: In Section 2 we introduce the formalization of [14, 15] for a strong logic programming language which is the methodological basis for our work. In Section 3 we give a short introduction to the underlying logic $\mathcal{FO}^{\supset}$ giving the necessary notions and results for the rest of the paper. In Section 4 we develop the $\mathcal{FO}^{\supset}$ strong axiomatization of the static scope programming language. We conclude, in Section 5, by summarizing the presented results and related work.

2 Preliminaries

In this section, we introduce the notions of *logic* and *strong logic programming language*, following [14, 15].
The notion of a *logic* is obtained by combining an *entailment system* (formalizing the proof-theoretical component of a logic) with an *institution* (formalizing the model-theoretical component) such that a *soundness condition* relating provability and satisfaction holds. An *entailment system* is a triple $(\underline{\mathsf{Sign}}, \mathsf{sen}, \vdash)$ with $\underline{\mathsf{Sign}}$ a category of *signatures*, sen a functor associating to each $\Sigma \in \underline{\mathsf{Sign}}$ a set $\mathsf{sen}(\Sigma)$ of Σ-sentences and $\vdash$ a function associating to each $\Sigma \in \underline{\mathsf{Sign}}$ a binary relation $\vdash_\Sigma \subseteq \mathcal{P}(\mathsf{sen}(\Sigma)) \times \mathsf{sen}(\Sigma)$, called Σ-entailment or Σ-derivability, which satisfies the properties of reflexivity, monotonicity, transitivity and $\vdash$-translation (i.e. preservation by signature morphisms). An *institution* is a 4-tuple $(\underline{\mathsf{Sign}}, \mathsf{sen}, \underline{\mathsf{Mod}}, \models)$ with $\underline{\mathsf{Sign}}$ and sen as above; $\underline{\mathsf{Mod}}$ is a functor associating to each $\Sigma \in \underline{\mathsf{Sign}}$ a corresponding category $\underline{\mathsf{Mod}}(\Sigma)$ whose objects are called Σ-structures (or Σ-models) and whose morphisms preserve the interpretation given to signature symbols; and $\models$ is a function associating to each $\Sigma \in \underline{\mathsf{Sign}}$ a binary relation $\models_\Sigma \subseteq \underline{\mathsf{Mod}}(\Sigma) \times \mathsf{sen}(\Sigma)$, called Σ-satisfaction, which satisfies the $\models$-invariance property (i.e. for any $M_2 \in \underline{\mathsf{Mod}}(\Sigma_2)$, $H : \Sigma_1 \to \Sigma_2$, $\varphi \in \mathsf{sen}(\Sigma_1)$: $\underline{\mathsf{Mod}}(H)(M_2) \models_{\Sigma_1} \varphi$ iff $M_2 \models_{\Sigma_2} H(\varphi)$). Given $\Gamma \subseteq \mathsf{sen}(\Sigma)$, $\underline{\mathsf{Mod}}(\Gamma)$ denotes the full subcategory of $\underline{\mathsf{Mod}}(\Sigma)$ determined by the structures $M \in \underline{\mathsf{Mod}}(\Sigma)$ such that $M \models_\Sigma \varphi$ for each $\varphi \in \Gamma$. The satisfaction relation induces a logical consequence relation between sets of sentences and sentences, also denoted $\models$, as follows: $\Gamma \models_\Sigma \varphi$ iff $M \models_\Sigma \varphi$ for each $M \in \underline{\mathsf{Mod}}(\Gamma)$.
A *logic* is given by an entailment system and an institution sharing the same signatures and sentences, such that it holds *soundness* of the derivability relation w.r.t. the logical consequence relation. A *logic* is a 5-tuple $\mathcal{L}=(\underline{\mathsf{Sign}}, \mathsf{sen}, \underline{\mathsf{Mod}}, \vdash, \models)$ such that:

- $(\underline{\mathsf{Sign}}, \mathsf{sen}, \vdash)$ is an entailment system
- $(\underline{\mathsf{Sign}}, \mathsf{sen}, \underline{\mathsf{Mod}}, \models)$ is an institution
- For any $\Sigma \in \underline{\mathsf{Sign}}$, $\Gamma \subseteq \mathsf{sen}(\Sigma)$ and $\varphi \in \mathsf{sen}(\Sigma)$, $\Gamma \vdash_\Sigma \varphi \implies \Gamma \models_\Sigma \varphi$ (*Soundness*).

In addition, there are some other useful properties that a logic could satisfy, like completeness, compactness, etc.

From the axiomatic point of view, a *strong logic programming language* is a 4-tuple $\mathcal{LPL}= (\mathcal{L}, \underline{\mathsf{Sign}}', prog, goal)$ with:

- $\mathcal{L}=(\underline{\mathsf{Sign}}, \mathsf{sen}, \underline{\mathsf{Mod}}, \vdash, \models)$ a logic, namely the *underlying logic* of $\mathcal{LPL}$
- $\underline{\mathsf{Sign}}'$ a subcategory of $\underline{\mathsf{Sign}}$
- $prog$ is a functor associating to each $\Sigma \in \underline{\mathsf{Sign}}'$ a set $prog(\Sigma)$ (of Σ-programs) included in $\mathcal{P}_{fin}(\mathsf{sen}(\Sigma))$
- $goal$ is a functor associating to each $\Sigma \in \underline{\mathsf{Sign}}'$ a set of Σ-goals, $goal(\Sigma) \subseteq \mathsf{sen}(\Sigma)$

such that the following properties are satisfied:

1. *Mathematical Semantics*: Each program $P \in prog(\Sigma)$ has a model M_P which is initial in the category $\underline{\mathsf{Mod}}(P)$ of all models in $\underline{\mathsf{Mod}}(\Sigma)$ satisfying P
2. *Goal Completeness (w.r.t. the initial model)*: For any program $P \in prog(\Sigma)$ and any goal $G \in goal(\Sigma)$, $P \vdash_\Sigma G \iff M_P \models_\Sigma G$
3. *Operational Semantics*: Existence of an effective proof subcalculus for the derivability relation $\vdash_\Sigma$ restricted to $prog(\Sigma) \times goal(\Sigma)$.

3 The Logic $\mathcal{FO}^{\supset}$

In this section we introduce the sound and complete logic $\mathcal{FO}^{\supset}$ which extends classical first-order logic with intuitionistic implication. We present its language, semantical structures, logical consequence relation, derivability relation and some other details which are relevant to understand the rest of the paper. A more detailed presentation of this logic is out of the scope of this paper and it can be found in [11], in particular there it is proved soundness and completeness of $\mathcal{FO}^{\supset}$ logic.

A signature $\Sigma \in \underline{\mathsf{Sign}}$ consists of countable sets FS_Σ of function symbols, and PS_Σ of predicate symbols, with some specific arity for each function and predicate symbol. We also assume a countable set VS_Σ of variable symbols. We denote by T_Σ the set of all well-formed first-order Σ-terms. A term is closed if no variable symbol does occur on it. Well-formed Σ-formulas are built, from atomic ones, using classical connectives $(\neg, \wedge, \vee, \rightarrow)$, intuitionistic implication $(\supset)$, and classical quantifiers $(\forall, \exists)$. Free and bound variables and substitution have the usual definitions. $\mathsf{sen}(\Sigma)$ is the set of Σ-sentences, that is, Σ-formulas with no free variables. We will denote formulas by lowercase Greek letters $\varphi, \psi, \gamma, \chi, \ldots$. The uppercase Greek letters Γ and Φ (probably with sub- and superscripts) will be used as metavariables for *sets* of formulas. Model theory is based on Kripke structures ([21]).

Definition 1. A *Kripke Σ-structure* is a triple $K = (W(K), \preceq, \langle \mathcal{A}_w \rangle_{w \in W(K)})$ where $(W(K), \preceq)$ is a non-empty partially ordered set (of worlds) and each $\mathcal{A}_w$ is a first-order Σ-structure (with universe A_w, over which predicate and function symbols are interpreted) such that for any pair of worlds $v \preceq w$ in $W(K)$:

- $A_v \subseteq A_w$,
- $p^{\mathcal{A}_v} \subseteq p^{\mathcal{A}_w}$, for all $p \in PS_\Sigma$
- $f^{\mathcal{A}_w}(a_1, \ldots, a_n) = f^{\mathcal{A}_v}(a_1, \ldots, a_n)$, for all $a_1, \ldots, a_n \in A_v$ and $f \in FS_\Sigma$. ■

$\underline{\mathsf{Mod}}(\Sigma)$ will denote the category whose objects are Kripke Σ-structures. The morphisms in this category will be given in Definition 6.
We denote by t^w the classical first-order interpretation $t^{\mathcal{A}_w}$ of $t \in T_\Sigma$. Terms interpretation behaves monotonically, that is, for any Kripke-structure K and any pair of worlds $v, w \in W(K)$ such that $v \preceq w$: $t^w = t^v \in A_v \subseteq A_w$. The satisfaction of sentences in worlds is handled by the following *forcing relation*:

Definition 2. Let $K \in \underline{\mathsf{Mod}}(\Sigma)$, the binary *forcing relation* $\Vdash \subseteq W(K) \times \mathsf{sen}(\Sigma)$ is inductively defined as follows:

$w \nVdash_F$
$w \Vdash p(t_1, \ldots, t_n)$ iff $(t_1^w, \ldots, t_n^w) \in p^{\mathcal{A}_w}$
$w \Vdash \neg\varphi$ iff $w \nVdash \varphi$
$w \Vdash \varphi \wedge \psi$ iff $w \Vdash \varphi$ and $w \Vdash \psi$

$w \Vdash \varphi \vee \psi$ iff $w \Vdash \varphi$ or $w \Vdash \psi$
$w \Vdash \varphi \rightarrow \psi$ iff if $w \Vdash \varphi$ then $w \Vdash \psi$
$w \Vdash \varphi \supset \psi$ iff for all $v \in W(K)$ such that $v \succeq w$: if $v \Vdash \varphi$ then $v \Vdash \psi$
$w \Vdash \exists x\varphi$ iff $w \Vdash \varphi(\hat{a}/x)$ for some $a \in A_w$ [1]
$w \Vdash \forall x\varphi$ iff $w \Vdash \varphi(\hat{a}/x)$ for all $a \in A_w$. ■

We will write $w, K \Vdash \varphi$ (instead of $w \Vdash \varphi$) whenever confusion on the structure K may occur. This forcing relation gives a non-intuitionistic semantics to negation, classical implication ($\rightarrow$) and universal quantification; as a consequence, the forcing relation on sentences does not behave monotonically w.r.t. the world ordering. We say that a sentence is *persistent* whenever the forcing relation behaves monotonically for it.

Definition 3. A Σ-sentence φ is *persistent* iff for any $K \in \underline{\mathsf{Mod}}(\Sigma)$ and $w \in W(K)$: if $w \Vdash \varphi$ then $v \Vdash \varphi$ for any $v \in W(K)$ such that $v \succeq w$. ■

Persistent sentences play an important role in the $\mathcal{FO}^{\supset}$-axiomatization of logic programming languages with embedded implications, since there is a subclass of persistent sentences (that can be syntactically delimited) which includes the class of goals.

Proposition 4. *Any atomic sentence is persistent. Any sentence $\varphi \supset \psi$ is persistent. If φ and ψ are persistent sentences, then $\varphi \vee \psi$ and $\varphi \wedge \psi$ are persistent. If $\varphi(\hat{a})$ is a persistent sentence, then $\exists x\varphi$ is persistent.*

Proof. For atoms the property is a trivial consequence of the Kripke structure definition. For intuitionistic implication it is also trivial from forcing relation definition. The other two cases are easily proved, by induction, using the forcing relation definition for $\vee, \wedge$ and $\exists$. ■

The satisfaction relation $\models_{\Sigma} \subseteq \underline{\mathsf{Mod}}(\Sigma) \times \mathsf{sen}(\Sigma)$ requires the sentence to be forced (only) in the minimal worlds of the structure. This satisfaction relation induces the logical consequence relation, denoted by the same symbol $\models_{\Sigma}$.

Definition 5. Let $K \in \underline{\mathsf{Mod}}(\Sigma)$ and $\Gamma \cup \{\varphi\} \subseteq \mathsf{sen}(\Sigma)$. We say that
(a) A world $w \in W(K)$ is *minimal* iff there does not exist $v \in W(K)$ such that $v \preceq w$ and $v \neq w$.
(b) $K \models_{\Sigma} \varphi$ (K satisfies φ) iff $w \Vdash \varphi$ for each minimal world $w \in W(K)$.
(c) $\Gamma \models_{\Sigma} \varphi$ (φ is logical consequence of Γ) iff $K \models_{\Sigma} \Gamma \Rightarrow K \models_{\Sigma} \varphi$, for each $K \in \underline{\mathsf{Mod}}(\Sigma)$. ■

Morphisms in $\underline{\mathsf{Mod}}(\Sigma)$ relate only minimal worlds, with the idea of preserving the satisfaction relation for ground atoms, in the following way:

Definition 6. For $i = 1, 2$, let $K_i = (W(K_i), \preceq_{K_i}, \langle \mathcal{A}^i_w \rangle_{w \in W(K_i)}) \in \underline{\mathsf{Mod}}(\Sigma)$ and let $W(K_i)^{min}$ be the set of minimal worlds in $W(K_i)$. A *morphism* $H : K_1 \rightarrow K_2$ is given by a mapping $\sigma_H : W(K_2)^{min} \rightarrow W(K_1)^{min}$ together with a collection of first-order Σ-homomorphisms $\langle H_w : \mathcal{A}^1_{\sigma_H(w)} \rightarrow \mathcal{A}^2_w \rangle_{w \in W(K_2)^{min}}$. If the mapping σ_H is unique (for instance when K_1 has only one minimal world) then we will identify H directly with its collection of first-order Σ-homomorphisms. ■

[1] The constant symbol $\hat{a}$ stands for the syntactic denotation of a (see e.g.[21]).

Remark 7. We recall that first-order Σ-homomorphisms are mappings that preserve the operations and relations which (respectively) interpret function and predicate symbols. In particular they preserve ground atoms. ■

Actually, the above-defined 4-tuple $\mathcal{FO}^{\supset} = (\underline{\mathsf{Sign}}, \mathsf{sen}, \underline{\mathsf{Mod}}, \models)$ forms an institution. The satisfaction relation is preserved: for each signature morphism $H : \Sigma \to \Sigma'$, each $K' \in \underline{\mathsf{Mod}}(\Sigma')$ and each $\varphi \in \mathsf{sen}(\Sigma)$, it is the case that $\underline{\mathsf{Mod}}(H)(K') \models_{\Sigma} \varphi$ iff $K' \models_{\Sigma'} \mathsf{sen}(H)(\varphi)$, where $\mathsf{sen}(H) : \mathsf{sen}(\Sigma) \to \mathsf{sen}(\Sigma')$ is the translation of sentences induced by H and where $\underline{\mathsf{Mod}}(H) : \underline{\mathsf{Mod}}(\Sigma') \to \underline{\mathsf{Mod}}(\Sigma)$ is the forgetful functor associated to H. This functor applies each Σ'-structure K' into a Σ-structure K with the same ordered set of worlds and it associates each first-order structure ${\mathcal{A}'}_w$ into its forgetful first-order structure $V_H({\mathcal{A}'}_w)$.

Structural Rules

$(Init)\ \Delta \rhd A$ if A is atomic and $A \in \Delta$ $\qquad (_F L)\ \Delta; \Gamma, {}_F; \Delta' \rhd \chi$

$$(Cut)\ \frac{\Delta; \Gamma \rhd \varphi \quad \Delta; \Gamma, \varphi; \Delta' \rhd \chi}{\Delta; \Gamma; \Delta' \rhd \chi} \qquad (R_F)\ \Delta; \Gamma, \varphi, \neg\varphi; \Delta' \rhd {}_F$$

Connective Rules

$$(\neg L)\ \frac{\Delta; \Gamma; \neg\chi \rhd \varphi}{\Delta; \Gamma, \neg\varphi \rhd \chi} \qquad (R\neg)\ \frac{\Delta; \Gamma, \varphi \rhd {}_F}{\Delta; \Gamma \rhd \neg\varphi}$$

$$(\vee L)\ \frac{\Delta; \Gamma, \varphi; \Delta' \rhd \chi \quad \Delta; \Gamma, \psi; \Delta' \rhd \chi}{\Delta; \Gamma, \varphi \vee \psi; \Delta' \rhd \chi} \qquad (R\vee)\ \frac{\Delta \rhd \varphi}{\Delta \rhd \varphi \vee \psi} \quad \frac{\Delta \rhd \psi}{\Delta \rhd \varphi \vee \psi}$$

$$(\wedge L)\ \frac{\Delta; \Gamma, \varphi, \psi; \Delta' \rhd \chi}{\Delta; \Gamma, \varphi \wedge \psi; \Delta' \rhd \chi} \qquad (R\wedge)\ \frac{\Delta \rhd \varphi \quad \Delta \rhd \psi}{\Delta \rhd \varphi \wedge \psi}$$

$$(\to L)\ \frac{\Delta; \Gamma \rhd \varphi \quad \Delta; \Gamma, \psi; \Delta' \rhd \chi}{\Delta; \Gamma, \varphi \to \psi; \Delta' \rhd \chi} \qquad (R\to)\ \frac{\Delta; \Gamma, \varphi \rhd \psi}{\Delta; \Gamma \rhd \varphi \to \psi}$$

$$(\supset L)\ \frac{\Delta; \Gamma; \Delta'; \Gamma' \rhd \varphi \quad \Delta; \Gamma; \Delta'; \Gamma', \psi; \Delta'' \rhd \chi}{\Delta; \Gamma, \varphi \supset \psi; \Delta'; \Gamma'; \Delta'' \rhd \chi} \qquad (R\supset)\ \frac{\Delta; \{\varphi\} \rhd \psi}{\Delta \rhd \varphi \supset \psi}$$

Quantifier Rules

$$(\exists L)\ \frac{\Delta; \Gamma, \varphi(c/x); \Delta' \rhd \chi}{\Delta; \Gamma, \exists x\varphi; \Delta' \rhd \chi} \qquad (R\exists)\ \frac{\Delta \rhd \varphi(t/x)}{\Delta \rhd \exists x\varphi}$$

$$(\forall L)\ \frac{\Delta; \Gamma, \varphi(t/x); \Delta' \rhd \chi}{\Delta; \Gamma, \forall x\varphi; \Delta' \rhd \chi} \qquad (R\forall)\ \frac{\Delta \rhd \varphi(c/x)}{\Delta \rhd \forall x\varphi}$$

Fig. 1. A sound and complete sequent calculus for $\mathcal{FO}^{\supset}$.

We will complete the definition of $\mathcal{FO}^{\supset}$ logic by giving a derivability relation $\vdash_{\Sigma} \subseteq \mathcal{P}(\mathsf{sen}(\Sigma)) \times \mathsf{sen}(\Sigma)$ in terms of *sequent calculus* proofs. The original Gentzen's notion considers sequents $\Gamma \rhd \Phi$ whose *antecedent* Γ and *consequent* Φ are both finite (possibly empty) sequences of formulas. In $\mathcal{FO}^{\supset}$ logic, to deal with classical and intuitionistic implications inside the same logic, it is essential to introduce extra structure in sequent antecedents. That is, to achieve soundness and completeness for $\mathcal{FO}^{\supset}$ logic, we consider sequents consisting of pairs $\Delta \rhd \varphi$ where the antecedent Δ is a (finite) sequence of (finite) sets of formulas, and the consequent φ is (like in intuitionistic logic) a single formula. Uppercase Greek letters Δ, Δ', Δ'', ... will be used as metavariables for *sequences of sets* of formulas. In order to simplify sequent notation: the semicolon sign $(;)$ will represent the infix operation for concatenation of sequences, $\Gamma \cup \{\varphi\}$ will be abbreviated by Γ, φ; and a set Γ will be identified with the sequence consisting of this

unique set. On these bases, we present a sound and complete sequent calculus for the logic $\mathcal{FO}^{\supset}$ in Figure 1 where (in quantifier rules) c stands for a new fresh constant symbol and t stands for a closed term.
Notice that every rule in the calculus of Fig.1 is a natural generalization (to sequences of sets in the antecedent) of some classical first-order sequent rule. Moreover, by viewing the antecedent as a single set of formulas, the rules for both implication connectives would coincide. It is also easy to see that $(R\supset)$ is the unique rule creating a new set in the antecedent.

Definition 8. For any (possibly infinite) set $\Gamma \cup \{\varphi\} \subseteq \mathsf{sen}(\Sigma)$ we say that $\Gamma \vdash_\Sigma \varphi$ iff for some finite $\Gamma' \subseteq \Gamma$ there exists a proof of the sequent $\Gamma' \rhd \varphi$ using the calculus in Figure 1. ■

In general, a *proof* for the sequent $\Delta \rhd \varphi$ is a finite tree constructed using inference rules of the calculus, such that the root is the sequent $\Delta \rhd \varphi$ and whose leaves are labeled with initial sequents (in our case, these are $(Init)$, $(_FL)$, (R_F)). In particular, the antecedent Δ may be a unitary sequence of one finite set Γ. We recall that $\vdash_\Sigma$ is the relation induced (by the calculus in Fig.1) on the set $\mathcal{P}(\mathsf{sen}(\Sigma)) \times \mathsf{sen}(\Sigma)$. It is worthwhile noting that this relation satisfies reflexivity, monotonicity and transitivity, although any rule in the calculus (Fig.1) does not directly correspond with them. Besides, the $\vdash$-translation property is also satisfied. However, the extension to a relation between sequences of sets of formulas and formulas lacks to satisfy the former three properties.

4 The Logic Programming Language $Horn^{\supset}$

In this section we give the strong $\mathcal{FO}^{\supset}$ axiomatization for the static scope programming language introduced in [8]. Its syntax is an extension of the Horn clause language, by adding the intuitionistic implication $\supset$ in goals. We define this language as the following 4-tuple $Horn^{\supset} = (\mathcal{FO}^{\supset}, \underline{\mathsf{Sign}}', prog, goal)$, where $\underline{\mathsf{Sign}}'$ is the class of finite signatures in $\underline{\mathsf{Sign}}$ and, for each Σ in $\underline{\mathsf{Sign}}'$, $prog(\Sigma)$ is the set of all Σ-programs, which are finite sets of closed D-clauses (called Σ-*clauses*), and $goal(\Sigma)$ is the set of all closed G-clauses (called Σ-*goals*). D- and G-clauses are recursively defined as follows (where A stands for an atomic formula):

$$G := A \mid G_1 \wedge G_2 \mid D \supset G \mid \exists x G \qquad\qquad D := A \mid G \to A \mid D_1 \wedge D_2 \mid \forall x D$$

Following [8], we use a simple definition of the operational semantics of $Horn^{\supset}$, given by a nondeterministic set of rules which define when a Σ-goal G is operationally derivable from a program sequence $\Delta = P_0;\ldots;P_n$, in symbols $\Delta \vdash_s G$. Moreover, to deal with clauses in $P \in prog(\Sigma)$ of the form $D_1 \wedge D_2$ and $\forall x D$, we utilize the closure (w.r.t. conjunction and instantiation) set $[P]$ of all clauses in P. This abstract definition of the operational semantics is more suitable to be compared with the mathematical semantics of $Horn^{\supset}$.

Definition 9. $[P]$ is defined as the set $\cup\{[D] \mid D \in P\}$ where $[D]$ is recursively defined as follows: $[A] = \{A\}$, $[G \to A] = \{G \to A\}$, $[D_1 \wedge D_2] = [D_1] \cup [D_2]$, $[\forall x D] = \cup\{[D(t/x)] \mid t \in T_\Sigma \text{ and } t \text{ is closed}\}$. ■

(1) $\Delta \vdash_s A$ if A is atomic and $A \in [\Delta]$

(2) $\dfrac{P_0;\ldots;P_i \vdash_s G}{P_0;\ldots;P_i;\ldots;P_n \vdash_s A}$ if $G \to A \in [P_i]$ and $0 \leq i \leq n$

(3) $\dfrac{\Delta \vdash_s G_1 \quad \Delta \vdash_s G_2}{\Delta \vdash_s G_1 \wedge G_2}$ (4) $\dfrac{\Delta \vdash_s G(t/x)}{\Delta \vdash_s \exists x G}$ (5) $\dfrac{\Delta;\{D\} \vdash_s G}{\Delta \vdash_s D \supset G}$

Fig. 2. Operational Semantics for $Horn^{\supset}$.

Notice that $w \Vdash P \Leftrightarrow w \Vdash [P]$ and also that all clauses in $[P]$ match the pattern $G \to A$ (with G possibly empty for handling the case A). We extend the notation $[P]$ to $[\Delta]$ by $[P_0;\ldots;P_n] = \bigcup_{i=0}^{n}[P_i]$. Now, we define $\Delta \vdash_s G$ by means of the rules given in Figure 2. In order to illustrate the operational behaviour of this language we give the Example 10.

Example 10. Let the program with two clauses $P = \{((b \to c) \supset c) \to a,\ b\}$ and let the goal $G1 = a$. A proof of $P \vdash_s G1$ is given by the following steps (applying rules in Figure 2):

$$\begin{array}{rl} P \vdash_s a & \text{by Rule (2)} \\ \text{if } P \vdash_s (b \to c) \supset c & \text{by Rule (5)} \\ \text{if } P;\{b \to c\} \vdash_s c & \text{by Rule (2)} \\ \text{if } P;\{b \to c\} \vdash_s b & \text{by Rule (1) since } b \in P;\{b \to c\} \end{array}$$

However, let now the program with a unique clause $Q = \{((b \to c) \supset c) \to a\}$ and let the goal $G2 = b \supset a$. The only way to obtain a proof of $Q \vdash_s G2$ would make the following steps:

$$\begin{array}{rl} Q \vdash_s b \supset a & \text{by Rule (5)} \\ \text{if } Q;\{b\} \vdash_s a & \text{by Rule (2)} \\ \text{if } Q \vdash_s (b \to c) \supset c & \text{by Rule (5)} \\ \text{if } Q;\{b \to c\} \vdash_s c & \text{by Rule (2)} \\ \text{if } Q;\{b \to c\} \vdash_s b & \end{array}$$

Since the last sequent can not be proved then $Q \nvdash_s G2$. ■

This example shows the "static scope rule" meaning: the set of clauses which can be used to solve a goal depends on the program block's structure. Whereas $G1 = a$ can be proved from the program P because b was defined in P, in the case of $G2 = b \supset a$ and the program Q the "external" definition of b is not permitted for proving the body of the clause in Q. This is a mayor difference with the "dynamic scope rule" used in [16].

In the Appendix A we prove that the proof-subcalculus $\vdash_s$ is sound with respect to the $\mathcal{FO}^{\supset}$-calculus when restricted to the programming language $Horn^{\supset}$.

In the rest of this section we show that $Horn^{\supset}$ satisfies all the desirable properties to be a strong $\mathcal{FO}^{\supset}$ logic programming language. In Subsection 4.1 we present the mathematical (or model) semantics and we prove the goal completeness property. The operational semantics is studied in Subsection 4.2, showing the equivalence between mathematical and operational semantics. Also completeness of $\vdash_s$ w.r.t. the $\mathcal{FO}^{\supset}$-calculus will be proved there as a consequence of previous results. Along the whole section $\models$ (respectively $\vdash$) stands for the satisfaction and the logical consequence relations $\models_{\Sigma}$ (respectively the derivability relation $\vdash_{\Sigma}$) of $\mathcal{FO}^{\supset}$.

4.1 Mathematical Semantics and Goal Completeness

In this subsection we first define the subcategory $\underline{\mathsf{FMod}}(\Sigma)$ of $\underline{\mathsf{Mod}}(\Sigma)$. Its objects are Kripke structures with Herbrand interpretations associated to worlds, with a unique minimal world and closed w.r.t. superset. Then, we show that to deal with $Horn^{\supset}$ programs (as particular $\mathcal{FO}^{\supset}$-theories) the category $\underline{\mathsf{Mod}}(P)$ of Kripke Σ-structures satisfying P can be restricted to the subcategory $\underline{\mathsf{FMod}}(P)$. Notice that, for Horn clauses, the Herbrand models constitute the corresponding subcategory of the general first-order structures. We will prove the existence of a model in $\underline{\mathsf{FMod}}(P)$ which is initial in the whole category $\underline{\mathsf{Mod}}(P)$. Again, one can observe the parallelism with the least Herbrand model of Horn clauses. Finally, we will prove the goal completeness property w.r.t. this initial model.

Given a signature Σ, U_Σ and B_Σ will denote the Herbrand universe and the Herbrand base, respectively. Consider the complete lattice $\mathcal{P}(B_\Sigma)$ of all Herbrand (first-order) Σ-interpretations over the universe U_Σ. Any subset K of $\mathcal{P}(B_\Sigma)$, ordered by set inclusion, can be viewed as a Kripke Σ-structure. On these structures, $I, K \Vdash \varphi$ (or simply $I \Vdash \varphi$) will denote $w, K \Vdash \varphi$ for the world w whose first-order associated Σ-structure is I.

Definition 11. $\underline{\mathsf{FMod}}(\Sigma)$ is the full subcategory of $\underline{\mathsf{Mod}}(\Sigma)$ whose objects are the Kripke Σ-structures $\{Fil(I) \mid I \subseteq B_\Sigma\}$ where $Fil(I)$ denotes the *filter* $\{J \subseteq B_\Sigma \mid J \supseteq I\}$. $(\underline{\mathsf{FMod}}(\Sigma), \sqsubseteq)$ is the partial order given by $Fil(I_1) \sqsubseteq Fil(I_2)$ iff $I_1 \subseteq I_2$. The morphisms in $\underline{\mathsf{FMod}}(\Sigma)$ can be seen as these inclusions, that is $Fil(I_1) \sqsubseteq Fil(I_2)$ is the morphism $H \in \underline{\mathsf{Mod}}(\Sigma)$ defined by $\sigma_H(I_2) = I_1$ and the singleton $\{\subseteq : I_1 \to I_2\}$. ■

Remark 12. Note that the morphisms $H : K_1 \to K_2$ with $K_1 \in \underline{\mathsf{FMod}}(\Sigma)$ are unique since: (i) K_1 has only one minimal world and (ii) if A and B are first-order Σ-structures and A is finitely generated then the Σ-homomorphism $A \to B$ is unique. ■

Hence, for formulas $\varphi \supset \psi$, the forcing relation restricted to the class $\underline{\mathsf{FMod}}(\Sigma)$ satisfies: $I \Vdash \varphi \supset \psi$ iff for all $J \subseteq B_\Sigma$ such that $I \subseteq J$, if $J \Vdash \varphi$ then $J \Vdash \psi$.

Proposition 13. *Let I_1,I_2 be two Σ-interpretations, $\{I_j\}_{j \in J}$ a (possibly infinite) set of Σ-interpretations, D a Σ-clause and G a Σ-goal.*
(a) *If $I_1 \Vdash G$ then for all I_2 such that $I_1 \subseteq I_2$, $I_2 \Vdash G$*
(b) *If $I_j \Vdash D$ for each $j \in J$ then $\cap_j I_j \Vdash D$*

Proof. (a) is a direct consequence of persistence of goals (see Proposition 4). The proof of (b) can be made by structural induction on D: For $D = A$ it is trivial, since $I \Vdash A$ iff $A \in I$. Cases $D = D_1 \wedge D_2$ and $D = \forall x D_1$ can be easily proved by applying the induction hypothesis. For $D = G \to A$, the case $\cap_j I_j \Vdash A$ is trivial. Now suppose that $\cap_j I_j \nVdash A$, then there exists $j \in J$ such that $I_j \nVdash A$ and $I_j \nVdash G$. Hence $\cap_j I_j \nVdash G$ holds by (a), and therefore $\cap_j I_j \Vdash G \to A$. ■

Proposition 14. $(\underline{\mathsf{FMod}}(\Sigma), \sqsubseteq)$ *is a complete lattice with bottom $Fil(\emptyset) = \mathcal{P}(B_\Sigma)$.*

Proof. It is enough to define the operations $\sqcup$ and $\sqcap$ for any (possibly infinite) collection $\{Fil(I_i)\}_i$ as follows: $\sqcup_i Fil(I_i) = Fil(\cup_i I_i)$ and $\sqcap_i Fil(I_i) = Fil(\cap_i I_i)$. ■

The notion of satisfaction between elements in $\underline{\mathsf{FMod}}(\Sigma)$ and Σ-clauses (respectively -goals), borrowed from the underlying logic, is given by $Fil(I) \models D$ iff $I \Vdash D$ (respectively for G).
The class of models of a Σ-program P, denoted $\underline{\mathsf{FMod}}(P)$, is defined as $\underline{\mathsf{FMod}}(P) = \{K \in \underline{\mathsf{FMod}}(\Sigma) \mid K \models P\}$ or equivalently as $\{Fil(I) \mid I \subseteq B_\Sigma, I \Vdash P\}$. $\underline{\mathsf{FMod}}(P)$ is a full subcategory of $\underline{\mathsf{FMod}}(\Sigma)$.

Proposition 15. *There exists a least element M_P in $\underline{\mathsf{FMod}}(P)$ with respect to $\sqsubseteq$.*

Proof. $\underline{\mathsf{FMod}}(P)$ is not empty since $Fil(B_\Sigma) = \{B_\Sigma\}$ satisfies P. As a consequence of Proposition 13(b), the intersection ($\sqcap$) of elements in $\underline{\mathsf{FMod}}(P)$ is an element of $\underline{\mathsf{FMod}}(P)$. Then $M_P = \sqcap\{K \in \underline{\mathsf{FMod}}(\Sigma) \mid K \models P\}$ belongs to $\underline{\mathsf{FMod}}(P)$ and it is the least element w.r.t. $\sqsubseteq$. Moreover, $M_P = Fil(I_P)$ with $I_P = \cap\{I \subseteq B_\Sigma \mid I \Vdash P\}$. ■

Then, M_P is the initial object in the category $\underline{\mathsf{FMod}}(P)$. Now, we will prove the initiality of M_P in the (more general) category $\underline{\mathsf{Mod}}(P)$. Then, following [15], the denotation function $P \mapsto M_P$ is called the *mathematical semantics* of $Horn^{\supset}$.

Definition 16. A Σ-program P is *satisfiable* (respectively *F-satisfiable*) iff there exists $K \in \underline{\mathsf{Mod}}(\Sigma)$ (respectively $K \in \underline{\mathsf{FMod}}(\Sigma)$) such that $K \models P$. ■

Lemma 17. *For each $K \in \underline{\mathsf{Mod}}(\Sigma)$ there exists $I_K \in \mathcal{P}(B_\Sigma)$ (therefore $Fil(I_K) \in \underline{\mathsf{FMod}}(\Sigma)$) such that, for every Σ-clause D and every Σ-goal G:*
(a) *If $K \models D$ then $Fil(I_K) \models D$*
(b) *If $Fil(I_K) \models G$ then $K \models G$*
Moreover, there exists a unique morphism $H_K : Fil(I_K) \to K$.

Proof. Let $K = (W(K), \preceq, \langle \mathcal{A}_w \rangle_{w \in W(K)})$. We consider, for each $w \in W(K)$, the H-interpretation $I_w = \{p(t_1, ..., t_n) \in B_\Sigma \mid w, K \Vdash p(t_1, ..., t_n)\}$ and let $I_K = \cap\{I_w \mid w \in W(K)\}$. That is, $I_K = \{p(t_1, ..., t_n) \in B_\Sigma \mid K \models p(t_1, ..., t_n)\}$. Then, for each Σ-clause D and each Σ-goal G:

(i) If $w, K \Vdash D$ then $I_w \Vdash D$
(ii) If $I_w \Vdash G$ then $w, K \Vdash G$

The proof of above facts (i) and (ii) is made by simultaneous induction on D and G. (i) and (ii) for an atom A: $w, K \Vdash A$ iff $A \in I_w$ iff $I_w \Vdash A$. (i) for $D_1 \wedge D_2$, $\forall x D$ and (ii) for $G_1 \wedge G_2$, $\exists x G$, can be easily proved by applying the induction hypothesis. To prove (i) for $G \to A$, let us suppose that $w, K \Vdash G \to A$, then $w, K \Vdash A$ or $w, K \not\Vdash G$. By the induction hypothesis, $I_w \Vdash A$ or $I_w \not\Vdash G$ holds. Therefore $I_w \Vdash G \to A$. To prove (ii) for $D \supset G$, suppose that $w, K \not\Vdash D \supset G$, then there exists $v \in W(K)$ such that $w \preceq v$, $v, K \Vdash D$ and $v, K \not\Vdash G$. By induction, $I_v \Vdash D$ and $I_v \not\Vdash G$ hold. Then $I_w \not\Vdash D \supset G$, since $w \preceq v$ implies $I_w \subseteq I_v$.
Now, to prove (a), let us suppose that $K \models D$, then for all minimal $w \in W(K)$: $w, K \Vdash D$. Hence, by (i), for all minimal $w \in W(K)$: $I_w \Vdash D$. Then, by Proposition 13(b), $I_K \Vdash D$ holds. Therefore $Fil(I_K) \models D$. The proof for (b) is symmetric, suppose that $Fil(I_K) \models G$, this means that $I_K \Vdash G$. Then, by Proposition 13(a), $I_w \Vdash G$ holds for all minimal $w \in W(K)$. Therefore by (ii), $w, K \Vdash G$ for all minimal $w \in W(K)$. Hence $K \models G$.
The unique morphism $H_K : Fil(I_K) \to K$ is given by the collection of unique first-order Σ-homomorphisms $\{H_w : I_K \to \mathcal{A}_w \mid w \text{ minimal in } W(K)\}$. ■

Theorem 18. *M_P is initial in the category $\underline{\mathsf{Mod}}(P)$.*

Proof. Given $K \in \underline{\mathsf{Mod}}(P)$, the unique morphism from M_P into K is $H = H_K \circ \sqsubseteq$ obtained by composing the two morphisms $\sqsubseteq$: $M_P \to Fil(I_K)$ and $H_K : Fil(I_K) \to K$ of the previous lemma. ■

Corollary 19. *A Σ-program P is satisfiable iff it is F-satisfiable.* ■

Now, we will show that M_P is typical in $\underline{\mathsf{Mod}}(P)$ (and also in $\underline{\mathsf{FMod}}(P)$) w.r.t. goal satisfaction.

Proposition 20. *For each Σ-program P and each Σ-goal G: $P \models G$ iff $Fil(I) \models G$ for all $Fil(I) \in \underline{\mathsf{FMod}}(P)$.*

Proof. The only-if part is trivial. For the if part let $K \in \underline{\mathsf{Mod}}(P)$, that means $K \models P$. Then by Lemma 17 $Fil(I_K) \models P$. Then $Fil(I_K) \models G$ and, again by Lemma 17, $K \models G$. ■

Theorem 21. *For each Σ-program P and each Σ-goal G: $P \models G$ iff $M_P \models G$.*

Proof. The only-if part is trivial. Conversely, $M_P \models G$ is equivalent to $\cap\{I \subseteq B(\Sigma) \mid I \Vdash P\} \Vdash G$. Therefore $I \Vdash G$ for all $I \subseteq B(\Sigma)$ such that $I \Vdash P$, hence $Fil(I) \models G$ for all $Fil(I) \in \underline{\mathsf{FMod}}(P)$. Then by Proposition 20, $P \models G$. ■

From this result and the fact of that $\mathcal{FO}^{\supset}$ is a complete logic, the goal completeness property is obtained:

Theorem 22. *For each Σ-program P and each Σ-goal G, $P \vdash G$ iff $M_P \models G$.* ■

Remark 23. It is worthwhile noting that: $Fil(I) \models G$ (or $I \Vdash G$) iff G is logical consequence of I. This can be proved by Proposition 20, by seeing I as a (posibly infinite) program of ground atoms, and by persistency of G. ■

4.2 Operational Semantics

In this subsection we first define, for each Σ-program P, an immediate consequence operator T_P (on $\underline{\mathsf{FMod}}(\Sigma)$). The monotonicity and continuity of T_P in the lattice $(\underline{\mathsf{FMod}}(\Sigma), \sqsubseteq)$ allow us to use the fixpoint semantics as a bridge between the mathematical and the operational semantics. First we prove the equivalence between mathematical and fixpoint semantics and then between fixpoint and operational semantics (given by $\vdash_s$). Specifically, given a Σ-program P, we will use the fixpoint characterization of the least model M_P of P in terms of T_P, to prove that for every Σ-goal G, $M_P \models G$ if and only if $P \vdash_s G$. We will also show that the proof-subcalculus $\vdash_s$ is sound and complete with respect to the $\mathcal{FO}^{\supset}$ derivability relation, restricted to $Horn^{\supset}$-programs and -goals.

Definition 24. The immediate consequence operator $T_P : \underline{\mathsf{FMod}}(\Sigma) \to \underline{\mathsf{FMod}}(\Sigma)$ is given by $T_P(Fil(I)) = Fil(\{A \mid \text{there exists } G \to A \in [P] \text{ such that } Fil(I) \models G\})$. ■

The operator T_P has been defined in terms of the satisfaction relation of $\mathcal{FO}^{\supset}$. That is, given a filter (generated by a set of ground atoms), it generates the head of the clauses whose bodies are satisfied by this filter. We want to remark that T_P is indeed a $\mathcal{FO}^{\supset}$ logical consequence operator because we can replace (see Remark 23) the satisfaction of G in the model $Fil(I)$ (or equivalently the forcing relation of G in the minimal world I) by the logical consequence of G from I. Unlikely for Horn clauses, logical consequence can not be replaced by set membership since goals are not just conjunction of atoms. It is well-known that the least fixpoint and the least pre-fixpoint of a continuous operator in a complete lattice is $T^{\omega}(\bot)$ where $\bot$ is the bottom in the lattice. In the Appendix B we prove that the above-defined operator T_P is monotone and continuous in the complete lattice $(\underline{\mathsf{FMod}}(\Sigma), \sqsubseteq)$ and also that the models of P are the pre-fixpoints of T_P . Therefore, the least fixpoint of T_P is $T_P^w(\mathcal{P}(B_\Sigma))$ which will be simply denoted T_P^w. Then the correspondence between mathematical and fixpoint semantics is a direct consequence of these results.

Theorem 25. *For all Σ-program P, $T_P^w = M_P$.* ■

Now we will prove the equivalence between mathematical, fixpoint and operational semantics. We need the following lemma to complete such equivalences. This result was proved in [8] and our proof is an adaptation (for our operator T_P) of the proof given there. For that reason we will give a sketch of this proof detailing only the main differences.

Lemma 26. *Given a Σ-program P and a Σ-goal G, if $T_P^w \models G$ then $P \vdash_s G$.*

Sketch of the proof. Let I_n denote the minimal world in $T_P^n(\mathcal{P}(B_\Sigma))$, for each $n \geq 0$. Since $T_P^w = \sqcup_{n<w} T_P^n(\mathcal{P}(B_\Sigma))$, the minimal world in T_P^w is $\cup_{n<w} I_n$. Then by continuity of T_P it suffices to prove that $I_n \Vdash G \Longrightarrow P \vdash_s G$ holds for each $n \geq 0$. The proof is made by induction on the highest number m of $(\supset)$-nesting levels in P and G. If $m = 0$ (there are no occurrences of $\supset$ either in P or in G), then the proof can be done by double induction on n and G. The induction hypothesis holds for at most $m-1$ $(\supset)$-nesting levels in P and G. For the case $m > 0$, let us develop in detail only the subcase $n > 0$ and $G = D_1 \supset G_1$. Let D_1' be the program $I_n \cup D_1$ (seen the atoms in I_n as clauses with empty bodies) and let $I_{D_1'}$ be the minimal world in $T_{D_1'}^{\omega}$. Then $I_{D_1'} \Vdash D_1$ and $I_n \subseteq I_{D_1'}$. Therefore $I_{D_1'} \Vdash G_1$. By induction on D_1' and G_1 (note that the highest number of $(\supset)$-nesting levels in D_1' and G_1 is less than m), $I_n \cup D_1 \vdash_s G_1$ holds. Finally, some $\vdash_s$-properties easy to prove (see [8] for details) are used to obtain the following implications: $I_n \cup D_1 \vdash_s G_1 \Longrightarrow I_n; D_1 \vdash_s G_1 \Longrightarrow I_n \vdash_s (D_1 \supset G_1) \Longrightarrow \{A \mid P \vdash_s A\} \vdash_s (D_1 \supset G_1) \Longrightarrow P \vdash_s (D_1 \supset G_1)$. ■

The following Theorem summarizes all the obtained results. In particular, the equivalence between mathematical and operational semantics is given by $(c) \Leftrightarrow (e)$.

Theorem 27. *For each Σ-program P and each Σ-goal G, the following sentences are equivalent:*

(a) $P \models G$
(b) $P \vdash G$
(c) $M_P \models G$
(d) $T_P^w \models G$
(e) $P \vdash_s G$

Proof. $(a) \Leftrightarrow (b)$ by the soundness and completeness of $\mathcal{FO}^{\supset}$
$(b) \Leftrightarrow (c)$ by the goal completeness property (Th. 22)
$(c) \Leftrightarrow (d)$ by the equivalence between mathematical and fixpoint semantics (Th. 25)
$(d) \Rightarrow (e)$ by the Lemma 26
$(e) \Rightarrow (b)$ by the soundness of $\vdash_s$ w.r.t. $\vdash$ (Th. 30 in Appendix A) ■

Corollary 28. *The proof-subcalculus $\vdash_s$ is complete with respect to the $\mathcal{FO}^{\supset}$-calculus when restricted to the programming language $Horn^{\supset}$.*

We have used an abstract formulation of the operational semantics, given by the proof-subcalculus $\vdash_s$. The effectiveness of such subcalculus means the capability for implementing it. This task is out of the scope of this paper, however we would like to mention here some works giving the main ideas towards such implementation. In [8] a less abstract operational semantics is given by using notions of substitution, unification and variable renaming for the notation $[P]$. Whereas this semantics is equivalent to the given one, it provides an abstract interpreter for the language $Horn^{\supset}$. In [1] are also shown, by means of examples, some of the most relevant points taken into account to make a concrete implementation.

5 Conclusions and Related Work

We have presented a new characterization for the language $Horn^{\supset}$ of Horn clauses extended with static embedded implication (introduced in [8]). Our characterization is based on the methodology proposed in [14, 15] for define logic programming languages. Hence, we have enriched the underlying logic ($\mathcal{FO}$) of the original language (Horn clauses) with intuitionistic implication, in a very natural way, obtaining the complete logic $\mathcal{FO}^{\supset}$. Then we have given a $\mathcal{FO}^{\supset}$-axiomatization of $Horn^{\supset}$, showing that it satisfies all the desirable mathematical and computational properties. The fact of fixing the underlying logic $\mathcal{FO}^{\supset}$ allows us to deal with $Horn^{\supset}$-programs as special $\mathcal{FO}^{\supset}$-theories. Therefore, metalogical properties of programs and goals can be studied in a clean and sound way relative to fixed notions (as model, satisfaction, morphism, derivability, etc.) in the underlying framework. Following this methodology, we have obtained a subclass ($\underline{\mathsf{FMod}}(\Sigma)$) of logical structures powerful enough for dealing with $Horn^{\supset}$-programs, like the subclass of Herbrand interpretations is for Horn clauses in the first-order case. Indeed, we show that a program (as a theory) has a (general) model iff it has a model in the subclass $\underline{\mathsf{FMod}}(\Sigma)$. We believe that this is an important result about the model-theoretic semantics of $Horn^{\supset}$. Actually, the equivalence between the two model-theoretic semantics presented in [8] is a direct consequence of the definition of $\underline{\mathsf{FMod}}(\Sigma)$. Moreover, $\underline{\mathsf{FMod}}(\Sigma)$ is crucial for both: the initial and the fixpoint semantics. On one hand, for any program P, $\underline{\mathsf{FMod}}(P)$ has a least element M_P which can be obtained by intersection of all models of P and also as the ω-iteration of a continuous immediate consequence operator T_P defined on $\underline{\mathsf{FMod}}(\Sigma)$. Our fixpoint semantics is essentially equivalent to the fixpoint semantics of [8], although it is obtained in a very different way. As we pointed out in Subsection 4.2, the operator T_P is indeed based on the logical consequence of the underlying logic (or equivalently on its satisfaction relation). However, the immediate consequence operator of [8] is based on the notion of environment and it requires an ad-hoc satisfaction relation between Herbrand interpretations and goals. Moreover, we prove that the operational semantics of $Horn^{\supset}$ is equivalent to the underlying logical derivability relation. In fact, this derivability

relation is induced by a calculus (Figure 1) designed as an extension of the operational semantics of $Horn^{\supset}$. On the other hand, we have showed that $Horn^{\supset}$-programs are $\mathcal{FO}^{\supset}$-theories with initial semantics: M_P (or equivalently T_P^{ω}) is the initial object in the class $\underline{\mathsf{Mod}}(P)$ of all (general) models of the program P. Hence, our characterization of $Horn^{\supset}$, firstly, places some well-known results into the logical framework given by $\mathcal{FO}^{\supset}$ and, secondly, it extends these results to a strong axiomatization providing a well-established model-theoretic semantics and an initial semantics.

We believe that further extensions of this logic programming language, for example with some kind of negation, could be better developed using the logical foundation provided by this strong $\mathcal{FO}^{\supset}$-axiomatization. With respect to this matter, there are several papers dealing with dynamic intuitionistic implication and some kind of negation, e.g. [3, 5, 9, 10, 12, 13]. We plan to investigate also a possible $\mathcal{FO}^{\supset}$-axiomatization of the dynamic scope language of [16] in order to place both languages (from [8] and [16]) into the common underlying logic $\mathcal{FO}^{\supset}$. $\mathcal{FO}^{\supset}$ and intuitionistic logic are essentially equivalent to deal with the latter language. We mean, although these two logics differ in the universal quantifier interpretation, both coincide in clause interpretation over structures with constant universe, and it is well-known (cf. [2, 7]) that these structures are powerful enough. In [16] it is proved that the operational semantics of its language corresponds to intuitionistic derivability. In [2] it is shown that the canonical model (of a program), obtained in [16] by a fixpoint construction, is indeed an intuitionistic model of the program. They also give an intuitionistic (Kripke's based) model-theory for this language. Apart from the difference in the considered programming language, there are three most remarkable differences with our Kripke's based approach: their logical structures are generated by terms, our notions of satisfaction and logical consequence are different, and the worlds of their canonical model are indexed by programs.

A different approach to give logical foundations to this kind of logic programming languages (or in general to Horn clause extensions) is the transformational one which consists in translating programs to the language of some well-known logic. In [7] the language defined in [8] is translated to $S4$-modal logic. They also translate the language defined in [16] in order to set both languages into a common logical framework.

The transformational approach is also taken in [19, 20] where logic programs with embedded implications are translated to Horn clause programs. In [20] the definition of a predicate in a new module overrides its definition in previous modules, therefore nested definitions are independent of definitions in outer modules. The semantics of such languages can be defined by a direct mapping from programs in the extended language to Horn clause programs. Then, Horn clause theory can be used to give logical and computational foundation to the extended language. However, as it is pointed in [2, 20], when predicate extension is allowed, the translation of each predicate definition (inside a module) raises different predicate definitions, each one depending on the collection of modules that have to be used. In dynamic scoped languages this collection can only be determined in run-time, forcing to add new arguments to the translated predicates to represent the modules currently in use. This makes the transformational approach inadequate for both semantics and implementation issues. For static scoped languages, such as the language studied in this paper, this approach could be still useful for implementation issues, since there is a lexical way to determine such collection of modules (for each goal). However, the translation would not be so direct because of the multiple transformation of each original predicate. Therefore, in our opinion, for semantical foundation it is more adequate the model-theoretic approach started in [8], whose results we have enriched by setting a well-stablished logical framework.

Acknowledgment: The authors are greatly indebted to Fernando Orejas for fruitful discussions and suggestions.

References

1. Arruabarrena, R. and Navarro, M. On Extended Logic Languages Supporting Program Structuring, In: *Proc. of APPIA-GULP-PRODE'96*, 191-203, (1996).
2. Bugliesi, M., Lamma, E. and Mello, P., Modularity in Logic Programming, *Journal of Logic Programming*, (19-20): 443-502, (1994).
3. Bonner, A. J., and McCarty, L. T., Adding Negation-as-Failure to Intuitionistic Logic Programming, In: *Proc. of the North American Conf. on Logic Programming*, MIT Press, 681-703, (1990).
4. Bonner, A. J., McCarty, L. T., and Vadaparty, K., Expresing Database Queries with Intuitionistic Logic. In: *Proc. of the North American Conf. on Logic Programming*, MIT Press, 831-850, (1989).
5. Gabbay, D. M., N-Prolog: An Extension of Prolog with Hypothetical Implications. II. Logical Foundations and Negation as Failure, *Journal of Logic Programming* 2(4):251-283 (1985).
6. Gabbay, D. M. and Reyle, U., N-Prolog: An Extension of Prolog with Hypothetical Implications. I., *Journal of Logic Programming* 1(4):319-355 (1984).
7. Giordano, L., and Martelli, A.; Structuring Logic Programs: A Modal Approach, *Journal of Logic Programming* 21:59-94 (1994).
8. Giordano, L., Martelli, A., and Rossi, G., Extending Horn Clause Logic with Implication Goals, *Theoretical Computer Sscience*, 95:43-74, (1992).
9. Giordano, L., and Olivetti, N.; Combining Negation as Failure and Embedded Implications in Logic Programs, *Journal of Logic Programming* 36:91-147 (1998).
10. Harland., J. Succecs and Failure for Hereditary Harrop Formulae, *Journal of Logic Programming*, 17:1-29, (1993).
11. Lucio, P. $\mathcal{FO}^{\supset}$: A Complete Extension of First-order Logic with Intuitionistic Implication, Technical Research Report UPV-EHU/LSI/TR-6-98, URL address: http://www.sc.ehu.es/paqui, Submitted to a journal for publication.
12. McCarty, L. T., Clausal Intuitionistic Logic I. Fixed-Point Semantics, *Journal of Logic Programming*, 5:1-31, (1988).
13. McCarty, L. T., Clausal Intuitionistic Logic II. Tableau Proof Procedures, *Journal of Logic Programming*, 5:93-132, (1988).
14. Meseguer, J., General Logics, In: Ebbinghaus H.-D. et al. (eds), *Logic Colloquium'87*, North-Holland, 275-329, (1989).
15. Meseguer, J., Multiparadigm Logic Programming, In: *Proccedings of ALP'92*, L.N.C.S. 632. Springer-Verlag, 158-200, (1992).
16. Miller, D., A Logical Analysis of Modules in Logic Programming, In: *Journal of Logic Programming*, 6:79-108, (1989).
17. Miller, D., Abstraction in Logic Programs. In: Odifreddi, P. (ed), *Logic and Computer Science*, Academic Press, 329-359, (1990).
18. Miller, D., Nadathur, G., Pfenning, F. and Scedrov, A., Uniform Proofs as a Foundation for Logic Programming, *Annals of Pure and App. Logic*, 51:125-157, (1991).
19. Monteiro, L., Porto, A., Contextual Logic Programmming, In: *Proc. 6th International Conf. on Logic Programming*, 284-299, (1989).
20. Moscowitz, Y., and Shapiro, E., Lexical logic programs, In: *Proc. 8th International Conf. on Logic Programming*, 349-363, (1991).
21. van Dalen, D., and Troelstra, *Constructivism in Mathematics: An Introduction* Vol.1 and Vol.2, Elsevier Science, North-Holland, (1988).

A Appendix: Soundness of the proof-subcalculus

We prove here that the proof-subcalculus $\vdash_s$ is sound with respect to the $\mathcal{FO}^{\supset}$-calculus when restricted to the programming language $Horn^{\supset}$. In the following, the rules in the $\mathcal{FO}^{\supset}$-calculus and the rules in $\vdash_s$ will be respectively called the *logical* and *operational* rules.

Lemma 29. *Let Δ be a sequence of Σ-programs $P_0; ...; P_n$ $(n \geq 0)$ and G be a Σ-goal. If $\Delta \vdash_s G$ then $\Delta \vdash G$.*

Proof. Soundness of $\vdash_s$ w.r.t. $\vdash$ would be obvious if each operational rule was a logical rule, but there is a slight difference: the use of $(\forall L)$ and $(\wedge L)$ logical rules is compensated by the use of notation $[\Delta]$ in operational rules (1) and (2). So that, each of the operational rules (1) through (5) is derivable in the $\mathcal{FO}^{\supset}$-calculus in the following way: Rule (1) is derivable using a number of steps of $(\forall L)$ and $(\wedge L)$ and one step of $(Init)$. Rule (2) can be seen as a particular case of $(\rightarrow L)$ when $\chi = \psi$. For this reason Rule (2) does not need a second premise which holds by $(Init)$. Therefore, Rule (2) is a combination of $(\forall L)$, $(\wedge L)$, $(\rightarrow L)$ and $(Init)$. Rule (3) is $(R\wedge)$, Rule (4) is $(R\exists)$ and Rule (5) is $(R \supset)$.
Now, a proof of the sequent $\Delta \rhd G$ can be made by substituting the corresponding step(s) in the $\mathcal{FO}^{\supset}$-calculus for each step in the proof of $\Delta \vdash_s G$. ■

As a particular case of this lemma, for Δ being a single program P, the following result is obtained:

Theorem 30. *Given a Σ-program P and Σ-goal G, if $P \vdash_s G$ then $P \vdash G$.* ■

B Appendix: Fixpoint Semantics

In this part, we prove the results that are sufficient to establish that T_P^{ω} is the least fixpoint of the operator T_P defined in Subsection 4.2 and that T_P^{ω} is the least model of P.

Proposition 31. *T_P is monotone.*

Proof. Suppose that $Fil(I_1) \sqsubseteq Fil(I_2)$, that is $I_1 \subseteq I_2$. Then (by Proposition 13(a)) $\{A \mid G \rightarrow A \in [P],\ I_1 \Vdash G\} \subseteq \{A \mid G \rightarrow A \in [P],\ I_2 \Vdash G\}$ holds. Therefore $T_P(Fil(I_1)) \sqsubseteq T_P(Fil(I_2))$. ■

In order to prove the continuity of T_P, we first establish the following key lemma:

Lemma 32. *For every chain $I_1 \subseteq I_2 \subseteq ... \subseteq I_j \subseteq ...$, of Herbrand Σ-interpretations, every Σ-clause D and every Σ-goal G,*
(a) $\cup_j I_j \Vdash G \Longrightarrow$ *there exists j_0 such that $I_{j_0} \Vdash G$*
(b) $\cup_j I_j \nVdash D \Longrightarrow$ *there exists j_0 such that $I_{j_0} \nVdash D$*

Proof. We proceed by simultaneous induction. For atoms (a) and (b) are trivial since $I \Vdash A$ iff $A \in I$. (a) for $G = \exists x G_1$ and (b) for $D = D_1 \wedge D_2$, $D = \forall x D_1$ can be easily proved by the induction hypothesis.
To prove (a) for $G = G_1 \wedge G_2$ suppose that $\cup_j I_j \Vdash G_1 \wedge G_2$. Then for some indices j_1,j_2: $I_{j_1} \Vdash G_1$ and $I_{j_2} \Vdash G_2$. Hence $I_j \Vdash G_1 \wedge G_2$ holds for $j = max(j_1, j_2)$.
Now, consider (b) for $D = G_1 \rightarrow A_1$. If $\cup_j I_j \nVdash D$ then $\cup_j I_j \Vdash G_1$ and $A_1 \notin \cup_j I_j$. By induction, there exists j_0 such that $I_{j_0} \Vdash G_1$ and $A_1 \notin I_{j_0}$. Therefore $I_{j_0} \nVdash D$.
In order to prove (a) for $G = D_1 \supset G_1$, we proceed by contradiction. Let us suppose that for all index j: $I_j \nVdash D_1 \supset G_1$. Then, for each j, there exists I'_j such that $I_j \subseteq I'_j$, $I'_j \Vdash D_1$ and $I'_j \nVdash G_1$. Considering, for each j, the non-empty set of interpretations $C_j = \{I \mid I_j \subseteq I,\ I \Vdash D_1,\ I \nVdash G_1\}$ and taking, for each j, the interpretation $I'_j = \cap\{I \mid I \in C_j\}$, the following facts are verified:

(i) $I_j \subseteq I'_j$, for all j
(ii) $I'_j \Vdash D_1$ and $I'_j \nVdash G_1$, for all j
(iii) $\{I'_j\}_j$ form the chain $I'_1 \subseteq I'_2 \subseteq ... \subseteq I'_j \subseteq ...$

By applying the induction hypothesis on D_1, G_1 and the chain $\{I'_j\}_j$, we have $\cup_j I'_j \Vdash D_1$ and $\cup_j I'_j \nVdash G_1$. Since $\cup_j I_j \subseteq \cup_j I'_j$, then $\cup_j I_j \nVdash D_1 \supset G_1$, in contradiction with the hypothesis. ■

Theorem 33. *Let* $Fil(I_1) \sqsubseteq Fil(I_2) \sqsubseteq ... \sqsubseteq Fil(I_j) \sqsubseteq ...$ *be a chain of elements in* $\underline{\mathsf{FMod}}(\Sigma)$. *Then* $T_P(\sqcup_j Fil(I_j)) = \sqcup_j T_P(Fil(I_j))$.

Proof. $\sqcup_j T_P(Fil(I_j)) \sqsubseteq T_P(\sqcup_j Fil(I_j))$ holds by monotonicity. The reverse inclusion is equivalent to prove that $\{A \mid G \rightarrow A \in [P],\ \cup_j I_j \Vdash G\} \subseteq \cup_j \{A \mid G \rightarrow A \in [P],\ I_j \Vdash G\}$. Let $A \in \{A \mid G \rightarrow A \in [P],\ \cup_j I_j \Vdash G\}$. Then, for some G: $G \rightarrow A \in [P]$ and $\cup_j I_j \Vdash G$. Since $I_1 \subseteq I_2 \subseteq ... \subseteq I_j \subseteq ...$, there exists an index j_0 such that $I_{j_0} \Vdash G$. Then $A \in \{A \mid G \rightarrow A \in [P],\ I_{j_0} \Vdash G\} \subseteq \cup_j \{A \mid G \rightarrow A \in [P],\ I_j \Vdash G\}$. ■

The following lemma states that the models of P are the pre-fixpoints of T_P.

Lemma 34. *Let* P *be a* Σ*-program and* $Fil(I) \in \underline{\mathsf{FMod}}(\Sigma)$. *Then* $Fil(I) \in \underline{\mathsf{FMod}}(P)$ *iff* $T_P(Fil(I)) \sqsubseteq Fil(I)$.

Proof. Let $Fil(I) \in \underline{\mathsf{FMod}}(P)$ and let us show that $\{A \mid G \rightarrow A \in [P],\ I \Vdash G\} \subseteq I$. If $A \in \{A \mid G \rightarrow A \in [P],\ I \Vdash G\}$, then there exists some G such that $G \rightarrow A \in [P]$ and $I \Vdash G$. Therefore $A \in I$, since $I \Vdash [P]$. Conversely, let $\{A \mid G \rightarrow A \in [P],\ I \Vdash G\} \subseteq I$. We have to show that $Fil(I) \models G \rightarrow A$, for each $G \rightarrow A \in [P]$. Suppose $Fil(I) \models G$. Then $A \in \{A \mid G \rightarrow A \in [P],\ I \Vdash G\} \subseteq I$. Hence $Fil(I) \models A$. ■