A Fixed Point Semantics for an Extended *CLP* Language

Miguel García-Díaz and Susana Nieva *

Departamento de Sistemas Informáticos y Programación
Universidad Complutense de Madrid, Spain
{miguel,nieva}@sip.ucm.es

Abstract. In [10] an extension of traditional Logic Programming was introduced combining two improving approaches. On one hand, extending Horn logic to hereditary Harrop formulas, in order to enhance the expressive power, and on the other incorporating constraints, in order to improve the efficiency. The constraint logic programming language obtained from such combination was in need of a declarative semantics. In this paper, we present a fixed point semantics for it. Taking as starting point the technique used by Miller to interpret intuitionistic implication in goals, we have formulated a novel extension to deal with universal quantifiers and constraints. The corresponding theorems of soundness and completeness are proved.

1 Introduction

One of the main features of Logic Programming (*LP*) is that, in a logic program, the operational interpretation and the mathematical (declarative) meaning agree each other, in the sense that the declarative meaning of a program can be interpreted operationally as a goal-oriented search for solutions. In [13] the notion of abstract logic programming language is formulated as a formalization of this idea. There the declarative meaning of a program is identified with the set of goals that can be proved from it by means of uniform proofs in a deduction system. Several logic extensions of Horn logic, both of first and higher order, have been proved to be abstract logic programming languages that enhance the weak expressive power of logic programs based on Horn clauses ([13,14]). This is also the case of the language $HH(\mathcal{C})$, on which the present paper focuses. It was introduced in [10] as a combination of the logic of Hereditary Harrop formulas (*HH*) and Constraint Logic Programming (*CLP*), obtaining a scheme $HH(\mathcal{X})$ that may be particularized with any constraint system $\mathcal{C}$, providing for an instance $HH(\mathcal{C})$. This language is not only an extension of traditional *LP* (based on Horn logic) improving its expressivity, but also incorporating the efficiency advantages of *CLP* [7]. *HH* extends Horn logic including disjunctions, intuitionistic implications and universal quantifiers in goals. These constructions are essential in capturing module structure, hypothetical queries and data abstraction. On the other hand, the purpose of the incorporation of the *CLP* approach is to overcome the inherent limitations in dealing efficiently with elements of domains different from Herbrand terms. Satisfiability of constraints of particular domains may be checked in an efficient way, apart from the logic. In [6] an interesting and useful constraint system that combines real numbers with Herbrand terms is presented as an instance of our scheme.

* The authors are partially supported by the Spanish project TIC2002-01167 'MELODIAS'.

In addition, in [10] a goal solving procedure for the scheme $HH(\mathcal{C})$ was presented and it is proved to be sound and complete w.r.t. the intuitionistic deduction system $\mathcal{UC}$, previously defined. This goal solving procedure could be considered an operational semantics of $HH(\mathcal{C})$.

Of course, an operational interpretation is needed in order to specify programs which can be executed with certain efficiency. But a clear declarative semantics would simplify the programmer's work. If the deduction system is supported by model-theoretic semantics involving more abstract elements, then additional properties of programs can be analyzed in a formal way. The attempts to provide declarative semantics for *LP* languages based on mathematical foundations are extensive and fruitful (see i.e. [11,2,3]). This is also the case of *CLP* [8,5]. In both, *LP* and *CLP*, most of the studies are based on fixed point theories, in which it is easy to define program analysis frameworks.

The aim of the present work is to define a fixed point semantics for $HH(\mathcal{C})$. That definition is inspired in the semantics for a fragment of *HH* described in [12]. Our purpose is to find a model such that for any program Δ, finite set of constraints Γ and goal G, G can be proved in the deduction system $\mathcal{UC}$, if and only if, G is satisfied in that model in the context $\langle \Delta, \Gamma \rangle$. However, in order to build such model, it is important to realize that, during the search of a proof for a goal from a program Δ and a set of constraints Γ, both Δ and Γ may grow. So we will identify the notion of interpretation with functions that associate to every pair $\langle \Delta, \Gamma \rangle$ a set of "true" atoms, in such a way that, if Δ or Γ are augmented, the set of true atoms cannot decrease. The model we are looking for will be the least fixed point of a continuous operator that transforms such interpretations.

The rest of this paper is organized as follows: Section 2 gathers the syntax of constraint systems, as well as the syntax of programs and goals of $HH(\mathcal{C})$, and shows some examples of its use as logic programming language. In Section 3 we recapitulate the definition of the proof system $\mathcal{UC}$ in $HH(\mathcal{C})$, which permits only uniform proofs of goals from programs and constraints. Section 4 contains the main new results of the paper. A fixed point semantics for $HH(\mathcal{C})$ is presented, and soundness and completeness results are obtained. The proofs are compressed, but they can be found extended in the Appendix. Finally a new version of this semantics for an interesting class of constraint systems is summarized. In Section 5 related works and future research lines are commented.

2 The programming language $HH(\mathcal{C})$

The purpose of the present section is to briefly describe the syntax of $HH(\mathcal{C})$, introduced in [10]. $HH(\mathcal{C})$ can be regarded as a constraint logic programming language, not founded in Horn logic, as usual, but in the extended logic of hereditary Harrop formulas. As most *CLP* languages, it is in fact a parameterized scheme that can be instantiated by particular constraint systems. The requirements imposed to such generic constraint systems are gathered below.

Given a signature Σ, containing constants, function and predicate symbols, including the equality predicate $\approx$, a constraint system $\mathcal{C}$ over Σ is a

pair $(\mathcal{L}_\mathcal{C}, \vdash_\mathcal{C})$, where $\mathcal{L}_\mathcal{C}$ is the set of formulas that play the role of constraints, and $\vdash_\mathcal{C} \subseteq \mathcal{P}(\mathcal{L}_\mathcal{C}) \times \mathcal{L}_\mathcal{C}$ [1] is the entailment or deduction relation between sets of constraints and constraints. $\mathcal{C}$ must fulfill the following conditions:

- $\mathcal{L}_\mathcal{C}$ is a set of first-order formulas built up using the signature Σ, which must specifically include $\top$ (true), $\bot$ (false), and the equations $t \approx t'$ for any Σ-terms t and t'.
- $\mathcal{L}_\mathcal{C}$ is closed under $\wedge, \Rightarrow, \exists, \forall$ and the application of substitutions of terms for variables.
- $\vdash_\mathcal{C}$ is *compact*, i.e., $\Gamma \vdash_\mathcal{C} C$ iff $\Gamma_0 \vdash_\mathcal{C} C$ for some finite $\Gamma_0 \subseteq \Gamma$. $\vdash_\mathcal{C}$ is also *generic*, i.e., $\Gamma \vdash_\mathcal{C} C$ implies $\Gamma\sigma \vdash_\mathcal{C} C\sigma$ for any substitution σ. $\Gamma\sigma$ is the result of applying the substitution σ to each formula in Γ, avoiding the capture of free variables.
- All the inference rules related to $\wedge, \Rightarrow, \exists, \forall$ and $\approx$ valid in the intuitionistic fragment of first-order logic are also valid to infer entailments in the sense of $\vdash_\mathcal{C}$.

Hereafter, we will consider a fixed signature Σ and a constraint system $\mathcal{C}$ over Σ. C will stand for $\mathcal{C}$-constraints and Γ for finite sets of $\mathcal{C}$-constraints. $\bigwedge \Gamma$ stands for the conjunction of constraints of Γ.

Let the *set of program predicate symbols* Π_P be a set of predicate symbols such that $\Sigma \cap \Pi_P = \emptyset$. In the rest of the paper Σ and Π_P are assumed fixed. Let At be the set of atomic formulas built up with the predicate symbols in Π_P and Σ-terms. The set $\mathcal{G}$ of *goals* G, and the set $\mathcal{D}$ of *clauses* D over Σ and Π_P are defined by the mutually-recursive rules below. Notice that constraints can be found embedded in goals and clauses.

$G ::= A \mid C \mid G_1 \wedge G_2 \mid G_1 \vee G_2 \mid D \Rightarrow G \mid C \Rightarrow G \mid \exists x G \mid \forall x G,$

$D ::= A \mid G \Rightarrow A \mid D_1 \wedge D_2 \mid \forall x D,$

where $A \in At$.

A program over Σ and Π_P is a finite subset of $\mathcal{D}$. The symbol Δ will be used for programs. Let $\mathcal{W}$ be the set of programs over Σ and Π_P.

The following definition will be useful in order to simplify the usage of program clauses.

Definition 1. Given a clause D, the set of its elaborations, $elab(D)$, is the set of clauses defined by the following rules:

- $elab(A) \stackrel{\text{def}}{=} \{\top \Rightarrow A\}$.
- $elab(D_1 \wedge D_2) \stackrel{\text{def}}{=} elab(D_1) \cup elab(D_2)$.
- $elab(G \Rightarrow A) \stackrel{\text{def}}{=} \{G \Rightarrow A\}$.
- $elab(\forall x D) \stackrel{\text{def}}{=} \{\forall x D' \mid D' \in elab(D)\}$.

This definition is naturally extended to sets $S \subseteq \mathcal{D}$: $elab(S) \stackrel{\text{def}}{=} \bigcup_{D \in S} elab(D)$. The clauses of $elab(S)$ for any S are said to be *elaborated*. Notice that elaborated clauses have always the form $\forall \overline{x}(G \Rightarrow A)$ [2]. A *variant* of $\forall \overline{x}(G \Rightarrow A)$ is a clause $\forall \overline{y}((G \Rightarrow A)[\overline{y}/\overline{x}])$, where no $y \in \overline{y}$ occurs in $G \Rightarrow A$. $F[\overline{y}/\overline{x}]$ is the result of applying to F the substitution that replaces x_i by y_i for each $x_i \in \overline{x}$.

[1] Here and in the rest of the paper, given a set S, $\mathcal{P}(S)$ denotes its powerset.

[2] $\forall \overline{x}$ is an abbreviation for $\forall x_1 \ldots \forall x_n$, and analogously for $\exists \overline{x}$.

One of the outstanding features of the logic programming language $HH(\mathcal{C})$ is its high expressive power. In order to illustrate it, a couple of examples is presented below, in a Prolog-like notation, enriched with the logic symbols $\forall$ and $\Rightarrow$.

Example 1. Let us consider the constraint system $\mathcal{R}$, consisting of the field of real numbers with the usual arithmetic, and predicate symbols $=, <, >, \leq$ and $\geq$. Taking $\Pi_P = \{\texttt{triangle}, \texttt{isosceles}\}$, let us consider the singleton program Δ_1 with the clause:

```
triangle(A, B, C):- A > 0, B > 0, C > 0,
                    A < C + B, B < A + C, C < A + B.
```

The variables `A`, `B` and `C` are intended to be lengths, so that the predicate `triangle(A,B,C)` becomes true when it is possible to build a triangle with sides `A`, `B` and `C`. Let Δ_2 be the program:

```
isosceles(A, B, C):- triangle(A, B, C), A = B.
isosceles(A, B, C):- triangle(A, B, C), A = C.
isosceles(A, B, C):- triangle(A, B, C), B = C.
```

Suppose we want to know which conditions over Y guarantee that, for any $X > 1$, it is possible to build an isosceles triangle with sides $\langle X, X, Y\rangle$. The goal which captures that query is:

$$G \equiv (\Delta_2 \Rightarrow \forall \texttt{X (X>1}\Rightarrow\texttt{isosceles(X, X, Y))}.$$

In G, similarly as in [12], the program Δ_2 is being used as a *module* that is loaded over Δ_1 when solving G. Notice that such goal cannot be written in *CLP* languages based on Horn clauses, because the connectives $\Rightarrow$ and $\forall$ would not be allowed in goals. Given the program Δ_1 and the goal G, according to the proof theory that will be described in Section 3, `0<Y`$\wedge$`Y<=2` is a correct answer constraint for G from Δ_1.

Example 2. This example shows an efficient and reversible program to compute Fibonacci numbers. It is borrowed from [10]. The constraint system used is $\mathcal{R}$ again.

```
fib(N,X):- memfib(0, 1) ⇒
     (memfib(1, 1) ⇒ getfib(N, X, 1)).
getfib(N, X, M):- 0 <= N, N <= M, memfib(N, X).
getfib(N, X, M):- N > M, memfib(M-1, F1), memfib(M, F2),
     (memfib(M + 1, F1 + F2) ⇒ getfib(N, X, M + 1)).
```

The goal `getfib(N,X,M)` computes the `N`-th Fibonacci number in `X`, assuming that the Fibonacci numbers fib_i, with $0 \leq i \leq$ `M`, are stored in the local program as atoms for `memfib`. During the computation, atoms `memfib` for fib_i, with `M`$< i \leq$`N`, are locally memorized.

Other examples can be found in [10,9,6]. The ones in [9] belong to the higher-order version of $HH(\mathcal{C})$, and those in [6] to the instance $HH(\mathcal{RH})$.

3 The calculus $\mathcal{UC}$

We follow the ideas of Miller et al. [13], in which logic programming languages are identified with those such that non-uniform proofs of goals in a deduction system can be discarded. Those languages are called abstract logic programming languages. This characterization captures the fact that in *LP* the search of a proof for a goal is directed by the structure of such goal. For the classical and intuitionistic logics there are known deduction systems, based on sequent calculus, such that certain fragments of those logics have been proved to be abstract logic programming languages, w.r.t. them. For the case of $HH(\mathcal{C})$, the existence of constraints implies the necessity of a different calculus. In [10] a sequent calculus that combines intuitionistic rules for the logic connectives with the entailment relation $\vdash_{\mathcal{C}}$ is presented, and it is proved to be equivalent to another calculus, designated by $\mathcal{UC}$, such that every $\mathcal{UC}$-proof is uniform, therefore demonstrating that $HH(\mathcal{C})$ is also an abstract logic programming language.

The calculus $\mathcal{UC}$ is now briefly described. $\mathcal{UC}$ consists of the set of deduction rules below. For any program Δ, finite set of constraints Γ, and goal G, the notation $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ stands for the assertion that there is a proof for the sequent $\Delta; \Gamma \vdash\!\!- G$ using, in a bottom-up fashion, the rules of the calculus $\mathcal{UC}$. So $\mathcal{UC}$-proofs will be regarded as trees.

$\mathcal{UC}$-Rules

Rules for constraints and atomic goals:

$$\frac{\Gamma \vdash_{\mathcal{C}} C}{\Delta; \Gamma \vdash\!\!- C}\ (C_R) \qquad \frac{\Delta; \Gamma \vdash\!\!- \exists \overline{x}(A \approx A' \wedge G)}{\Delta; \Gamma \vdash\!\!- A}\ (Clause)$$

where $\forall \overline{x}(G \Rightarrow A')$ is a variant of some clause in $elab(\Delta)$; the variables of $\overline{x}$ do not occur free in the lower sequent; $A \equiv P(t_1, \ldots, t_n)$, $A' \equiv P(s_1, \ldots, s_n)$, and $A \approx A'$ denotes the conjunction $t_1 \approx s_1 \wedge \ldots \wedge t_n \approx s_n$.

Rules introducing connectives:

$$\frac{\Delta; \Gamma \vdash\!\!- G_1 \quad \Delta; \Gamma \vdash\!\!- G_2}{\Delta; \Gamma \vdash\!\!- G_1 \wedge G_2}\ (\wedge_R) \qquad \frac{\Delta; \Gamma \vdash\!\!- G_i}{\Delta; \Gamma \vdash\!\!- G_1 \vee G_2}\ (\vee_R), i \in \{1, 2\}$$

$$\frac{\Delta, D; \Gamma \vdash\!\!- G}{\Delta; \Gamma \vdash\!\!- D \Rightarrow G}\ (\Rightarrow_R) \qquad \frac{\Delta; \Gamma, C \vdash\!\!- G}{\Delta; \Gamma \vdash\!\!- C \Rightarrow G}\ (\Rightarrow_{C_R})$$

$$\frac{\Delta; \Gamma, C \vdash\!\!- G[y/x] \quad \Gamma \vdash_{\mathcal{C}} \exists y C}{\Delta; \Gamma \vdash\!\!- \exists x G}\ (\exists_R) \qquad \frac{\Delta; \Gamma \vdash\!\!- G[y/x]}{\Delta; \Gamma \vdash\!\!- \forall x G}\ (\forall_R)$$

In rules $(\exists_R)$ and $(\forall_R)$ the variable y does not occur free in any formula of the lower sequent.

When $\Delta; C \vdash_{\mathcal{UC}} G$ holds, if C is satisfiable, it is said to be a *correct answer constraint* for G from Δ.

In [10] a goal solving procedure for $HH(\mathcal{C})$ is introduced and proved to be sound and complete w.r.t the deducibility $\vdash_{\mathcal{UC}}$.

4 Fixed Point Semantics

The goal solving procedure defined in [10] may be regarded as an operational semantics for $HH(\mathcal{C})$. However, from the theoretical point of view, the programming language $HH(\mathcal{C})$ presented lacks a declarative semantics. The only meanings that we may associate to programs, so far, are sets of proofs. In addition, having in mind that $\mathcal{UC}$ is not a traditional sequent calculus due to the presence of constraints, its correspondence with any of the known logical inference relations ($\models$) cannot be direct, and the definition of a specific model-theoretic semantics merging the intuitionistic behavior of HH and the interpretation of constraints is a hard task.

In this section, alternative semantics based on a fixed point construction, widely utilized in LP and CLP, are introduced. For the traditional LP language, given a program P there is a continuous operator T_P transforming models (sets of atoms) such that a goal G can be proved from P, if and only if, G "is true" in the least fixed point of T_P [17]. As analyzed in [12], for the fragment of HH that includes implications in goals, the situation is more complex, since while building a proof for a goal G the program Δ may be augmented. Therefore programs play the role of contexts, and interpretations become monotonous functions mapping each program into a set of atoms. Instead of a family $\{T_\Delta\}_{\Delta\in\mathcal{W}}$ of continuous operators, there is a unique operator T, and the main result is that G can be proved from Δ, if and only if, G "is true" in the least fixed point of T at the context Δ. In the present paper we extend this approach for the language $HH(\mathcal{C})$. New difficulties arise since the universal quantifier, as well as constraints, are allowed in goals, and then embedded into programs. When proving a goal G from a program Δ there is also the presence of a set of constraints Γ; both Δ and Γ may result augmented, therefore the notion of context is extended to pairs $\langle\Delta,\Gamma\rangle$. So an interpretation of Δ and Γ should depend on interpretations of $\langle\Delta',\Gamma'\rangle$ with $\Delta'\subseteq\Delta$, $\Gamma'\subseteq\Gamma$. In this reason, interpretations are defined as monotonous functions able to interpret every pair $\langle\Delta,\Gamma\rangle$. A continuous operator transforming such interpretations is defined. We prove that for any Δ,Γ and G, $\Delta;\Gamma\vdash_{\mathcal{UC}} G$ if and only if G is satisfied by the least fixed point of this operator at the context $\langle\Delta,\Gamma\rangle$.

4.1 Interpretations

Let us assume that Σ, Π_P, a constraint system $\mathcal{C}$ over Σ and a set Π_P of program predicate symbols have been chosen.

Definition 2. An *interpretation* I is a function $I:\mathcal{W}\times\mathcal{P}(\mathcal{L}_\mathcal{C})\to\mathcal{P}(At)$ that is monotonous, i. e. for any Δ_1,Δ_2 and Γ_1,Γ_2 such that $\Delta_1\times\Gamma_1\subseteq\Delta_2\times\Gamma_2$, $I(\Delta_1,\Gamma_1)\subseteq I(\Delta_2,\Gamma_2)$ holds. Let $\mathcal{I}$ denote the set of interpretations.

So, interpretations are notions of truth. $I(\Delta,\Gamma)$ is the set of atoms that are true for I in the context $\langle\Delta,\Gamma\rangle$. The property that, when $\langle\Delta,\Gamma\rangle$ becomes greater, the set of true atoms cannot decrease, is guaranteed by the monotonicity of interpretations.

Definition 3. For any $I_1, I_2 \in \mathcal{I}$, $I_1 \sqsubseteq I_2$ if for each Δ and Γ, $I_1(\Delta, \Gamma) \subseteq I_2(\Delta, \Gamma)$ holds.

It is straightforward to check that $(\mathcal{I}, \sqsubseteq)$ is a poset, i. e. $\sqsubseteq$ is a partial order.

However, in order to define fixed point semantics, the set of interpretations needs to be a complete lattice, not just a poset.

Lemma 1. *The poset $(\mathcal{I}, \sqsubseteq)$ is a complete lattice. Furthermore, given $S \subseteq \mathcal{I}$, its least upper bound and greatest lower bound, denoted by $\bigsqcup S$ and $\bigsqcap S$, are characterized by the following equations:*

$$(\bigsqcup S)(\Delta, \Gamma) = \bigcup_{I \in S} I(\Delta, \Gamma) \text{ for any } \Delta \text{ and } \Gamma,$$
$$(\bigsqcap S)(\Delta, \Gamma) = \bigcap_{I \in S} I(\Delta, \Gamma) \text{ for any } \Delta \text{ and } \Gamma.$$

Proof. The claim follows from the fact that $\mathcal{I}$ is a set of monotonic functions whose range, $\mathcal{P}(At)$, is a complete lattice. □

As a particular case, $(\mathcal{I}, \sqsubseteq)$ has an infimum $\bigsqcap \mathcal{I}$, denoted by $I_\perp$, the constant function $\emptyset$.

The following definition formalizes the notion of a goal G being "true" for an interpretation I in a context $\langle \Delta, \Gamma \rangle$.

Definition 4. Given $I \in \mathcal{I}$, Δ and Γ, a goal G *is forced* by I, Δ and Γ if $I, \Delta, \Gamma \Vdash G$, where $\Vdash$ is the relation recursively defined depending on the structure of G, as follows:

- $I, \Delta, \Gamma \Vdash C \overset{\text{def}}{\Longleftrightarrow} \Gamma \vdash_{\mathcal{C}} C$.
- $I, \Delta, \Gamma \Vdash A \overset{\text{def}}{\Longleftrightarrow} A \in I(\Delta, \Gamma)$.
- $I, \Delta, \Gamma \Vdash G_1 \wedge G_2 \overset{\text{def}}{\Longleftrightarrow} I, \Delta, \Gamma \Vdash G_i$ for each $i \in \{1, 2\}$.
- $I, \Delta, \Gamma \Vdash G_1 \vee G_2 \overset{\text{def}}{\Longleftrightarrow} I, \Delta, \Gamma \Vdash G_i$ for some $i \in \{1, 2\}$.
- $I, \Delta, \Gamma \Vdash D \Rightarrow G \overset{\text{def}}{\Longleftrightarrow} I, \Delta \cup \{D\}, \Gamma \Vdash G$.
- $I, \Delta, \Gamma \Vdash C \Rightarrow G \overset{\text{def}}{\Longleftrightarrow} I, \Delta, \Gamma \cup \{C\} \Vdash G$.
- $I, \Delta, \Gamma \Vdash \exists x G \overset{\text{def}}{\Longleftrightarrow}$ there is a constraint C and a variable y such that:
  - y does not occur free in Δ, Γ, $\exists x G$.
  - $\Gamma \vdash_{\mathcal{C}} \exists y C$.
  - $I, \Delta, \Gamma \cup \{C\} \Vdash G[y/x]$.
- $I, \Delta, \Gamma \Vdash \forall x G \overset{\text{def}}{\Longleftrightarrow}$ there is a variable y such that:
  - y does not occur free in Δ, Γ, $\forall x G$.
  - $I, \Delta, \Gamma \Vdash G[y/x]$.

Since the deduction system $\mathcal{UC}$ has a constraint-oriented formulation, when a proof of an existential quantified goal $\exists x G$ must be found, instead of guessing a witness of x, by means of a substitution $[t/x]$ or by the counterpart constraint $x \approx t$, some extra generality is necessary. Any satisfiable constraint C may be considered,that represents a property characterizing x, v.g. $x^2 \approx 2$. The semantics of a program provides for information regarding the goals which can be proved from it. So, the definition of the forcing relation for the case $\exists x G$ should exhibit the same generality of the rule $(\exists_R)$.

Defined this way, the relation $\Vdash$ has several properties that will help us to prove other more significant results.

Lemma 2. *If $I_1, I_2 \in \mathcal{I}$ and $I_1 \sqsubseteq I_2$, then for any G, Δ, and Γ, $I_1, \Delta, \Gamma \models G$ implies $I_2, \Delta, \Gamma \models G$.*

Proof. The proof is inductive on the structure of G. Only a few cases are considered here, the rest can be found in the Appendix.

- $G \equiv A$, atomic goal. $I_1, \Delta, \Gamma \models A \iff A \in I_1(\Delta, \Gamma)$. $I_1 \sqsubseteq I_2$ implies that $I_1(\Delta, \Gamma) \subseteq I_2(\Delta, \Gamma)$, so $A \in I_2(\Delta, \Gamma)$ and therefore $I_2, \Delta, \Gamma \models A$.
- $G \equiv \forall x G'$. $I_1, \Delta, \Gamma \models \forall x G' \iff$ there is a variable y such that: y does not occur free in Δ, Γ, $\forall x G'$ and $I_1, \Delta, \Gamma \models G'[y/x]$. By induction hypothesis, it holds that $I_2, \Delta, \Gamma \models G'[y/x]$, so $I_2, \Delta, \Gamma \models \forall x G'$. □

The following lemma states a usual property of this kind of semantic approach.

Lemma 3. *Let $\{I_i\}_{i\geq 0}$ be a denumerable family of interpretations such that $I_0 \sqsubseteq I_1 \sqsubseteq I_2 \sqsubseteq \ldots$, and let G be a goal. Then, for any Δ and Γ,*
$$\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \models G \Rightarrow \exists k \geq 0 \text{ such that } I_k, \Delta, \Gamma \models G.$$

Proof. We already know that $\bigsqcup_{i\geq 0} I_i(\Delta, \Gamma) = \bigcup_{i\geq 0} I_i(\Delta, \Gamma)$. The proof is inductive on the structure of G. Here we deal with a few cases, the rest can be found in the Appendix.

- $G \equiv C \in \mathcal{L}_{\mathcal{C}}$. $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \models C \iff \Gamma \vdash_{\mathcal{C}} C \iff I_k, \Delta, \Gamma \models C$, independently of k.
- $G \equiv C \Rightarrow G'$. $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \models C \Rightarrow G' \iff \bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \cup \{C\} \models G'$. By induction hypothesis, there is $k \geq 0$ such that $I_k, \Delta, \Gamma \cup \{C\} \models G'$. Therefore, $I_k, \Delta, \Gamma \models C \Rightarrow G'$. □

As it has been mentioned before, the semantic approach we are formulating is based in searching for a model such that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ iff G is true in such model in the context $\langle \Delta, \Gamma \rangle$. We have shown that each interpretation provides for a version of truth of goals in such contexts. The next step is to define an operator over interpretations whose least fixed point supplies the desired version of truth.

Definition 5. The *operator* $T : \mathcal{I} \longrightarrow \mathcal{I}$ transforms interpretations as follows. For any $I \in \mathcal{I}$, Δ, Γ and $A \in At$, $A \in T(I)(\Delta, \Gamma)$ if there is a variant $\forall \overline{x}(G \Rightarrow A')$ of a clause in $elab(\Delta)$ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $I, \Delta, \Gamma \models \exists \overline{x}(A \approx A' \wedge G)$.

Lemma 4 (Monotonicity of T). *Let $I_1, I_2 \in \mathcal{I}$ such that $I_1 \sqsubseteq I_2$. Then, $T(I_1) \sqsubseteq T(I_2)$.*

Lemma 5 (Continuity of T). *Let $\{I_i\}_{i\geq 0}$ be a denumerable family of interpretations such that $I_0 \sqsubseteq I_1 \sqsubseteq I_2 \sqsubseteq \ldots$. Then $T(\bigsqcup_{i\geq 0} I_i) = \bigsqcup_{i\geq 0} T(I_i)$.*

Proof. Let us deal with both inclusions.

$\supseteq$) This inclusion is always a consequence of the monotonicity of T.

$\subseteq$) Let Δ, Γ and $A \in T(\bigsqcup_{i\geq 0} I_i)(\Delta, \Gamma)$. Due to the definition of T, there is a variant $\forall \overline{x}(G \Rightarrow A')$ of a clause of $elab(\Delta)$ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \models \exists \overline{x}(A \approx A' \wedge G)$. Thanks to Lemma 3, there exists $k \geq 0$ such that $I_k, \Delta, \Gamma \models \exists \overline{x}(A \approx A' \wedge G)$, and therefore $A \in T(I_k)(\Delta, \Gamma)$. So, we have proved that $T(\bigsqcup_{i\geq 0} I_i)(\Delta, \Gamma) \subseteq \bigcup_{k\geq 0} T(I_k)(\Delta, \Gamma) = (\bigsqcup_{k\geq 0} T(I_k))(\Delta, \Gamma)$, for any Δ and Γ, thus $T(\bigsqcup_{i\geq 0} I_i) \sqsubseteq \bigsqcup_{k\geq 0} T(I_k)$. $\square$

Theorem 1. *The operator T has a least fixed point, which is $\bigsqcup_{i\geq 0} T^i(I_\perp)$.*

Proof. The claim is an immediate consequence of Lemmas 4 and 5, and the Knaster-Tarski fixed point theorem. $\square$

From now on, $lfp(T)$ denotes the least fixed point of T.

Example 3. Let Δ_3 be the program in Example 2. Figure 1 shows some of the goals that are forced by the first interpretations $T^i(I_\perp)$ in the contexts $\langle \Delta, \Gamma \rangle$, where $\Gamma = \{z_1 = 1, z_2 = 1, x = z_1 + z_2\}$, $\Delta_3' = \Delta_3 \cup \{mf(0,1), mf(1,1)\}$ and $\Delta_3'' = \Delta_3 \cup \{mf(0,1), mf(1,1), mf(2, z_1 + z_2)\}$.

$\langle \Delta, \Gamma \rangle$	$T(I_\perp)$	$T^2(I_\perp)$	$T^3(I_\perp)$	$T^4(I_\perp)$	$T^5(I_\perp)$
$\langle \Delta_3, \Gamma \rangle$	$\ldots$	$\ldots$	$\ldots$	$mf(0,1) \Rightarrow$	$fib(2,x)$
	$\ldots$	$\ldots$	$\ldots$	$(mf(1,1) \Rightarrow gf(2,x,1))$	$\ldots$
$\langle \Delta_3', \Gamma \rangle$	$mf(0,z_1)$,	$\ldots$	$mf(2, z_1 + z_2)$	$gf(2,x,1)$	$\ldots$
	$mf(1,z_2)$	$\ldots$	$\Rightarrow gf(2,x,2)$	$\ldots$	$\ldots$
$\langle \Delta_3'', \Gamma \rangle$	$mf(2,x)$	$gf(2,x,2)$	$\ldots$	$\ldots$	$\ldots$

Fig. 1. Steps leading to $T^5(I_\perp), \Delta_3, \Gamma \models fib(2,x)$.

The chart contains the main steps leading to $T^5(I_\perp), \Delta_3, \Gamma \models fib(2,x)$. *memfib* is abbreviated with *mf*, and *getfib* with *gf*.

4.2 Soundness and Completeness

The following theorem states the soundness and completeness we were looking for, establishing the full connection between the fixed point semantics presented and the calculus $\mathcal{UC}$. The definitions below will be used in its proof.

Let $\mathcal{S} = \{\langle \Delta, \Gamma, G \rangle \in \mathcal{W} \times \mathcal{P}(\mathcal{L}_\mathcal{C}) \times \mathcal{G} \mid lfp(T), \Delta, \Gamma \models G\}$. We define the function $ord : \mathcal{S} \longrightarrow \mathbb{N}$ as follows. Given any $\langle \Delta, \Gamma, G \rangle \in \mathcal{S}$, Lemma 3 guarantees that the set of natural numbers k such that $T^k(I_\perp), \Delta, \Gamma \models G$ is nonempty. Therefore, it is possible to define $ord(\langle \Delta, \Gamma, G \rangle)$ as the least element of such set. Let us consider the partial order $(\mathcal{S}, <)$ defined as follows. Given any $\langle \Delta_1, \Gamma_1, G_1 \rangle, \langle \Delta_2, \Gamma_2, G_2 \rangle \in \mathcal{S}$, $\langle \Delta_1, \Gamma_1, G_1 \rangle < \langle \Delta_2, \Gamma_2, G_2 \rangle$ if

- $ord(\langle \Delta_1, \Gamma_1, G_1 \rangle) < ord(\langle \Delta_2, \Gamma_2, G_2 \rangle)$, or
- $ord(\langle \Delta_1, \Gamma_1, G_1 \rangle) = ord(\langle \Delta_2, \Gamma_2, G_2 \rangle)$ and G_1 is a strict subformula of a goal G_2', where G_2' is obtained by renaming the free variables in G_2.

Such partial order is well-founded, because $(\mathbb{N}, <)$ is also well-founded and formulas are finite sequences of symbols.

Theorem 2. *For any Δ, Γ and G goal, $lfp(T), \Delta, \Gamma \models G \iff \Delta; \Gamma \vdash_{\mathcal{UC}} G$.*

Proof. The whole proof can be found in the Appendix. Only the cases when G is atomic or an existential quantification are considered below.

$\Leftarrow$) Let h be the height of a $\mathcal{UC}$-proof for $\Delta; \Gamma \vdash_{\mathcal{UC}} G$. The claim is proved inductively on h. For the base case $h = 1$ and $G \equiv C$ (see the Appendix). For the inductive case, we suppose that $\Delta; \Gamma \vdash G$ has a proof of height h. Let us prove that $lfp(T), \Delta, \Gamma \models G$ by case analysis on the $\mathcal{UC}$-rule employed in the bottom of such proof.

- $(Clause)$. It must be the case that there exist a variant $\forall \overline{x}(G \Rightarrow A')$ of a clause of $elab(\Delta)$, such that the variables in $\overline{x}$ do not occur free in Δ, Γ, A, and that the sequent $\Delta; \Gamma \vdash \exists \overline{x}(A \approx A' \wedge G)$ has a proof of height $h - 1$. By induction hypothesis, $lfp(T), \Delta, \Gamma \models \exists \overline{x}(A \approx A' \wedge G)$. Using the definition of the operator T, the latter implies $A \in T(lfp(T))(\Delta, \Gamma)$, which is equivalent to $T(lfp(T)), \Delta, \Gamma \models A$. But since $T(lfp(T)) = lfp(T)$, the proof is complete.
- $(\exists_R)$. G must be of the form $\exists x G'$, and there must be a constraint C and a variable y not occurring free in Δ, Γ, $\exists x G'$, such that $\Delta; \Gamma, C \vdash G'[y/x]$ has a proof of height $h-1$ and $\Gamma \vdash_{\mathcal{C}} \exists y C$. By induction hypothesis, $lfp(T), \Delta, \Gamma \cup \{C\} \models G'[y/x]$, and therefore $lfp(T), \Delta, \Gamma \models \exists x G'$.

$\Rightarrow$) By induction on the order $(\mathcal{S}, <)$. Let us take $\langle \Delta, \Gamma, G \rangle \in \mathcal{S}$ and assume that, for any other $\langle \Delta', \Gamma', G' \rangle \in \mathcal{S}$, $\langle \Delta', \Gamma', G' \rangle < \langle \Delta, \Gamma, G \rangle$ implies $\Delta'; \Gamma' \vdash_{\mathcal{UC}} G'$. Then, let us conclude $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ by case analysis on the structure of G.

- $G \equiv A$. $\langle \Delta, \Gamma, A \rangle \in \mathcal{S}$ implies that $lfp(T), \Delta, \Gamma \models A$.
  Let $k = ord(\langle \Delta, \Gamma, A \rangle)$, so $T^k(I_\perp), \Delta, \Gamma \models A$, which is equivalent to $A \in (T^k(I_\perp))(\Delta, \Gamma)$. This implies that there is a variant $\forall \overline{x}(G \Rightarrow A')$ of a clause of $elab(\Delta)$ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $T^{k-1}(I_\perp), \Delta, \Gamma \models \exists \overline{x}(A \approx A' \wedge G)$. So $\langle \Delta, \Gamma, \exists \overline{x}(A \approx A' \wedge G) \rangle < \langle \Delta, \Gamma, A \rangle$, and the induction hypothesis can be applied, obtaining that $\Delta; \Gamma \vdash_{\mathcal{UC}} \exists \overline{x}(A \approx A' \wedge G)$. Using the rule $(Clause)$ with the elaborated clause $\forall \overline{x}(G \Rightarrow A')$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} A$.
- $G \equiv \exists x G'$. Then $\langle \Delta, \Gamma, G \rangle \in \mathcal{S}$ implies that there is a constraint C and a variable y such that y does not occur free in Δ, Γ, $\exists x G'$, $\Gamma \vdash_{\mathcal{C}} \exists y C'$ and $lfp(T), \Delta, \Gamma \cup \{C\} \models G'[y/x]$. Clearly, $ord(\langle \Delta, \Gamma, G \rangle) = ord(\langle \Delta, \Gamma \cup \{C\}, G'[y/x] \rangle)$ and $G'[y/x]$ is a renaming of a strict subformula of G, so $\langle \Delta, \Gamma \cup \{C\}, G'[y/x] \rangle < \langle \Delta, \Gamma, G \rangle$. Therefore, by the induction hypothesis we obtain $\Delta; \Gamma, C \vdash_{\mathcal{UC}} G'[y/x]$. Thanks to the rule $(\exists_R)$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$. □

This fixed point semantics supplies a framework in which properties of programs can be easily analyzed. For instance, two programs can be compared using the interpretation $lfp(T)$. Let us consider that two programs Δ and Δ' are said to be equivalent if, for any Γ and G, $\Delta; \Gamma \vdash_{\mathcal{UC}} G \iff \Delta'; \Gamma \vdash_{\mathcal{UC}} G$. In other words, for every Γ, the same goals can be deduced from them. Then the problem of check the equivalence between Δ and Δ' can be reduced to prove that $lfp(T)(\Delta, \Gamma) = lfp(T)(\Delta', \Gamma)$, for every Γ. This is due to the

previous results, since intuitively $lfp(T)$ provides the atoms that can be proved from a program in the context of a set of constraints.

Example 4. Let $\Delta = \{\forall x(x \geq 0 \Rightarrow p(x)), \forall x(x < 0 \Rightarrow p(x))\}$, and $\Delta' = \{\forall x(x \geq 0 \vee x < 0 \Rightarrow p(x))\}$, two programs for the instance $HH(\mathcal{R})$. Δ and Δ' are not equivalent because $lfp(T)(\Delta, \emptyset) = \emptyset$, but $p(y) \in lfp(T)(\Delta', \emptyset)$. This happens because the entailment relation in the constraint system $\mathcal{R}$ is classical deduction, but, for programs, an intuitionistic interpretation approach is considered.

On the contrary, if Δ and Δ' are such that $\{\forall x(q(x) \Rightarrow p(x)), \forall x(q'(x) \Rightarrow p(x))\} \subseteq \Delta$, and $\{\forall x(q(x) \vee q'(x) \Rightarrow p(x))\} \subseteq \Delta'$, these programs could be equivalent.

4.3 Models

At this stage, $lfp(T)$ has already been proved to be a sound and complete semantics with respect to $\mathcal{UC}$ in a sense. However, instead of having a unique model, it would also be desirable to provide for a more general notion of model such that $\Delta, \Gamma \vdash_{\mathcal{UC}} G$ iff G is true in the context $\langle \Delta, \Gamma \rangle$ for every model. Such notion of model is provided below, together with the expected results.

Definition 6. Given $D \equiv \forall \overline{x}(G \Rightarrow A)$, an interpretation I is a *model* of D, denoted by $I \models D$, if for any Δ, Γ and $A' \in At$ such that D is a variant of a clause in $elab(\Delta)$ and no variable $x \in \overline{x}$ occurs free in Δ, Γ, A', if $I, \Delta, \Gamma \models \exists \overline{x}(G \wedge A \approx A')$ then $A' \in I(\Delta, \Gamma)$.

Intuitively, an interpretation I is model of an elaborated clause D if, whenever D is available, I gathers all the atoms possibly inferred by using the clause D.

Definition 7. An interpretation I is said to be a *model* if $I \models D$ holds for every elaborated clause D.

Lemma 6. *For any interpretation I, $I \in \mathcal{I}$ is a model $\iff$ $T(I) \sqsubseteq I$.*

Proof. $T(I) \sqsubseteq I \iff$ for any Δ and Γ, $T(I)(\Delta, \Gamma) \subseteq I(\Delta, \Gamma) \iff$ for any Δ, Γ, A and any variant $\forall \overline{x}(G \Rightarrow A')$ of a clause in $elab(\Delta)$ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, if $I, \Delta, \Gamma \models \exists \overline{x}(A \approx A' \wedge G)$ then $A \in I(\Delta, \Gamma)$. However, by Definition 6, this is equivalent to say that $I \models D$ for any elaborated clause D, i. e., I is a model. □

Lemma 7. *For any $I \in \mathcal{I}$, if $T(I) \sqsubseteq I$ then $lfp(T) \sqsubseteq I$.*

Proof. It is well known that, for continuous operators in complete lattices, any postfixed point is greater (or equal to) the least fixed point. □

Theorem 3. *For any Γ, Δ and G,*

$$\Delta; \Gamma \vdash_{\mathcal{UC}} G \iff I, \Delta, \Gamma \models G \text{ holds for every model } I.$$

Proof. $I, \Delta, \Gamma \models G$ for every model $I \iff I, \Delta, \Gamma \models G$ for every I such that $T(I) \sqsubseteq I$, thanks to Lemma 6 $\iff$ $lfp(T), \Delta, \Gamma \models G$, from Lemmas 2 and 7 $\iff$ $\Delta; \Gamma \vdash_{\mathcal{UC}} G$, by virtue of Theorem 2. □

4.4 Considering particular classes of constraint systems

A fixed point semantics has just been presented for $HH(\mathcal{C})$ for any general constraint system $\mathcal{C}$. In it, the constraint system has been used as a black box, through the entailment relation $\vdash_{\mathcal{C}}$, which is a syntactic tool. See, for example, the cases C and $\exists x G$ of Definition 4. The conditions imposed in Section 2 are meant as minimal requirements for a $\mathcal{C}$ to be a constraint system, but in many useful cases $\mathcal{C}$ satisfies additional properties. For example, it is many times the case when $\mathcal{L}_{\mathcal{C}}$ is the whole set of first-order formulas over a signature, and $\Gamma \vdash_{\mathcal{C}} C$ holds iff $Ax_{\mathcal{C}} \cup \Gamma \vdash C$, where $Ax_{\mathcal{C}}$ is a suitable set of first-order axioms and $\vdash$ is the entailment relation of classical first-order logic with equality. There are many well known constraint systems of this form. For instance, $\mathcal{CFT}$, where $Ax_{\mathcal{CFT}}$ is Smolka and Treinen's axiomatization of the domain of *feature trees* [15]; or $\mathcal{R}$, where $Ax_{\mathcal{R}}$ is Tarski's axiomatization of the real numbers [16]. See also the system $\mathcal{RH}$ defined in [6]. In these cases, the syntactic relation $\vdash_{\mathcal{C}}$ has a clear connection with the inference relation $\models$ in classical logic, and the requirements specified by $\vdash_{\mathcal{C}}$ in the preceding semantics can be replaced by conditions over $\models$.

As in the frame of *CLP* we are interested in generalized conditions for the constraint systems that would guarantee the existence of semantics for constraints, based on a model theory, and that could be incorporated to the fixed point semantics of logic programs.

Usually the semantics of constraint logic programs are based on the assumption that the domain of computation (model), which is the structure used to interpret the constraints; the solver, which checks if a constraint is satisfiable; and the constraint theory, that describes the logical semantics of the constraints, *agree*. See [8] for details.

Now we will focus on constraint systems for which an additional condition is required. This condition represents the same idea of agreement mentioned before, and it is specified now. Some notation is introduced for that purpose.

Given a constraint system $\mathcal{C}$ over a signature Σ, the $\mathcal{C}$-constraints will be interpreted by means of a Σ-structure $\mathcal{A}_{\mathcal{C}}$ consisting of a carrier set and an interpretation over it for every symbol of Σ. An assignment for $\mathcal{A}_{\mathcal{C}}$, denoted ν, is a function mapping variables into elements of the carrier of $\mathcal{A}_{\mathcal{C}}$. $[\![\_]\!]_{\nu}^{\mathcal{A}_{\mathcal{C}}}$ is a boolean function that applied to a constraint C produces the classical interpretation of C under $\mathcal{A}_{\mathcal{C}}$ and ν.

$\mathcal{A}_{\mathcal{C}}$ and $\mathcal{C}$ are said to *agree* if for any Γ, C and ν, $\Gamma \vdash_{\mathcal{C}} C$ if and only if $[\![\bigwedge \Gamma \Rightarrow C]\!]_{\nu}^{\mathcal{A}_{\mathcal{C}}} = \mathit{true}$.

We have defined a semantics for this class of constraint systems, based on a notion of forcing similar to that in Definition 4, but in which the sets of constraints are replaced by the assignments making them true in $\mathcal{A}_{\mathcal{C}}$. The introduction of the whole theory leads to several intermediate definitions and technical lemmas that are not developed here. However, the main ideas and results are summarized.

Let us assume that $\mathcal{A}_{\mathcal{C}}$ and $\mathcal{C}$ agree, and that ν henceforth denotes assignments for $\mathcal{A}_{\mathcal{C}}$. Constraints will be interpreted by the sets of such assignments that make them true. Formally, given a constraint C, the set $[\![C]\!]$ is defined as

$\{\nu : dom(\nu) = free(C) \mid [\![C]\!]_\nu = true\}$[3], where $dom(\nu)$ is the set of variables mapped by ν. Such definition is directly extended to finite sets of constraints, i. e. $[\![\Gamma]\!] = [\![\bigwedge \Gamma]\!]$. Notice that $\Gamma \vdash_{\mathcal{C}} C$ iff $[\![\Gamma]\!] \subseteq [\![C]\!]$.

Example 5. Let $\mathcal{C} = \mathcal{R}$ and $\mathcal{A}_{\mathcal{R}}$ be the Σ-structure whose carrier is $\mathbb{R}$ and that interprets constants for real numbers and arithmetic symbols in the natural way. If $C \equiv x * x + y * y = 1$, then $[\![C]\!] = \{\nu : \{x, y\} \to \mathbb{R}^2 \mid (\nu(x))^2 + (\nu(y))^2 = 1\}$. Once each variable is associated to a coordinate axis, this can be assimilated to the set $\{\langle x, y\rangle \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}$, the circle of radius 1 centered at the origin of the real plane. Thus, the syntactic object $x * x + y * y = 1$ is replaced by the circle which is its intended meaning in $\mathcal{A}_{\mathcal{R}}$.

As in the case of $\models$, we are looking for a model $\overline{I}$ and a relation $\models^{\mathcal{C}}$ such that for any Δ, Γ and G, $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ iff $\overline{I}, \Delta, [\![\Gamma]\!] \models^{\mathcal{C}} G$. Several technicalities are required in the proof of such result, but the foundations are in some sense similar to those for the preceding semantics. A new notion of interpretation $\overline{I}$ is defined. Interpretations are monotonous functions applied to pairs $\langle \Delta, \nu\rangle$. An operator $\overline{T}$ that transforms such interpretations is defined, whose least fixed point is the model we were looking for. The main result is the following theorem.

Theorem 4. *For any Δ, Γ and G, $lfp(T), \Delta, \Gamma \models G \Longleftrightarrow lfp(\overline{T}), \Delta, [\![\Gamma]\!] \models^{\mathcal{C}} G$. Therefore, $\Delta; \Gamma \vdash_{\mathcal{UC}} G \iff lfp(\overline{T}), \Delta, [\![\Gamma]\!] \models^{\mathcal{C}} G$.*

5 Conclusions

In previous papers [10,9] combinations of *HH* and *CLP* were proposed, producing first and higher order schemes *HH*($\mathcal{C}$) parametric w.r.t. the constraint system. These amalgamated languages gather the expressivity and the efficiency advantages of *HH* and *CLP*, respectively. A proof system that merges inference rules from intuitionistic sequent calculus with the entailment relation of a constraint system was defined. This proof system guarantees uniform proofs, which are the basis of abstract logic programming languages [13]. A goal solving procedure that is sound and complete w.r.t. the proof system was also presented. Such procedure could be seen as an operational semantics of *HH*($\mathcal{C}$), however the absence of a more declarative semantics for this new language was evident. In this paper we have defined semantics for *HH*($\mathcal{C}$) based on fixed point constructions as is usually done in the *LP* and *CLP* fields [11,2,3,8,5].

As far as we know, our work is the first published attempt to give declarative semantics to an amalgamated logic that combines the Hereditary Harrop fragment of intuitionistic first-order logic with a constraint system. Due to the embedding of implications and universal quantifiers inside goals (and so inside programs), finding a fixed point semantics becomes a hard task, further obstructed by the presence of constraints.

In [12] a model theory is presented for an extension of Horn clauses including implications in goals based on a fixed point construction, and it is proved

[3] *free*(O) is the set of free variables in O, where O stands for a formula or set of formulas.

that the operational meaning of implication is sound and complete w.r.t. this semantics. Our approach is close to this framework, but it incorporates the semantics of universal quantifiers in goals and solves the new difficulties due to the presence of constraints.

A semantics for the fragment of λ-prolog —that is based on the higher-order logic *HH* without constraints—, in which classical and intuitionistic theories coincide, is presented in [18]. But this is not the case if implications and universal quantifiers are considered.

Referring to *CLP*, most of the defined semantics use different fixed point constructions. For instance in [8] fixed point semantics constitute a bridge between operational and algebraic semantics. This is also our aim. But notice that in traditional *CLP* the programs are limited to be Horn clauses with constraints. So in the frame of constraint systems which are complete w.r.t. a theory, programs (with embedded constraints) may be interpreted using classical logical inference. However, this is not the case in our language. A classical theory can be considered for the constraint system, but anyway the intuitionism remains, even in the interpretation of pure programs.

We are now researching for more abstract model theories based on indexed categories or uniform algebras [4,1], that could provide a pure model-theoretic semantics, not so directly connected with the operational semantics. Models should provide for meanings of constraints, programs and goals in a homogeneous way, and the expected general result would claim that C is a correct answer constraint for G from Δ, if and only if, every model satisfying Δ and C satisfies G.

Acknowledgements We appreciate the comments and suggestions from James Lipton concerning the presented work.

References

1. G. Amato and J. Lipton. Indexed categories and bottom-up semantics of logic programs. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR'01*, LNCS 2250, pages 438–454. Springer, 2001.
2. A. Bossi, M. Gabrielli, G. Levi, and M. C. Meo. A compositional semantics for logic programs. *Theoretical Computer Science*, 122(1-2):3–47, 1994.
3. M. Comini, G. Levi, and M. C. Meo. A theory of observables for logic programs. *Information and Computation*, 69(1–2):23–80, 2001.
4. S. E. Finkelstein, P. Freyd, and J. Lipton. A new framework for declarative programming. *Theoretical Computer Science*, 300(1–3):91–160, 2003.
5. M. Gabbrielli, G. M. Dore, and G. Levi. Observable semantics for constraint logic programs. *Journal of Logic and Computation*, 5(2):133–171, 1995.
6. M. García-Díaz and S. Nieva. Solving mixed quantified constraints over a domain based on real numbers and Herbrand terms. In Z. Hu and M. Rodríguez-Artalejo, editors, *FLOPS'02*, LNCS 2441, pages 103–118. Springer, 2002.
7. J. Jaffar and M. J. Maher. Constraint logic programming: a survey. *Journal of Logic Programming*, 19/20:503–581, 1994.
8. J. Jaffar, M. J. Maher, K. Marriott, and P. J. Stuckey. The semantics of constraint logic programs. *Journal of Logic Programming*, 37(1-3):1–46, 1998.
9. J. Leach and S. Nieva. A higher-order logic programming language with constraints. In H. Kuchen and K. Ueda, editors, *FLOPS'01*, LNCS 2024, pages 108–122. Springer, 2001.

10. J. Leach, S. Nieva, and M. Rodríguez-Artalejo. Constraint logic programming with hereditary Harrop formulas. *Theory and Practice of Logic Programming*, 1(4):409–445, 2001.
11. J. W. Lloyd. *Foundations of logic programming*. Springer-Verlag, 1987.
12. D. Miller. A logical analysis of modules in logic programming. *Journal of Logic Programming*, 6(1-2):79–108, 1989.
13. D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.
14. G. Nadathur. A proof procedure for the logic of hereditary Harrop formulas. *Journal of Automated Reasoning*, 11:111–145, 1993.
15. G. Smolka and R. Treinen. Records for logic programming. *Journal of Logic Programming*, 18(3):229–258, 1994.
16. A. Tarski. *A decision method for elementary algebra and geometry*. University of California Press, 1951.
17. M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.
18. D. A. Wolfram. A semantics for λ-prolog. *Theoretical Computer Science*, 136:277–288, 1994.

Appendix

Lemma 2. *If $I_1, I_2 \in \mathcal{I}$ and $I_1 \sqsubseteq I_2$, then for any goal G, Δ, and Γ, $I_1, \Delta, \Gamma \models G$ implies $I_2, \Delta, \Gamma \models G$.*

Proof. The proof is inductive on the structure of G:

- $G \equiv C \in \mathcal{L}_\mathcal{C}$.
  $I_1, \Delta, \Gamma \models C \iff \Gamma \vdash_\mathcal{C} C \iff I_2, \Delta, \Gamma \models C$.
- $G \equiv A$, atomic goal.
  $I_1, \Delta, \Gamma \models A \iff A \in I_1(\Delta, \Gamma)$. $I_1 \sqsubseteq I_2$ implies that $I_1(\Delta, \Gamma) \subseteq I_2(\Delta, \Gamma)$, so $A \in I_2(\Delta, \Gamma)$ and therefore $I_2, \Delta, \Gamma \models A$.
- $G \equiv G_1 \wedge G_2$.
  $I_1, \Delta, \Gamma \models G_1 \wedge G_2 \iff I_1, \Delta, \Gamma \models G_i$ for each $i \in \{1, 2\}$. In both cases the induction hypothesis can be used, so $I_2, \Delta, \Gamma \models G_i$ for each $i \in \{1, 2\}$, which implies that $I_2, \Delta, \Gamma \models G_1 \wedge G_2$.
- $G \equiv G_1 \vee G_2$.
  $I_1, \Delta, \Gamma \models G_1 \vee G_2 \iff$ there is $i \in \{1, 2\}$ such that $I_1, \Delta, \Gamma \models G_i$. By induction hypothesis, $I_2, \Delta, \Gamma \models G_i$, which implies that $I_2, \Delta, \Gamma \models G_1 \vee G_2$.
- $G \equiv D \Rightarrow G'$.
  $I_1, \Delta, \Gamma \models D \Rightarrow G' \iff I_1, \Delta \cup \{D\}, \Gamma \models G'$. By induction hypothesis, $I_2, \Delta \cup \{D\}, \Gamma \models G'$ holds, which implies that $I_2, \Delta, \Gamma \models D \Rightarrow G'$.
- $G \equiv C \Rightarrow G'$.
  $I_1, \Delta, \Gamma \models C \Rightarrow G' \iff I_1, \Delta, \Gamma \cup \{C\} \models G'$. By induction hypothesis, $I_2, \Delta, \Gamma \cup \{C\} \models G'$ holds, which implies that $I_2, \Delta, \Gamma \models C \Rightarrow G'$.
- $G \equiv \exists x G'$.
  $I_1, \Delta, \Gamma \models \exists x G' \iff$ there is a $\mathcal{C}$-constraint C and a variable y such that:
  - y does not occur free in Δ, Γ, $\exists x G'$.
  - $\Gamma \vdash_\mathcal{C} \exists y C$.
  - $I_1, \Delta, \Gamma \cup \{C\} \models G'[y/x]$.

  By induction hypothesis, for the same C and y it holds that $I_2, \Delta, \Gamma \cup \{C\} \models G'[y/x]$, therefore $I_2, \Delta, \Gamma \models \exists x G'$.
- $G \equiv \forall x G'$.
  $I_1, \Delta, \Gamma \models \forall x G' \iff$ there is a variable y such that: y does not occur free in Δ, Γ, $\forall x G'$ and $I_1, \Delta, \Gamma \models G'[y/x]$. By induction hypothesis, it holds that $I_2, \Delta, \Gamma \models G'[y/x]$, therefore $I_2, \Delta, \Gamma \models \forall x G'$. □

Lemma 3. *Let $\{I_i\}_{i \geq 0}$ be a denumerable family of interpretations such that $I_0 \sqsubseteq I_1 \sqsubseteq I_2 \sqsubseteq \ldots$, and let G be a goal. Then, for any Δ and Γ, $\bigsqcup_{i \geq 0} I_i, \Delta, \Gamma \models G$ implies that there exists $k \geq 0$ such that $I_k, \Delta, \Gamma \models G$.*

Proof. We already know that $(\bigsqcup_{i \geq 0} I_i)(\Delta, \Gamma) = \bigcup_{i \geq 0} I_i(\Delta, \Gamma)$. By induction on the structure of G:

- $G \equiv C \in \mathcal{L}_\mathcal{C}$.
  $\bigsqcup_{i \geq 0} I_i, \Delta, \Gamma \models C \iff \Gamma \vdash_\mathcal{C} C \iff I_k, \Delta, \Gamma \models C$ is true independently of $k \geq 0$.

- $G \equiv A$, atomic formula.
  $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash A \iff A \in (\bigsqcup_{i\geq 0} I_i)(\Delta, \Gamma) = \bigcup_{i\geq 0} I_i(\Delta, \Gamma)$. Therefore, there exists $k \geq 0$ such that $A \in I_k(\Delta, \Gamma)$, hence, for that k, $I_k, \Delta, \Gamma \Vdash A$.
- $G \equiv G_1 \wedge G_2$.
  $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash G_1 \wedge G_2 \iff \bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash G_j$ for each $j \in \{1, 2\}$. In both cases the induction hypothesis can be used, so there exist $k_1, k_2 \geq 0$ such that $I_{k_j}, \Delta, \Gamma \Vdash G_j$ for each $j \in \{1, 2\}$. Let $k = max(k_1, k_2)$. Then $I_k, \Delta, \Gamma \Vdash G_j$ for each $j \in \{1, 2\}$ in virtue of Lemma 2, and therefore $I_k, \Delta, \Gamma \Vdash G_1 \wedge G_2$.
- $G \equiv G_1 \vee G_2$.
  $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash G_1 \vee G_2 \iff$ there is $j \in \{1, 2\}$ such that $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash G_j$. The induction hypothesis can be used, so there exist $k \geq 0$ such that $I_k, \Delta, \Gamma \Vdash G_j$, and therefore $I_k, \Delta, \Gamma \Vdash G_1 \vee G_2$.
- $G \equiv D \Rightarrow G'$.
  $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash D \Rightarrow G' \iff \bigsqcup_{i\geq 0} I_i, \Delta \cup \{D\}, \Gamma \Vdash G'$. By induction hypothesis, there is $k \geq 0$ such that $I_k, \Delta \cup \{D\}, \Gamma \Vdash G'$. Therefore, $I_k, \Delta, \Gamma \Vdash D \Rightarrow G'$.
- $G \equiv C \Rightarrow G'$.
  $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash C \Rightarrow G' \iff \bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \cup \{C\} \Vdash G'$. By induction hypothesis, there is $k \geq 0$ such that $I_k, \Delta, \Gamma \cup \{C\} \Vdash G'$. Therefore, $I_k, \Delta, \Gamma \Vdash C \Rightarrow G'$.
- $G \equiv \exists x G'$.
  $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash \exists x G' \iff$ there is a $\mathcal{C}$-constraint C and a variable y such that:
  - y does not occur free in Δ, Γ, $\exists x G'$.
  - $\Gamma \vdash_{\mathcal{C}} \exists y C$.
  - $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \cup \{C\} \Vdash G'[y/x]$.

  By induction hypothesis, it holds that there is a $k \geq 0$ such that $I_k, \Delta, \Gamma \cup \{C\} \Vdash G'[y/x]$. Therefore $I_k, \Delta, \Gamma \Vdash \exists x G'$.
- $G \equiv \forall x G'$.
  $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash \forall x G' \iff$ there is a variable y such that:
  - y does not occur free in Δ, Γ, $\forall x G'$.
  - $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash G'[y/x]$.

  By induction hypothesis, it happens that there exists $k \geq 0$ such that $I_k, \Delta, \Gamma \Vdash G'[y/x]$. Therefore $I_k, \Delta, \Gamma \Vdash \forall x G'$. □

Lemma 4 (Monotonicity of T). *Let $I_1, I_2 \in \mathcal{I}$ such that $I_1 \sqsubseteq I_2$. Then, $T(I_1) \sqsubseteq T(I_2)$.*

Proof. Let us consider any Δ, Γ and $A \in T(I_1)(\Delta, \Gamma)$. The latter implies that there is a variant $\forall \overline{x}(G \Rightarrow A')$ of a clause of $elab(\Delta)$, such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $I_1, \Delta, \Gamma \Vdash \exists \overline{x}(A \approx A' \wedge G)$. Using Lemma 2 and the fact that $I_1 \sqsubseteq I_2$, we obtain $I_2, \Delta, \Gamma \Vdash \exists \overline{x}(A \approx A' \wedge G)$, which implies $A \in T(I_2)(\Delta, \Gamma)$. Since no particular choice was made for A, Δ, Γ, this argument proves $T(I_1)(\Delta, \Gamma) \subseteq T(I_2)(\Delta, \Gamma)$ for any Δ and Γ, therefore $T(I_1) \sqsubseteq T(I_2)$. □

Theorem 2. *For any Δ, Γ and G, $lfp(T), \Delta, \Gamma \Vdash G \iff \Delta; \Gamma \vdash_{\mathcal{UC}} G$.*

Proof. Both implications are proved by induction.

$\Leftarrow$) Let h be the height of a $\mathcal{UC}$-proof for $\Delta; \Gamma \vdash_{\mathcal{UC}} G$. The claim is proved inductively on h.

- Base case: $h = 1$. The only possibility is that $G \equiv C \in \mathcal{L}_{\mathcal{C}}$. Then $\Delta; \Gamma \vdash_{\mathcal{UC}} C$ implies that $\Gamma \vdash_{\mathcal{C}} C$, and therefore $lfp(T), \Delta, \Gamma \Vdash C$ holds.
- Inductive case. Assuming that $\Delta; \Gamma \vdash G$ has a proof of height h, let us prove that $lfp(T), \Delta, \Gamma \Vdash G$ by case analysis on the $\mathcal{UC}$-rule employed in the bottom of such proof.

($Clause$) So there is a variant $\forall \overline{x}(G \Rightarrow A')$ of a clause of $elab(\Delta)$ such that the variables in $\overline{x}$ do not occur free in Δ, Γ, A, and that the sequent $\Delta; \Gamma \vdash \exists \overline{x}(A \approx A' \wedge G)$ has a proof of height $h-1$. By induction hypothesis, $lfp(T), \Delta, \Gamma \Vdash \exists \overline{x}(A \approx A' \wedge G)$. Using the definition of the operator T, the latter implies that $A \in (T(lfp(T)))(\Delta, \Gamma)$, which is equivalent to $T(lfp(T)), \Delta, \Gamma \Vdash A$. But since $T(lfp(T)) = lfp(T)$, the proof is complete.

($\wedge_R$) It must be the case that there are G_1, G_2 such that $G \equiv G_1 \wedge G_2$ and the sequents $\Delta; \Gamma \vdash G_i$ have a proof of height less than h for each $i \in \{1, 2\}$. By induction hypothesis, $lfp(T), \Delta, \Gamma \Vdash G_i$ holds for $i \in \{1, 2\}$, which implies $lfp(T), \Delta, \Gamma \Vdash G$.

($\vee_R$) Then there are G_1, G_2 such that $G \equiv G_1 \vee G_2$ and the sequent $\Delta; \Gamma \vdash G_i$ has a proof of height $h-1$ for some $i \in \{1, 2\}$. By induction hypothesis, $lfp(T), \Delta, \Gamma \Vdash G_i$, hence $lfp(T), \Delta, \Gamma \Vdash G$.

($\Rightarrow_R$) In this case, $G \equiv D \Rightarrow G'$ for some D and G', and the sequent $\Delta, D; \Gamma \vdash G'$ has a proof of height $h-1$. By induction hypothesis, $lfp(T), \Delta \cup \{D\}, \Gamma \Vdash G'$. Therefore $lfp(T), \Delta, \Gamma \Vdash D \Rightarrow G'$.

($\Rightarrow_{C_R}$) It must be the case that there are C and G' such that $G \equiv C \Rightarrow G'$ and the sequent $\Delta; \Gamma, C \vdash G'$ has a proof of height $h-1$. By induction hypothesis, $lfp(T), \Delta, \Gamma \cup \{C\} \Vdash G'$. So, $lfp(T), \Delta, \Gamma \Vdash C \Rightarrow G'$.

($\exists_R$) G must be of the form $\exists x G'$, and there must exist C and a variable y not occurring free in Δ, Γ, $\exists x G'$, such that $\Delta; \Gamma, C \vdash G'[y/x]$ has a proof of height $h-1$ and $\Gamma \vdash_{\mathcal{C}} \exists y C$. By induction hypothesis, $lfp(T), \Delta, \Gamma \cup \{C\} \Vdash G'[y/x]$, and therefore $lfp(T), \Delta, \Gamma \Vdash \exists x G'$.

($\forall_R$) G must be of the form $\forall x G'$, and there must exist a variable y not occurring free in Δ, Γ, $\forall x G'$ such that $\Delta; \Gamma \vdash G'[y/x]$ has a proof of height $h-1$. By induction hypothesis, $lfp(T), \Delta, \Gamma \Vdash G'[y/x]$, and therefore $lfp(T), \Delta, \Gamma \Vdash \forall x G'$.

$\Rightarrow$) By induction on the order $(\mathcal{S}, <)$. Let us take $\langle \Delta, \Gamma, G \rangle \in \mathcal{S}$ and assume that, for any other $\langle \Delta', \Gamma', G' \rangle \in \mathcal{S}$, $\langle \Delta', \Gamma', G' \rangle < \langle \Delta, \Gamma, G \rangle$ implies $\Delta'; \Gamma' \vdash_{\mathcal{UC}} G'$. Then, let us conclude $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ by case analysis on the structure of G.

- $G \equiv C \in \mathcal{L}_{\mathcal{C}}$. Then $\langle \Delta, \Gamma, C \rangle \in \mathcal{S}$ implies that $\Gamma \vdash_{\mathcal{C}} C$, and therefore $\Delta; \Gamma \vdash_{\mathcal{UC}} C$.
- $G \equiv A$. In this case $\langle \Delta, \Gamma, A \rangle \in \mathcal{S}$ implies that $lfp(T), \Delta, \Gamma \Vdash A$. Let $k = ord(\langle \Delta, \Gamma, A \rangle)$, so $T^k(I_\perp), \Delta, \Gamma \Vdash A$, which is equivalent to $A \in T^k(I_\perp)(\Delta, \Gamma)$. This implies that there is $\forall \overline{x}(G \Rightarrow A') \in elab(\Delta)$ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and in addition

$T^{k-1}(I_\perp), \Delta, \Gamma \models \exists \overline{x}(A \approx A' \wedge G)$. Due to the way in which the order in $\mathcal{S}$ is defined, $\langle \Delta, \Gamma, \exists \overline{x}(A \approx A' \wedge G)\rangle < \langle \Delta, \Gamma, A\rangle$, so the induction hypothesis can be applied, obtaining that $\Delta; \Gamma \vdash_{\mathcal{UC}} \exists \overline{x}(A \approx A' \wedge G)$. Using the rule $(Clause)$ with the clause $\forall \overline{x}(G \Rightarrow A')$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} A$.

- $G \equiv G_1 \wedge G_2$. Then $\langle \Delta, \Gamma, G\rangle \in \mathcal{S}$ implies $lfp(T), \Delta, \Gamma \models G_1$ and $lfp(T), \Delta, \Gamma \models G_2$. It is obvious that $ord(\langle \Delta, \Gamma, G\rangle) = ord(\langle \Delta, \Gamma, G_1\rangle = ord(\langle \Delta, \Gamma, G_2\rangle)$ and G_1, G_2 are strict subformulas of G, and hence $\langle \Delta, \Gamma, G_1\rangle, \langle \Delta, \Gamma, G_2\rangle < \langle \Delta, \Gamma, G\rangle$. Therefore, by the induction hypothesis we obtain $\Delta; \Gamma \vdash_{\mathcal{UC}} G_1$ and $\Delta; \Gamma \vdash_{\mathcal{UC}} G_2$. Thanks to the rule $(\wedge_R)$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$.
- $G \equiv G_1 \vee G_2$. Then $\langle \Delta, \Gamma, G\rangle \in \mathcal{S}$ implies that there is $i \in \{1, 2\}$ such that $lfp(T), \Delta, \Gamma \models G_i$. Clearly, $ord(\langle \Delta, \Gamma, G\rangle) = ord(\langle \Delta, \Gamma, G_i\rangle)$ and G_i is a strict subformula of G, so $\langle \Delta, \Gamma, G_i\rangle < \langle \Delta, \Gamma, G\rangle$. Therefore, by the induction hypothesis we obtain $\Delta; \Gamma \vdash_{\mathcal{UC}} G_i$ for some $i \in \{1, 2\}$. Thanks to the rule $(\vee_{R_i})$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$.
- $G \equiv D \Rightarrow G'$. Then $\langle \Delta, \Gamma, G\rangle \in \mathcal{S}$ implies that $lfp(T), \Delta \cup \{D\}, \Gamma \models G'$. Clearly, $ord(\langle \Delta, \Gamma, G\rangle) = ord(\langle \Delta \cup \{D\}, \Gamma, G'\rangle)$ and G' is a strict subformula of G, so $\langle \Delta \cup \{D\}, \Gamma, G'\rangle < \langle \Delta, \Gamma, G\rangle$. Therefore, by the induction hypothesis we obtain $\Delta, D; \Gamma \vdash_{\mathcal{UC}} G'$. Thanks to the rule $(\Rightarrow_R)$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$.
- $G \equiv C \Rightarrow G'$. Then $\langle \Delta, \Gamma, G\rangle \in \mathcal{S}$ implies that $lfp(T), \Delta, \Gamma \cup \{C\} \models G'$. Clearly, $ord(\langle \Delta, \Gamma, G\rangle) = ord(\langle \Delta, \Gamma \cup \{C\}, G'\rangle)$ and G' is a strict subformula of G, so $\langle \Delta, \Gamma \cup \{C\}, G'\rangle < \langle \Delta, \Gamma, G\rangle$. Therefore, by the induction hypothesis we obtain $\Delta; \Gamma, C \vdash_{\mathcal{UC}} G'$. Thanks to the rule $(\Rightarrow_{C_R})$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$.
- $G \equiv \exists x G'$. Then $\langle \Delta, \Gamma, G\rangle \in \mathcal{S}$ implies that there is a constraint C and a variable y such that:
  - y does not occur free in Δ, Γ, $\exists x G'$.
  - $\Gamma \vdash_{\mathcal{C}} \exists y C$.
  - $lfp(T), \Delta, \Gamma \cup \{C'\} \models G'[y/x]$.

  Clearly, $ord(\langle \Delta, \Gamma, G\rangle) = ord(\langle \Delta, \Gamma \cup \{C'\}, G'[y/x]\rangle)$ and $G'[y/x]$ is a renaming of a strict subformula of G, so $\langle \Delta, \Gamma \cup \{C\}, G'[y/x]\rangle < \langle \Delta, \Gamma, G\rangle$. Therefore, by the induction hypothesis we obtain $\Delta; \Gamma, C \vdash_{\mathcal{UC}} G'[y/x]$. Thanks to the rule $(\exists_R)$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$.
- $G \equiv \forall x G'$. Then $\langle \Delta, \Gamma, G\rangle \in \mathcal{S}$ implies that there is a variable y such that:
  - y does not occur free in Δ, Γ, $\forall x G'$.
  - $lfp(T), \Delta, \Gamma \models G'[y/x]$.

  Clearly, $ord(\langle \Delta, \Gamma, G\rangle) = ord(\langle \Delta, \Gamma, G'[y/x]\rangle)$ and $G'[y/x][x/y] \equiv G'$ is a strict subformula of G, so $\langle \Delta, \Gamma, G'[y/x]\rangle < \langle \Delta, \Gamma, G\rangle$. Therefore, by the induction hypothesis we obtain $\Delta; \Gamma \vdash_{\mathcal{UC}} G'[y/x]$. Thanks to the rule $(\forall_R)$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$. $\square$