Uniform proofs and fixpoint semantics of sequent calculi

Gianluca Amato

Abstract

We propose a generalization of the Miller's definition of abstract logic programming language. The new definition keeps in account the fact that computing in logic languages is a process of searching for proofs that gives a result: it can be a correct answer, a call pattern, a resultant, according to the observable we are interested in. In the meanwhile, we provide a fixpoint semantics for sequent calculi and we show that our generalization of completeness of uniform proofs corresponds to particular compositionality properties of the semantics. Hereditarily Harrop formulas are used trough the paper as a working example.

Keywords: logic programming, compositionality, uniform proofs, sequent calculi, hereditarily Harrop formulas

1 Introduction

One of the greatest benefits of logic programming, as presented in [11] and [1], is that it is based upon the notion of *executable specifications*. The text of a logic program is endowed with both an operational (algorithmic) interpretation and an independent mathematical meaning which agree each other in several ways.

An operational interpretation (the realm of programming languages) is needed if we really want to write programs that can be executed, while a clear mathematical meaning (the realm of logic) simplifies the work of the programmer, who can focus himself on "what to do" rather then "how to do it". The problem is that operational expressiveness (intended as the capability of directing the flow of execution of a program) tends to obscure the declarative meaning. Research in logic programming strives to find a good balance between these opposite needs.

Uniform proofs [12, 14] have widely been accepted as the main tool to distinguish between logic programming languages and logic tout-court. Logic programming languages are essentially defined as fragments of a broader logic, designed in such a way that uniform proofs are complete among all the cut-free proofs. Miller argues that an evaluation in logic programming is the search for simple, cut-free, sequent proofs. According to this idea, completeness of uniform proofs gives a strong operation flavour to the language. The search for proofs becomes goal-directed, i.e. logical

Gianluca Amato is with Dipartimento di Informatica, Università degli Studi di Pisa. E-mail: amato@di.unipi.it

connectives in goals are decomposed independently from the program. This is not enough to say that a logic language can be efficiently implemented, but it is a first requirement which must be satisfied.

The idea that evaluation is the same as search for proof seems a bit too restrictive. A number of papers (such as [2] and [4]) have shown that, to better understand the nature of logic programming, we need to consider the fact that the evaluation of a goal gives a result. This can be an answer substitution, a resultant, a call pattern, according to the *observable* we are interested in. Uniform proofs, on the contrary, are only tailored to the simplest of these observables, i.e. the existence of a proof.

Moreover, the notion of uniform proof is deeply related to the operational semantics. However, one of the main properties of standard logic programming is the existence of three different styles of semantics (operational, declarative and fixpoint) which give the same denotation, i.e. the least Herbrand model. In particular, the fixpoint semantics is a cornerstone in the definition of hierarchies of semantics for logic programs [4], based on abstract interpretation [6]. As discussed in [7], the fixpoint semantics can be viewed as a bottom-up variant of the classical top-down operational semantics. It seems natural, in force of that, to look for a fixpoint reformulation of the property of completeness of uniform proofs.

We start by providing a declarative and fixpoint semantics for sequent calculi. Given a logic language and a set of inference rules, these semantics are parametric w.r.t. to a *pre-interpretation*, which is essentially a choice of semantic domains and intended meanings for the inference rules. When a pre-interpretation is given, we have fixed a particular property of proofs (observable) we want to focus our attention on. Regardless of the chosen observable, the two semantics are proved to give the same denotation.

Then, we introduce the topic of compositionality. We give the basic definitions for a sort of general theory of compositional semantics for sequent calculi. Inside this framework, it is possible to treat classical problems, like and-compositionality of correct answers, and other issues more concerned with proof theory.

At last, uniform proofs are introduced, as proofs that do not use some of the inference rules. It is proved that, if we restrict the inference rules of a logic in such a way that all proofs are uniform, we obtain a fixpoint semantics with some of the compositional properties we have introduced before. When a logic and its restriced variant have the same semantics, we say that the logic is an abstract logic programming language. Since semantics is parametric w.r.t. pre-interpretations, the property of being an abstract logic programming language depends from the chosen observable.

Throughout the paper, we use hereditarily Harrop formulas as a concrete example of the properties that we first introduce in abstract terms.

2 Preliminaries

Logics can be presented in several different ways: we will stick to a Gentzen-like proof-theoretic formulation. Let us fix two languages $\mathcal{D}$ and $\mathcal{G}$. We call *clauses* the elements of $\mathcal{D}$ and *goals* the elements of $\mathcal{G}$. If Γ and Δ are two sequences of clauses and goals respectively, $\Gamma \rightarrow \Delta$ is a *sequent*. If Δ is a sequence of length one, we

have an *intuitionistic* sequent.

A *proof schema* is a rooted ordered tree with sequents as nodes. Given a proof schema P, we call $\mathrm{hyp}(P)$ (*hypotheses*) the sequence of leaves of P taken from a pre-order visit of P. We call $\mathrm{th}(P)$ (*theorem*) the root of P. We denote a proof schema P, with $\mathrm{hyp}(P) = S_1, \ldots, S_n$ and $\mathrm{th}(P) = S$, by

$$P : S_1, \ldots, S_n \vdash S \tag{2.1}$$

Note that a schema of height zero (when the root is also the only hypothesis) and a schema of height one with no hypotheses are different. If S is a sequent, a proof schema of S with height zero is the *empty proof schema* for S, and is denoted by ϵ_S.

An *inference rule* is just a proof schema of height one. A proof schema P, which is obtained by gluing together the empty proof schemas and the inference rules in a given set $\mathcal{R}$, is called *proof*. We write $P : S_1, \ldots, S_n \vdash_{\mathcal{R}} S$ to stress the fact that P is indeed a proof. If we omit P from the previous notation, we only mean that there is a proof of S from $S_1, \ldots, S_n$. We say that a sequent S is *provable* if there is a proof of S with no hypotheses. Finally, we call *logic* a triple $(\mathcal{D}, \mathcal{G}, \mathcal{R})$.

2.1 Hereditarily Harrop formulas

We instantiate the previous definitions for first-order *hereditarily Harrop formulas* [9]. This language is a powerful extension of Horn clauses, which has been proposed as a mean for embedding modularity and abstraction in logic programming. It will be the working example used throughout this paper to illustrate the definitions and properties that we introduce from an abstract perspective. Here and in the following, whenever we work on hereditarily Harrop formulas, we only consider intuitionistic sequents.

Given a set A of first-order atomic formulas, clauses and goals of the language are recursively defined as follows

$$D := \mathsf{true} \mid A \mid D_1 \vee D_2 \mid G \supset D \mid \forall x.D$$
$$G := \mathsf{true} \mid A \mid G_1 \vee G_2 \mid G_1 \wedge G_2 \mid \exists x.G \mid D \supset G \mid \forall x.G.$$

Figure 1 shows the inference rules, which are a subset of those for first order intuitionistic logic. We make the assumption that there are no two different bindings for the same variable in a formula (i.e. $\exists x.p(x) \wedge \exists x.t(x)$ is not a formula of the language). In this way, we do not lose generality, yet we can use variable names to uniquely identify a binding. This will appear to be really handy when we will talk about correct answers. Given a sequence $x_1, \ldots, x_n$ of variables, we often write $\exists \vec{x}.G$ as a short form for $\exists x_1 \exists x_2 \ldots \exists x_n.G$.

3 Semantics for sequent-based logics

As we mentioned before, we are interested not only in the operational flavour of proof theory but also in the declarative and fixpoint one. Therefore, we give a general concept of semantics.

$$\frac{\Gamma_1, B, \Gamma_2, C \to D}{\Gamma_1, C, \Gamma_2, B \to D}\ \mathit{interchange} \qquad \frac{\Gamma_1, B, B \to C}{\Gamma_1, B \to C}\ \mathit{contraction}$$

$$\frac{}{\Gamma, B \to B}\ \mathit{id} \qquad \frac{}{\Gamma \to \mathsf{true}}\ \mathsf{true}R$$

$$\frac{\Gamma \to B}{\Gamma \to B \vee C}\ \vee R_1 \quad \frac{\Gamma \to B}{\Gamma \to C \vee B}\ \vee R_2 \quad \frac{\Gamma \to B[t/x]}{\Gamma \to \exists x.B}\ \exists R$$

$$\frac{\Gamma, B_1, B_2 \to C}{\Gamma, B_1 \wedge B_2 \to C}\ \wedge L \qquad \frac{\Gamma \to B \quad \Gamma \to C}{\Gamma \to B \wedge C}\ \wedge R$$

$$\frac{\Gamma \to B \quad \Gamma, C \to E}{\Gamma, B \supset C \to E}\ \supset L \qquad \frac{\Gamma, B \to C}{\Gamma \to B \supset C}\ \supset R$$

$$\frac{\Gamma, B[t/x] \to C}{\Gamma, \forall x.B \to C}\ \forall L \qquad \frac{\Gamma \to B[y/x]}{\Gamma \to \forall x.B}\ \forall R$$

provided that

y in not free in the lower sequent of the rule $(\forall R)$
B is an atomic formula in the rule *id*

Figure 1: Inference rules for hereditarily Harrop formulas

Given a logic $(\mathcal{D}, \mathcal{G}, \mathcal{R})$, a *pre-interpretation* I is a choice of a cpo of denotations $I(\Gamma \to \Delta)$ for each sequent, an element $I(\epsilon_S) \in I(S)$ for each S and a continuos function $I(r)$ for each inference rule r, where if $\mathrm{hyp}(r) = S_1, \ldots, S_n$ and $\mathrm{th}(r) = S$,

$$I(r) = I(S_1) \times \cdots \times I(S_n) \to I(S) \tag{3.1}$$

Given a *pre-interpreted logic* $(\mathcal{D}, \mathcal{G}, \mathcal{R}, I)$, an *interpretation* is a choice of an element $[\![S]\!] \in I(S)$ for each sequent S. The set of interpretations is a cpo with the straightforward induced pointwise ordering. An intepretation is a *model* when, for each inference rule

$$r : S_1, \ldots, S_n \vdash S \in \mathcal{R} \tag{3.2}$$

the following relation holds

$$I(\epsilon_S) \sqcup I(r)([\![S_1]\!], \ldots, [\![S_n]\!]) \leq [\![S]\!] \tag{3.3}$$

This notion of pre-interpretation gives us a great flexibility. By a careful choice of I, it is possible to define a lot of different concrete semantics. For example, let $I(S)$ be the powerset of proof schemas of S, $I(r)$ be the function which glues proof schemas together by using the inference rule r and $I(\epsilon_S) = \{\epsilon_S\}$. Any interpretation built upon I is called *syntactical interpretation*. If we choose $[\![S]\!]$ to be the set of proofs of S, we obtain a model that corresponds to the *Heyting semantics* of logic.

As another example, if $I(S)$ is the set $\{\mathsf{false}, \mathsf{true}\}$ with $\mathsf{false} < \mathsf{true}$, $I(r)$ is the logical "and" of the input truth values, $I(\epsilon_S) = \mathsf{false}$ and $[\![S]\!]$ is true if and only if S is provable, then we obtain another model, the *semantics of provability*, which can be viewed as an abstraction of the Heyting one.

Moreover, both the Heyting semantics and the semantics of provability are the least models in their respective cpos. In general, we call *declarative semantics* the least model of a pre-interpreted logic.

Note, however, that the Heyting semantics is generally given compositionally w.r.t. the goals. Here, on the contrary, we have a "flat" definition. In the following sections we will try to characterize the conditions which must be satisfied in order to give a compositional reformulation of interpretations.

3.1 Correct answer semantics

We want to give a look to our running example and discuss a typical case of semantics widely used in the logic programming area.

In the case of Horn clauses and derived languages, correct answers are perhaps the most important observable of derivations. Real logic language interpreters, such as PROLOG, give correct answers as results for a query. Moreover, correct answers have been proved to be "and compositional" and "instance compositional" [2], thus providing a strong semantic framework for program analysis and diagnosis [5].

Given a sequent $S = \Gamma \to \exists \vec{x}.G$, where the top level connective of G is not an existential quantifier, a *(possible) answer* for S is a substitution for the variables in $\vec{x}$. A *correct answer* is a possible answer θ such that $\Gamma \to G\theta$ has a proof with no hypotheses.

A pre-interpretation I for correct answers can be defined as follows.

- $I(\Gamma \to G)$ is the powerset of all the possible answers for G. When G has no top level existential quantifier, the only possible answer is the empty substitution. Hence $I(\Gamma \to G)$ has only two elements, which can be viewed as logical values representing the existence of a proof (with no hypotheses) for the sequent.
- For all the inference rules r (with the exception of $\exists R$) $I(r)$ is the logical and of the input values (with the usual assumptions that 0-adic and is true and 1-ary and is the identity function). If one of the input domains has more than two elements, the empty set is viewed as false and all the remaining elements are viewed as true.
- $I(\exists R)$ maps an answer θ for $B[t/x]$ to the answer $\theta \cup \{t/x\}$ for $\exists x.B$, for each element of the input set.
- $I(\epsilon_S) = \emptyset$.

The logic of hereditarily Harrop formulas with the pre-interpretation I is denoted by $\mathcal{H}_c$.

It is easy to show that if we take the interpretation $[\![\_]\!]$, such that $[\![\Gamma \to G]\!]$ is the set of correct answers for $\Gamma \to G$, then $[\![\_]\!]$ is a model (actually, it is the least one). We will refer to $[\![\_]\!]$ as the *correct answer semantics*.

3.2 Fixpoint construction

Given a pre-interpretation for the set $\mathcal{R}$ of inference rules, we can build a corresponding model, by using a successive approximation operator similar to the T_P of logic programs. The new operator $T_{\mathcal{R}}$ takes interpretations to interpretations, according to the following definition

$$T_{\mathcal{R}}([\![\_]\!])(S) = I(\epsilon_S) \sqcup \bigsqcup_{r:S_1,\ldots,S_n \vdash S \in \mathcal{R}} I(r)([\![S_1]\!], \ldots, [\![S_n]\!]). \tag{3.4}$$

We can prove that all the results which hold for the T_P operator, apply to $T_{\mathcal{R}}$ as well (see the proofs in appendix A). In particular (see theorem A.1) an interpretation $[\![\_]\!]$ is a model iff it is a prefixpoint of $T_{\mathcal{R}}$. Moreover (theorem A.2) $T_{\mathcal{R}}$ is continuos, hence $T_{\mathcal{R}} \uparrow \omega$ is its least fixpoint. We call $T_{\mathcal{R}} \uparrow \omega$ the *fixpoint semantics* of a given pre-interpreted logic. Since the least fixpoint of a continuos operator is also the least prefixpoint, $T_{\mathcal{R}}$ is the least model of the logic, hence fixpoint and declarative semantics coincide.

4 Compositionality

We want now to prove that the existence of a compositional semantics for a pre-interpreted logic is somewhat related (although not equivalent) to the completeness of uniform proofs. First of all, we must clarify what we mean in general by compositionality in our setting.

We have defined an intepretation as a mapping from sequents S to denotations in $I(S)$. Assume $S_1 = \Gamma \to G_1$ and $S_2 = \Gamma \to G_2$. Given a "partial" interpretation,

which defines $[\![S_1]\!]$ and $[\![S_2]\!]$ only, we would like to automatically derive $[\![\Gamma \to G_1 \wedge G_2]\!]$. In general, there are several kinds of sequents $\Gamma \to \Delta$, whose semantics we want to derive from other sequents $S_1, \ldots, S_n$ with different antecedents and/or consequents.

A *decomposition rule* is defined exactly like an inference rule. However, it is used to specify the kind of decompositions we want to focus our attention on. Given a pre-interpreted logic and a set of decomposition rules, a *decomposition interpretation* I maps each decomposition rule $r : S_1, \ldots, S_n \vdash S$ to a continuos function $I(r)$, defined as if r were an inference rule. A *logic with decompositions* is a pre-interpreted logic together with a set of decompositions and a decomposition interpretation. A sequent S is *decomposable* when there is a decomposition rule rooted at S.

An important case is when the decomposition rules are a subset of the inference rules and the decomposition interpretation is the appropriate restriction of the pre-interpretation (logic with *inference driven decompositions*). Moreover, a logic has *full* inference driven decompositions when, for each decomposable S, all the inference rules rooted at S are decomposition rules.

Given a logic $\mathcal{L}$ with decompositions $\mathcal{K}$, an interpretation $[\![\_]\!]$ is *compositional* if, for each decomposable S

$$\begin{aligned}[\![S]\!] &= I(\epsilon_S) \sqcup \bigsqcup_{r:S_1,\ldots,S_s \vdash S \in \mathcal{K}} I(r)([\![S_1]\!], \ldots [\![S_n]\!]) \\ &= T_{\mathcal{K}}([\![\_]\!])(S)\end{aligned} \tag{4.1}$$

If $\mathcal{L}$ has full inference driven decompositions, (4.1) holds if $[\![\_]\!]$ is its least model (see theorem A.3). In general, a logic with such a property (compositionality of the least model) is said to be *compositional*.

Assume now we know $[\![\_]\!]$ only for the non-decomposable sequents, and assume also we know it to be compositional. This is not enough to be able to reconstruct the full interpretation. The problem is that the set of equations derived from (4.1) do not always uniquely define $[\![\_]\!]$ for the decomposable sequents.

A set of decomposition rules $\mathcal{K}$ induces a reflexive and transitive relation on sequents $\leq_{\mathcal{K}}$. Just set $S' \leq_{\mathcal{K}} S$ if there exists a decomposition rule rooted at S and with S' as one of its hypotheses, and take the appropriate closure. A set $\mathcal{K}$ of decompositions rules is *consistent* when $\leq_{\mathcal{K}}$ is a well-founded ordering. If this is the case, the equations in (4.1) are a good inductive definition for the decomposable sequents.

If $\mathcal{L}$ has consistent decompositions and is compositional, we can build the full least model, by just knowing its restriction to the non-decomposable sequents. In general, given an interpretation $[\![\_]\!]$, we denote by $p([\![\_]\!])$ its restriction to non-decomposable sequents. Moreover, if $[\![\_]\!]$ is a partial interpretation, and the logic has consistent decompositions, we denote by $c([\![\_]\!])$ its *completion*, i.e. the full compositional intepretation derived from (4.1). In formulas

$$c([\![\_]\!])(S) = \begin{cases} S & \text{if } S \text{ is not decomposable} \\ T_{\mathcal{K}}(c([\![\_]\!]))(S) & \text{if } S \text{ is decomposable} \end{cases} \tag{4.2}$$

In a compositional logic with consistent decompositions, we have $[\![\_]\!] = c(p([\![\_]\!]))$ if $[\![\_]\!]$ is the fixpoint semantics.

Another question is whether, in a such a logic, it is possible to define the least model by a successive approximation operator over the cpo of compositional interpretations (see theorem A.4 for a proof that compositional interpretations really form a cpo). We define the new $T^c_{\mathcal{R}}$ operator as follows

$$T^c_{\mathcal{R}}([\![\_]\!])(S) = c(p(T_{\mathcal{R}}([\![\_]\!]))(S) \tag{4.3}$$

Given $[\![\_]\!]' = T^c_{\mathcal{R}}([\![\_]\!])$, equation (4.3) means that $[\![S]\!]' = T_{\mathcal{R}}([\![\_]\!])(S)$ if S is not decomposable. For a decomposable S, $[\![S]\!]'$ is derived from $p([\![S]\!]')$ by using equations (4.1). If $T^c_{\mathcal{R}} \uparrow \omega = T_{\mathcal{R}} \uparrow \omega$, we say that the fixpoint semantics of $\mathcal{L}$ is *compositionally definable*.

We can prove (see theorem A.5) that, given a logic with consistent and full inference driven decomposition rules $\mathcal{L}$, its fixpoint semantics is compositionally definable.

Coming back to our working example, a rather important point in logic programming is the and-compositionality of goals. In the standard setting, a semantics $[\![\_]\!]$ for a logic language is and-compositional when, given goals G_1 and G_2 and a program P, there exists a semantic operator $\odot$, such that $[\![G_1 \wedge G_2]\!]_P = [\![G_1]\!]_P \odot [\![G_2]\!]_P$. In our framework, this corresponds to establish a semantic function $I(\wedge c)$ and a set $\mathcal{K}$ of decomposition rules as instances of the following schema

$$\wedge c : \frac{\Gamma \to \exists \vec{x}.G_1 \quad \Gamma \to \exists \vec{y}.G_2}{\Gamma \to \exists \vec{z}.G_1 \wedge G_2}, \tag{4.4}$$

where $\vec{z}$ is the concatenation of $\vec{x}$ and $\vec{y}$. It is obvious that such a $\mathcal{K}$ is consistent. The difficult step is finding (if it exists) the definition of $I(\wedge c)$, such that $\mathcal{H}_c$ is compositional.

5 Uniform proofs

Let us consider now the relationship betweem uniform proofs and compositionality. First of all, a uniform proof is just a proof that does not use some of the inference rules. For example, in the hereditarily Harrop case, a uniform proof does not contain any instance of the following schema:

$$\frac{\Gamma, B_1, B_2 \to C_1 \wedge C_2}{\Gamma, B_1 \wedge B_2 \to C_1 \wedge C_2} \tag{5.1}$$

Hence, given a logic $(\mathcal{D}, \mathcal{G}, \mathcal{R})$, we can obtain a *uniformed logic* $(\mathcal{D}, \mathcal{G}, \mathcal{R}_O)$ just by choosing the right $\mathcal{R}_O \subseteq \mathcal{R}$. The idea of eliminating inference rules, however, is much broader than the classical concept of uniform proofs. It can be applied in several different ways that have nothing to do with the enforcing of right introduction rules for composite goals.

Let us restrict our attention to logics with intuitionistic sequents only and with a distinguished language $A \subseteq \mathcal{G}$. We call *atomic goals* the elements of A. An inference rule on intuitionistic sequents

$$\frac{\Gamma_1 \to G_1 \quad \cdots \quad \Gamma_n \to G_n}{\Gamma \to G} \tag{5.2}$$

is a *right introduction rule* when all the G_i's are proper substrings of G and G is not atomic. If r is an inference rule rooted at S, such that it is not a right introduction rule and there exists a right introduction rule for S, it is said to be *hidden*. A *uniform proof* is a proof that does not use any hidden rule. A logic is *uniform* when all its proofs are uniform (i.e., there are no hidden rules). Given a logic $(\mathcal{D}, \mathcal{G}, \mathcal{R})$, the *uniformed logic* $(\mathcal{D}, \mathcal{G}, \mathcal{R}_O)$ is obtained by taking $\mathcal{R}_O$ to be $\mathcal{R}$ minus the hidden rules. $(\mathcal{D}, \mathcal{G}, \mathcal{R}_O)$ is uniform.

A uniform logic can be provided with full and consistent inference driven decompositions. Just let $\mathcal{K}$ be the set of right introduction rules (theorem A.6). We have already proved that, under such hypotheses, a logic is compositional and has a compositionally definable fixpoint semantics. Hence, we can say that the semantic counterpart of uniformity is compositionality.

In the case of hereditarily Harrop formulas, with the obvious choice of atomic formulas, the right introduction rules, according to our definition, are exactly the instances of $\mathsf{true}R$, $\wedge R$, $\vee R$, $\exists R$ and $\forall R$, as we expected. However, this is not a uniform logic, since there exist hidden rules (for example, all the rules derived from (5.1)).

5.1 Abstract logic languages

Which is the relationship between the semantics of a logic and the semantics of the corresponding uniformed logic? In the Miller's definition of abstract logic language, a sequent S is provable iff it is provable by uniform proofs. In our general setting, a pre-interpreted logic $\mathcal{L}$ is an *abstract logic programming* language when it has the same least model of the uniformed variant $\mathcal{L}_O$. Our definition corresponds with Miller's when we consider the pre-interpretation of provability.

It is well known [12] that, for the hereditarily Harrop formulas, a sequent is provable iff it has a uniform proof. Therefore, the logic $\mathcal{H}_p$ (hereditarily Harrop formulas with the pre-interpretation of provability) is an abstract logic language. On the contrary, if we consider the syntactical pre-interpretation, this property does not hold anymore. In general, a logic under the syntactical pre-interpretation is an abstract logic language iff it is a uniform logic. Otherwise, the set of uniform proofs for some sequent is a proper subset of the set of all the proofs. As a consequence, the least models in the normal and uniform variant are different.

In summary, given a logic $\mathcal{L}$, our notion of abstract logic language corresponds to Miller's definition under pre-interpretation of provability, and to uniformity of logic under the syntactical pre-interpretations. These two cases can be viewed as the extreme points of a broad set of possible semantics for logics.

In the case of hereditarily Harrop formulas we have seen a third example of pre-interpretation, namely correct answers. It is interesting to know whether $\mathcal{H}_c$ is an abstract logic language or not. If this is the case, it is possible to use uniform proofs not only for proving derivability of a sequent but also to compute its correct answers.

Theorem 5.1 *$\mathcal{H}_c$ is an abstract logic language.*

Proof. We know that the least model of $\mathcal{H}_c$ is an interpretation $[\![\_]\!]$, such that $[\![S]\!]$ is the set of correct answers for S. The same is true for the uniformed variant

$(\mathcal{H}_c)_O$, with the corresponding induced notion of correct answers. The problem is to establish whether the two logics have the same correct answers.

Remember that θ is a correct answer for $S = \Gamma \to \exists \vec{x} G$ in $(\mathcal{H}_c)_O$ iff $\Gamma \to G\theta$ has a proof. Since $\mathcal{H}_p$ is an abstract logic language, $\Gamma \to G\theta$ has a proof iff $\Gamma \to G\theta$ has a proof in $\mathcal{H}_c$, i.e. iff θ is a correct answer for S in $\mathcal{H}_c$. ■

Since uniform logics are compositional, and abstract logic language have the same fixpoint semantics of an uniform logic, abstract logic languages are compositional, too. Also, they have compositionally definable fixpoint semantics. In summary, they seem to share the same compositional properties of uniform logics.

6 Conclusions and future works

In this paper we presented a fixpoint and declarative semantics for sequent calculi, parametric w.r.t. a chosen observable. The aim was filling the gap between semantic methods used in Horn clause logic programming and the proof theoretic approach typical in various extensions to logic program languages [10, 8]. Actually, our semantic framework is general enough to be adapted with small effort to any kind of tree-based calculi.

We used the fixpoint semantics to give a new definition of abstract logic programming. Miller's definition corresponds to ours, under the pre-interpretation of provability. It is shown that hereditarily Harrop formulas make an abstract logic language not only in the old sense, but also under the observables of correct answers.

The major drawback of our approach is perhaps the way in which the observables are built. The definition of a pre-interpretation is a quite arbitrary process, especially for what concerns the inference rules. A possible solution could be sticking to the syntactical pre-interpretation and deriving all the other observables by resorting to the theory of abstract interpretation [6]. This idea is currently under investigation.

Moreover, it would be interesting to know how the general metodology instantiates to other concrete observables, such as resultants [3], or other languages, such as Lolli [10] and Forum [13].

References

[1] K. R. Apt. Introduction to Logic Programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 495–574. Elsevier and The MIT Press, 1990.

[2] 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.

[3] A. Bossi, M. Gabbrielli, G. Levi, and M. C. Meo. An OR-compositional Semantics for Logic Programs. In J.-M. Jacquet, editor, *Constructing Logic Programs*, pages 215–240. John Wiley & Sons Ltd, 1993.

[4] M. Comini, G. Levi, and M. C. Meo. A theory of observables for logic programs. *Information and Computation*, 1999. To appear.

[5] M. Comini, G. Levi, and G. Vitiello. Declarative diagnosis revisited. In J. Lloyd, editor, *Proceedings of the 1995 Int'l Symposium on Logic Programming*, pages 275–287. The MIT Press, 1995.

[6] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proc. Fourth ACM Symp. Principles of Programming Languages*, pages 238–252, 1977.

[7] S. Finkelstein, P. Freyd, and J. Lipton. A new framework for declarative programming. To appear in *Theoretical Computer Science*, 2000.

[8] L. Giordano and A. Martelli. Structuring logic programs: a modal approach. *Journal of Logic Programming*, 21 (2):59–94, 1994.

[9] R. Harrop. Concerning formulas of the types $A \to B \vee C$, $A \to (Ex)B(x)$ in intuitionistic formal systems. *Journal of Symbolic Logic*, pages 27–32, 1960.

[10] J.S. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. *Journal of Information and Computation*, 110(2):327–365, May 1994.

[11] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987. Second edition.

[12] D. Miller. A Logical Analysis of Modules in Logic Programming. *Journal of Logic Programming*, 6:79–108, 1989.

[13] D. Miller. FORUM: a multiple-conclusion specification logic. *Theoretical Computer Science*, 165(1):201–232, 1996.

[14] D. Miller, F. Pfenning, G. Nadathur, and A. Scedrov. Uniform proofs as a foundation for Logic Programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

A Proofs

Theorem A.1 *An interpretation $[\![\_]\!]$ is a model iff it is a pre-fixpoint of $T_{\mathcal{R}}$.*

Proof. Given a sequent S,

$$\begin{aligned} & [\![S]\!] \geq T_{\mathcal{R}}([\![\_]\!])(S) \\ \iff & [\![S]\!] \geq I(\epsilon_S) \sqcup \bigsqcup_{r:S_1,\ldots,S_n \vdash S \in \mathcal{R}} I(r)([\![S_1]\!],\ldots,[\![S_n]\!]) \\ \iff & [\![S]\!] \geq I(\epsilon_S) \sqcup I(r)([\![S_1]\!],\ldots,[\![S_n]\!]) \\ & \quad \text{for each } r : S_1,\ldots,S_n \vdash S \in \mathcal{R} \end{aligned} \tag{A.1}$$

Since this holds for every S, the result follows trivially. ∎

Theorem A.2 *The $T_{\mathcal{R}}$ operator is continuos.*

Proof. Assume we have a chain $(\llbracket\_\rrbracket^i)_{i\in\mathbb{N}}$ of interpretations. If we call $\llbracket\_\rrbracket^\omega$ the upper limit of the chain, it follows:

$$\begin{aligned} T_{\mathcal{R}}(\llbracket\_\rrbracket^\omega)(S) &= I(\epsilon_S) \sqcup \bigsqcup_{r:S_1,\ldots,S_n\vdash S\in\mathcal{R}} I(r)(\llbracket S_1\rrbracket^\omega,\ldots,\llbracket S_n\rrbracket^\omega) \\ &= I(\epsilon_S) \sqcup \bigsqcup_{r:S_1,\ldots,S_n\vdash S\in\mathcal{R}} \Big(\bigsqcup_{i\in\mathbb{N}} I(r)(\llbracket S_1\rrbracket^i,\ldots,\llbracket S_n\rrbracket^i)\Big) \\ &\qquad \text{[since } I(r) \text{ is continuous for each } r] \\ &= \bigsqcup_{i\in\mathbb{N}}\Big(I(\epsilon_s) \bigsqcup_{r:S_1,\ldots,S_n\vdash S\in\mathcal{R}} I(r)(\llbracket S_1\rrbracket^i,\ldots,\llbracket S_n\rrbracket^i)\Big) \\ &= \bigsqcup_{i\in\mathbb{N}} T_{\mathcal{R}}(\llbracket\_\rrbracket^i)(S) \end{aligned} \tag{A.2}$$

and this proves the theorem. ∎

Theorem A.3 *A logic $\mathcal{L}$ with full inference driven decompositions is compositional.*

Proof. We just need to prove that the least model of $\mathcal{L}$ is compositional. If $\mathcal{R}$ is the set of inference rules and $\llbracket\_\rrbracket$ is the least model, we know that $T_{\mathcal{R}}(\llbracket\_\rrbracket) = \llbracket\_\rrbracket$. Therefore, for each decomposable S

$$\begin{aligned} \llbracket S\rrbracket &= T_{\mathcal{R}}(\llbracket\_\rrbracket)(S) \\ &= I(\epsilon_S) \sqcup \bigsqcup_{r:S_1,\ldots,S_n\vdash S\in\mathcal{R}} I(r)(\llbracket S_1\rrbracket,\ldots,\llbracket S_1\rrbracket) \\ &= I(\epsilon_S) \sqcup \bigsqcup_{r:S_1,\ldots,S_n\vdash S\in\mathcal{K}} I(r)(\llbracket S_1\rrbracket,\ldots,\llbracket S_1\rrbracket) \end{aligned} \tag{A.3}$$

since all the inference rules rooted at S are decomposition rules. ∎

Theorem A.4 *If $\mathcal{L}$ is a compositional logic with consistent decompositions, the compositional interpretations for $\mathcal{L}$ form a cpo under the same order relation of interpretations.*

Proof. First of all, we note that every compositional interpretation is the image, through $c \circ p$, of an interpretation. Moreover, $c \circ p$ is a continuos function, hence maps cpo to cpo. As a result, compositional interpretations are a cpo.

In particular, the least compositional interpretation $\llbracket\_\rrbracket^c_\perp$ is defined as

$$\llbracket\_\rrbracket^c_\perp = c(p(\llbracket\_\rrbracket_\perp)) \tag{A.4}$$

where $\llbracket\_\rrbracket_\perp$ is the bottom of the cpo of the interpretations. The lowest upper bound $\bigsqcup^c$ for a chain $\llbracket\_\rrbracket^c_1,\ldots,\llbracket\_\rrbracket^c_n,\ldots$ is

$$\bigsqcup^c_{i\in\mathbb{N}}\llbracket\_\rrbracket^c_i = c(p(\bigsqcup_{i\in\mathbb{N}}\llbracket\_\rrbracket^c_i)) \tag{A.5}$$

∎

Theorem A.5 *If $\mathcal{L}$ is a logic with full and consistent inference driven decompositions, its fixpoint semantics is compositionally definable.*

Proof. Consider the ordering $\leq_{\mathcal{K}}$ where $\mathcal{K}$ is the set of decomposition rules of $\mathcal{L}$. Since $\mathcal{K}$ is consistent, $\leq_{\mathcal{K}}$ has no infinite descending chains. We now prove by transfinite induction on $\leq_{\mathcal{K}}$ that, for each sequent S and for each interpretation $[\![\_]\!]$ such that $T_{\mathcal{R}}([\![\_]\!]) \geq [\![\_]\!]$, it is

$$T_{\mathcal{R}}([\![\_]\!])(S) \leq T^c_{\mathcal{R}}([\![\_]\!])(S) \leq T^\omega_{\mathcal{R}}([\![\_]\!])(S) \tag{A.6}$$

The property trivially holds for non-decomposable sequents, since in this case $T^c_{\mathcal{R}}([\![\_]\!])(S) = T_{\mathcal{R}}([\![\_]\!])(S)$. Assuming that the property holds for all the sequents smaller than S, we prove it for S.

$$\begin{aligned} T^c_{\mathcal{R}}([\![\_]\!])(S) &= c(p(T_{\mathcal{R}}([\![\_]\!])))(S) \\ &= T_{\mathcal{R}}(T^c_{\mathcal{R}}([\![\_]\!]))(S) \end{aligned} \tag{A.7}$$

In the last formula, $T^c_{\mathcal{R}}([\![\_]\!])$ is only evaluated on sequents smaller than S, hence, by inductive hypothesis

$$\begin{aligned} T^c_{\mathcal{R}}([\![\_]\!])(S) &\leq T_{\mathcal{R}}(T^\omega_{\mathcal{R}}([\![\_]\!]))(S) \\ &= T^\omega_{\mathcal{R}}([\![\_]\!])(S) \end{aligned} \tag{A.8}$$

since $T_{\mathcal{R}}$ is continuous. On the other side,

$$T^c_{\mathcal{R}}([\![\_]\!])(S) \geq T^2_{\mathcal{R}}([\![\_]\!]) \geq T_{\mathcal{R}}([\![\_]\!]) \tag{A.9}$$

Now, it is easy to prove, by induction on i, that $T^c_{\mathcal{R}} \uparrow i \leq T_{\mathcal{R}} \uparrow \omega$, hence $T^c_{\mathcal{R}} \uparrow \omega \leq T_{\mathcal{R}} \uparrow \omega$. For $i = 0$, $T^c_{\mathcal{R}} \uparrow 0$ is the least compositional interpretation. Since $\mathcal{L}$ is compositional, $T_{\mathcal{R}} \uparrow \omega$ is compositional, too. Therefore, it is $T^c_{\mathcal{R}} \uparrow 0 \leq T_{\mathcal{R}} \uparrow \omega$. In the inductive step

$$T^c_{\mathcal{R}} \uparrow i \leq T_{\mathcal{R}} \uparrow \omega \Rightarrow T^c_{\mathcal{R}} \uparrow (i+1) \leq T^c_{\mathcal{R}}(T_{\mathcal{R}} \uparrow \omega) \tag{A.10}$$

Since we just proved that

$$T^c_{\mathcal{R}}(T_{\mathcal{R}} \uparrow \omega) \leq T^\omega_{\mathcal{R}}(T_{\mathcal{R}} \uparrow \omega) = T_{\mathcal{R}} \uparrow \omega, \tag{A.11}$$

this concludes the proof.

For what concerns the opposite disequality, we prove by induction on i that $T^c_{\mathcal{R}} \uparrow i \geq T_{\mathcal{R}} \uparrow i$. Again, for $i = 0$ the proof is trivial, since $T_{\mathcal{R}} \uparrow i$ is the bottom of the cpo of the interpretations. If the propery holds for i, it is

$$T^c_{\mathcal{R}} \uparrow (i+1) \geq T^c_{\mathcal{R}}(T_{\mathcal{R}} \uparrow i) \geq T_{\mathcal{R}} \uparrow (i+1) \tag{A.12}$$

In summary, we have $T_{\mathcal{R}} \uparrow \omega \leq T^c_{\mathcal{R}} \uparrow \omega \leq T_{\mathcal{R}} \uparrow \omega$. It is straightforward to conclude that $T_{\mathcal{R}} \uparrow \omega = T^c_{\mathcal{R}} \uparrow \omega$. ∎

Theorem A.6 *Given a uniform pre-interpreted logic $\mathcal{L}$, there is an induced uniform logic with full and consistent inference driven decompositions, obtained by just taking $\mathcal{K}$ to be the set of the right introduction rules.*

Proof. We start by proving fullness of the inference rules. If S is decomposable, there is a decomposition rule r rooted at S. Thanks to the choice of $\mathcal{K}$, r is a right introduction rule. Since $\mathcal{L}$ is uniform, all the inference rules rooted at S are right introduct rules, hence they are in $\mathcal{K}$. It trivially follows that $\mathcal{L}$ has full inference driven decompositions.

We prove now the consistency of $\mathcal{K}$. First of all, $\leq_{\mathcal{K}}$ is an order relation, since it enjoys the anti-simmetric property. Given two sequents $S = \Gamma \rightarrow G$ and $S' = \Gamma' \rightarrow G'$, if $S < S'$ and $S' < S$ there are two right introduction rules r and r'

$$r : \frac{\cdots\ \Gamma \rightarrow G\ \cdots}{\Gamma' \rightarrow G'} \qquad r' : \frac{\cdots\ \Gamma' \rightarrow G'\ \cdots}{\Gamma \rightarrow G} \tag{A.13}$$

such that G is proper substring of G' and G' is a proper substring of G. This is trivially an absurd, hence the anti-simmetric property holds.

Now, consider an infinite descending chain $S_0 >_{\mathcal{K}} S_1 >_K > \ldots$ of sequents, where $S_i = \Gamma_i \rightarrow G_i$. If l_i is the length of G_i, we have an induced descending chain $l_0 > l_1 > \ldots$. This is an absurd, since natural numbers are well founded, hence $\leq_{\mathcal{K}}$ has no infinite descendic chains either. As a result, $\leq_{\mathcal{K}}$ is a well founded order relation. ■