Formalizing Two Fixed Point Semantics for *HH*($\mathcal{C}$)

Miguel García-Díaz and Susana Nieva [⋆]

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

Abstract. The scheme *HH*($\mathcal{C}$) emerged as a double extension of traditional Logic Programming. On one hand, extending Horn logic to hereditary Harrop formulas (*HH*), in order to improve the expressive power; on the other, incorporating constraints, in order to increase the efficiency. The behavior of such extended *CLP* programs was explained by means of a sequent calculus that, from every program and set of constraints, exclusively generates uniform proofs —i. e. goal-oriented proofs— for any goal, as well as by means of a goal solving procedure. Hence *HH*($\mathcal{C}$) was provided for an operational semantics.

Recently, some attempts to define more declarative semantics for *HH*($\mathcal{C}$) has been done by the authors. Here such works are enlarged and improved. Two declarative semantics for *HH*($\mathcal{C}$) based on fixed point techniques are revisited. They are proved to be equivalent and the corresponding theorems of soundness and completeness, that relate the sequent calculus (and so the operational semantics) with these semantics, are stated and formally proved. Non trivial proofs are included in the current paper, and every technical aspect is developed.

1 Introduction

In Constraint Logic Programming (*LP*) as well as in Constraint Logic Programming (*CLP*), the operational (algorithmic) interpretation and the declarative (mathematical) meaning agree with each other, in the sense that the declarative meaning of a logic or constraint logic program can be interpreted operationally as a goal-oriented search for solutions. In [14] 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 traditional *LP*, enhancing the weak expressive power of logic programs based on Horn clauses, have been proved to be abstract logic programming languages ([14,15]). This is also the case of the language *HH*($\mathcal{C}$), on which the present paper focuses. It was introduced and proved to be an abstract logic programming language in [11].

HH($\mathcal{C}$) is a combination of the logic of Hereditary Harrop formulas (*HH*) and *CLP*, it can be considered as an 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* [8]. *HH* extends Horn logic allowing disjunction, intuitionistic implications and universal quantifiers in goals. These constructions are essential for capturing module

[⋆] The authors are partially supported by the Spanish project TIC2002-01167 'MELODIAS'.

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. For example, in [5] a constraint solver for an interesting and useful instance of our scheme with a constraint system which combines real numbers with Herbrand terms is described.

A proof system, called $\mathcal{UC}$, that exclusively generates uniform proofs was defined in [11] showing that $HH(\mathcal{C})$ is in fact an abstract logic programming language. In addition a goal solving procedure was presented as an operational semantics for the scheme $HH(\mathcal{C})$, and it was proved to be sound and complete w.r.t. the intuitionistic deduction system $\mathcal{UC}$.

Although an operational interpretation is needed in order to specify programs that can be executed with certain efficiency, a clear declarative semantics would indeed simplify the programmer's work. The use of provability as declarative interpretation is still too close to the operational behavior. The proof system, which is a syntactic tool, should be supported by model-theoretic semantics involving more abstract elements.

The attempts to provide declarative semantics for *LP* languages based on mathematical foundations are extensive and fruitful (see v.g. [12,1,2]). This is also the case of *CLP* [9,4]. In both, *LP* and *CLP*, most of the studies are based on fixed point theories. The semantics presented in this paper are inspired by the fixed point semantics for the fragment of *HH* that incorporates intuitionistic implication in goals, described in [13]. We extended it in two directions in order to interpret the whole *HH* logic: one to deal with universal quantifiers, and second to manage the presence of constraints.

A model must be found, such that for any program Δ, finite set of constraints Γ and goal G, G can be proved, from Δ and Γ, in the deduction system $\mathcal{UC}$, if and only if, G is satisfied in that model. However, in order to build such model it is important to realize that, during the search of a proof of a goal from a program Δ and a set of constraints Γ, both Δ and Γ may grow. Having this condition in mind, we have introduced a new notion of interpretation. A interpretation I will be a function that associates 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 that I associates to the augmented pair cannot decrease. The model we are looking for will be the least fixed point of a continuous operator that transforms such kind of interpretations.

The main difference between the two semantics we provide is the way in which constraint interpretation is dealt with. For the first one, the denotation of constraints is given in terms of the entailment relation of the constraint system. For the second, a class of constraint systems is considered for which the logical inference based on classical model theory can be used to interpret constraints. The fixed point semantics is reformulated incorporating the logical inference for constraints, instead of the entailment relation. That way, the sets of constraints are replaced by their denotations and the interpretations are applied to pairs $\langle \Delta, \nu \rangle$, where ν is an assignment of the variables into the constraint domain.

This work enhances [6] and [7]. In [6] only the first fixed point semantics described here is included. [7] includes both, but some technical aspects and many of the proofs are omitted or sketched there.

The rest of this paper is organized as follows: Section 2 gathers the syntactic aspects of *HH*($\mathcal{C}$), such as the syntax of the constraints, programs and goals, as well as the definition of constraint systems. In Section 3the rules of the proof system $\mathcal{UC}$ are presented. Some examples of the use of *HH*($\mathcal{C}$) as logic programming language are also shown. Section 4 is devoted to formalize the two fixed point semantics for *HH*($\mathcal{C}$). They are rigorously proved to be sound and complete w.r.t. provability in $\mathcal{UC}$. The first semantics is directly connected with the calculus. The second one is proved to be equivalent to the first. In Section 5 related works are commented.

2 The Language *HH*($\mathcal{C}$)

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 [11]. 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.

2.1 A general constraint system $\mathcal{C}$

Given a signature Σ containing constants, function symbols 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 Γ and constraints C. $\mathcal{C}$ must fulfill the following conditions:

- $\mathcal{L}_\mathcal{C}$ is 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[2].
- All the inference rules related to $\wedge, \Rightarrow, \exists, \forall$ and $\approx$ valid in the intuitionistic fragment of first-order logic are also valid in $\vdash_\mathcal{C}$.

[1] Here and in the rest of the paper, given a set S, $\mathcal{P}(S)$ denotes its power set.
[2] $\Gamma\sigma$ is the result of applying the substitution σ to each formula in Γ, avoiding the capture of variables.

The preceding conditions are minimal requirements for a $\mathcal{C}$ to be a constraint system, but in many useful cases $\mathcal{C}$ satisfies additional properties. For instance, if $Ax_{\mathcal{CFT}}$ is Smolka and Treinen's axiomatization of the domain of feature trees [16], the constraint system $\mathcal{CFT}$ can be defined considering the whole set of first-order formulas as constraints, and $\Gamma \vdash_{\mathcal{CFT}} C$ iff $\Gamma \cup Ax_{\mathcal{CFT}} \vdash C$, where $\vdash$ is the entailment relation of classical first-order logic with equality. Another example is the constraint system $\mathcal{R}$ that can be defined analogously to $\mathcal{CFT}$, but using $Ax_{\mathcal{R}}$, the Tarski's axiomatization of the closed field of real numbers [17]. See also the system $\mathcal{RH}$, that combines the field of real numbers with finite trees, defined in [5].

Hereafter, we will consider a fixed signature Σ and a constraint system $\mathcal{C}$ over Σ. Γ will stand for finite sets of constraints. A set of constraints Γ is said to be *$\mathcal{C}$-satisfiable* if $\emptyset \vdash_{\mathcal{C}} \overline{\exists}(\bigwedge \Gamma)$, where $\overline{\exists}$ denotes existential closure and $\bigwedge \Gamma$ the conjunction of constraints in Γ.

Constraints can be found embedded in goals and clauses as described in the following subsection.

2.2 Clauses and goals

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 over 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:
$G ::= A \mid C \mid G_1 \wedge G_2 \mid G_1 \vee G_2 \mid D \Rightarrow G \mid C \Rightarrow G \mid \exists x G \mid \forall x G,$
$D ::= A \mid G \Rightarrow A \mid D_1 \wedge D_2 \mid \forall x D,$
where $A \in At$, $C \in \mathcal{L}_{\mathcal{C}}$.

Definition 1. A *program*, noted Δ, over Σ and Π_P is a finite subset of $\mathcal{D}$.
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 2. Given a set of clauses S, the *elaboration* of S is the set of clauses $elab(S) \overset{\text{def}}{=} \bigcup_{D \in S} elab(D)$, where $elab(D)$ is defined by the following rules:

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

An *elaborated clause* is a clause of the form $\forall \overline{x}(G \Rightarrow A)$[3].

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

In order to simplify the notation, in this paper we will identify a program with its elaboration. And we will write Δ, to refer to $elab(\Delta)$. In this way, programs can be understood as sets of elaborated clauses.

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 $x_i \in \overline{x}$.

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 here, for the instance $HH(\mathcal{R})$.

Example 1. Taking $\Pi_P = \{$`triangle`, `isosceles`$\}$, let us consider the program Δ_1, in a prolog-like notation, enriched with constraints and the logic connectives $\forall$ and $\Rightarrow$:

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

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

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

Suppose that, from Δ_1, we want to know which conditions over a variable y guarantee that, for any $x > 1$, it is possible to build an isosceles triangle with sides $\langle x, x, y\rangle$. Δ_1 must import the clauses of Δ_2, and the goal which captures that query is:

$$G \equiv (\Delta_2 \Rightarrow \forall x(x > 1 \Rightarrow isosceles(x, x, y))),$$

where Δ_2 means here the conjunction of its clauses. Similarly as in [13], the first implication of G should mean that the right hand side must be solved loading the program Δ_2 as a *module*.

Example 2. Consider the program below, borrowed from [11].

```
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)).
```

It is a reversible program to compute Fibonacci numbers. Reversibility is also obtained in a pure *CLP* version, but with a program that runs in exponential time and that recalculates Fibonacci numbers. The $HH(\mathcal{R})$ version is more efficient since none Fibonacci number must be recalculated, and goals of the form $fib(n, x)$, n given, run in linear time.

Other examples can be found in [11,10,5]. The ones in [10] belong to the higher-order version of $HH(\mathcal{C})$, and those in [5] to the instance $HH(\mathcal{RH})$.

3 The Sequent Calculus $\mathcal{UC}$

We follow the ideas of Miller et al. [14], 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. The goal solving procedure of any of these languages and its respective deduction system agree, and in both (goal solving and deduction system) the search of a proof for a goal is directed by the structure of such goal. For the case of *HH*($\mathcal{C}$), the calculus that guarantees uniform proofs, called $\mathcal{UC}$, was defined in [11], and it is briefly described now.

$\mathcal{UC}$ is a sequent calculus which combines intuitionistic rules for the logic connectives with the entailment relation $\vdash_{\mathcal{C}}$. For any program Δ, finite set of constraints Γ, and goal G, $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ means that there is a proof for the sequent $\Delta; \Gamma \mid\!\!-\, G$ using, in a bottom-up fashion, the rules of the calculus $\mathcal{UC}$ that appear below. So $\mathcal{UC}$-proofs will be regarded as trees.

$\mathcal{UC}$-Rules

Rules for constraints and atomic goals:

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

where $\forall \overline{x}(G \Rightarrow A')$ is a variant of some clause in Δ; 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:

$$\frac{\Delta; \Gamma \mid\!\!-\, G_1 \quad \Delta; \Gamma \mid\!\!-\, G_2}{\Delta; \Gamma \mid\!\!-\, G_1 \wedge G_2}\ (\wedge_R) \qquad \frac{\Delta; \Gamma \mid\!\!-\, G_i}{\Delta; \Gamma \mid\!\!-\, G_1 \vee G_2}\ (\vee_R)$$

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

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

In rules $(\exists_R)$ and $(\forall_R)$ the variable y does not occur free in any formula of the lower sequent, and $i \in \{1, 2\}$ in rule $(\vee_R)$.

When $\Delta; C \vdash_{\mathcal{UC}} G$ holds, C is said to be a *correct answer constraint* for G from Δ. In [11] a goal solving procedure for *HH*($\mathcal{C}$) is introduced and proved to be sound and complete w.r.t the $\vdash_{\mathcal{UC}}$-deduction.

Example 3. From the program Δ_1 of example 1, solving the goal

$$G \equiv (\Delta_2 \Rightarrow \forall x(x > 1 \Rightarrow \mathit{isosceles}(x, x, y))),$$

in accordance with the rule ($\Rightarrow_R$), the clauses of Δ_2 are in fact locally added to Δ_1, as a module. Hence, according to the proof system $\mathcal{UC}$, $C \equiv 0 < y \wedge y \leq 2$ is a correct answer constraint for G from Δ_1.

Example 4. Consider the program of example 2 to compute Fibonacci numbers. For instance, both the goals $\mathit{fib}(9, x)$ or $\mathit{fib}(x, 55)$ can be solved, obtaining the constraint answers $x \approx 55$ and $n \approx 9$, respectively.

The goal $\mathit{getfib}(n, x, m)$ 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.

4 Declarative Semantics for *HH*($\mathcal{C}$)

The goal solving procedure defined in [11] 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 model semantics. The only meanings that we may associate to programs, so far, are sets of proofs.

In this section, alternative semantics based on fixed point constructions — widely utilized in *LP* and *CLP*— are introduced.

4.1 A fixed point semantics approach

For the traditional *LP* language, given a program P there is a continuous operator T_P transforming interpretations (sets of atoms) such that G can be proved from P, if and only if, G "is true" in the least fixed point of T_P [19]. As analyzed in [13], 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. New difficulties arise extending this approach to *HH*($\mathcal{C}$), 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 $\langle \Delta, \Gamma \rangle$ should depend on interpretations of $\langle \Delta', \Gamma' \rangle$ with $\Delta' \subseteq \Delta$, $\Gamma' \subseteq \Gamma$.

Interpretations and forcing relation We have extended the model theory presented in [13] in order to interpret full *HH*($\mathcal{C}$). The semantics there defined is based on a *forcing relation* between programs and goals that represents whether an interpretation makes true a goal in the context of a program. For the reasons explained before, in our language contexts must be extended to be pairs $\langle \Delta, \Gamma \rangle$, and interpretations are defined as monotonous functions able to interpret every pair $\langle \Delta, \Gamma \rangle$.

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

Definition 3. An *interpretation* I is a monotonous function $I : \mathcal{W} \times \mathcal{P}(\mathcal{L}_{\mathcal{C}}) \rightarrow \mathcal{P}(At)$, i.e. for any Δ_1, Δ_2 and Γ_1, Γ_2 such that $\Delta_1 \subseteq \Delta_2$ and $\Gamma_1 \subseteq \Gamma_2$, $I(\Delta_1, \Gamma_1) \subseteq I(\Delta_2, \Gamma_2)$ holds. Let $\mathcal{I}$ denote the set of interpretations.

A continuous operator transforming such interpretations will be defined and proved that for any Δ, Γ and G, $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ if and only if G is forced by the least fixed point of this operator at the context $\langle \Delta, \Gamma \rangle$.

The definition of such operator is founded on previous concepts and results, that are formulated now.

Definition 4. 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. In addition, $(\mathcal{I}, \sqsubseteq)$ is a complete lattice.

Lemma 1. *The poset $(\mathcal{I}, \sqsubseteq)$ is a complete lattice.*

Proof. It must be checked that, for any $S \subseteq \mathcal{I}$, the least upper bound and greatest lower bound of S, denoted by $\bigsqcup S$ and $\bigsqcap S$ respectively, exist. It can be proved that they are characterized by the following equations:

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

which in fact define interpretations. We now prove those facts for the case of $\bigsqcup S$, since the proof for $\bigsqcap S$ is analogous. The following elemental facts must be proved:

- $\bigsqcup S \in \mathcal{I}$. For Δ_1, Δ_2 and Γ_1, Γ_2, assuming that $\Delta_1 \subseteq \Delta_2, \Gamma_1 \subseteq \Gamma_2$ and $A \in (\bigsqcup S)(\Delta_1, \Gamma_1)$ hold, let us prove that $A \in (\bigsqcup S)(\Delta_2, \Gamma_2)$. $A \in (\bigsqcup S)(\Delta_1, \Gamma_1) = \bigcup_{I \in S} I(\Delta_1, \Gamma_1)$, so there exists $I_A \in S$ such that $A \in I_A(\Delta_1, \Gamma_1)$. Since each I_A is an interpretation, $I_A(\Delta_1, \Gamma_1) \subseteq I_A(\Delta_2, \Gamma_2)$, and therefore $A \in I_A(\Delta_2, \Gamma_2)$. But $I_A(\Delta_2, \Gamma_2) \subseteq \bigcup_{I \in S} I(\Delta_2, \Gamma_2) = (\bigsqcup S)(\Delta_2, \Gamma_2)$, so we obtain that $A \in (\bigsqcup S)(\Delta_2, \Gamma_2)$.
- $\bigsqcup S$ is an upper bound for S. Let $I \in S$. For any Δ and Γ, we have that $I(\Delta, \Gamma) \subseteq \bigcup_{I \in S} I(\Delta, \Gamma) = (\bigsqcup S)(\Delta, \Gamma)$. Therefore, $I \sqsubseteq \bigsqcup S$ for any $I \in S$.
- $\bigsqcup S \sqsubseteq I'$ for any $I' \in \mathcal{I}$ upper bound of S. Let us assume that I' is an upper bound for S. For each $I \in S$, $I \sqsubseteq I'$ implies $I(\Delta, \Gamma) \subseteq I'(\Delta, \Gamma)$ for any Δ and Γ. Therefore, $\bigcup_{I \in S} I(\Delta, \Gamma) \subseteq I'(\Delta, \Gamma)$ for any Δ and Γ. Thus, $\bigsqcup S \sqsubseteq I'$. $\square$

As a particular case, $(\mathcal{I}, \sqsubseteq)$ has an infimum $\bigsqcap \mathcal{I}$, denoted by $I_\perp$, which is the constant function $\emptyset$. Moreover, for any chain of interpretations $\{I_i\}_{i \geq 0}$, such that $I_0 \sqsubseteq I_1 \sqsubseteq I_2 \sqsubseteq \ldots$, $\bigsqcup_{i \geq 0} I_i(\Delta, \Gamma) = \bigcup_{i \geq 0} I_i(\Delta, \Gamma)$ for any Δ and Γ.

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

Definition 5. Given $I \in \mathcal{I}$, G, Δ and Γ, G is said to be *forced* by I, Δ and Γ, written $I, \Delta, \Gamma \Vdash G$, where $\Vdash$ is the relation recursively defined by the rules below:

$I, \Delta, \Gamma \Vdash C \overset{\text{def}}{\Longleftrightarrow} \Gamma \vdash_{\mathcal{C}} C$.
$I, \Delta, \Gamma \Vdash A \overset{\text{def}}{\Longleftrightarrow} A \in I(\Delta, \Gamma)$.
$I, \Delta, \Gamma \Vdash G_1 \wedge G_2 \overset{\text{def}}{\Longleftrightarrow} I, \Delta, \Gamma \Vdash G_i$ 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, $\forall x G$.
- $I, \Delta, \Gamma \Vdash G[y/x]$.

Now, we are ready to define the operator over interpretations whose least fixed point supplies the expected version of truth.

Definition 6. 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 variant $\forall \overline{x}(G \Rightarrow A')$ of a clause in Δ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $I, \Delta, \Gamma \Vdash \exists \overline{x}(A \approx A' \wedge G)$.

In order to establish the existence of a fixed point of T, it will be proved to be monotonous and continuous. The following lemmas are required in those proofs.

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

$$I_1, \Delta, \Gamma \Vdash G \Longrightarrow I_2, \Delta, \Gamma \Vdash G.$$

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

$C \in \mathcal{L}_{\mathcal{C}}$ $I_1, \Delta, \Gamma \Vdash C \iff \Gamma \vdash_{\mathcal{C}} C \iff I_2, \Delta, \Gamma \Vdash C$.

$A \in At$ $I_1, \Delta, \Gamma \Vdash A \iff A \in I_1(\Delta, \Gamma)$. $I_1 \sqsubseteq I_2$ implies that $I_1(\Delta, \Gamma) \subseteq I_2(\Delta, \Gamma)$, so $A \in I_2(\Delta, \Gamma)$ and therefore $I_2, \Delta, \Gamma \Vdash A$.

$G_1 \wedge G_2$ $I_1, \varDelta, \varGamma \Vdash G_1 \wedge G_2 \iff I_1, \varDelta, \varGamma \Vdash G_i$ for each $i \in \{1,2\}$. In both cases the induction hypothesis can be used, so $I_2, \varDelta, \varGamma \Vdash G_i$ for each $i \in \{1,2\}$, which implies that $I_2, \varDelta, \varGamma \Vdash G_1 \wedge G_2$.

$G_1 \vee G_2$ $I_1, \varDelta, \varGamma \Vdash G_1 \vee G_2 \iff$ there is $i \in \{1,2\}$ such that $I_1, \varDelta, \varGamma \Vdash G_i$. By induction hypothesis, $I_2, \varDelta, \varGamma \Vdash G_i$, hence $I_2, \varDelta, \varGamma \Vdash G_1 \vee G_2$.

$D \Rightarrow G'$ $I_1, \varDelta, \varGamma \Vdash D \Rightarrow G' \iff I_1, \varDelta \cup \{D\}, \varGamma \Vdash G'$. By induction hypothesis, $I_2, \varDelta \cup \{D\}, \varGamma \Vdash G'$ holds, so $I_2, \varDelta, \varGamma \Vdash D \Rightarrow G'$.

$C \Rightarrow G'$ $I_1, \varDelta, \varGamma \Vdash C \Rightarrow G' \iff I_1, \varDelta, \varGamma \cup \{C\} \Vdash G'$. By induction hypothesis, $I_2, \varDelta, \varGamma \cup \{C\} \Vdash G'$ holds, which implies that $I_2, \varDelta, \varGamma \Vdash C \Rightarrow G'$.

$\exists x G'$ $I_1, \varDelta, \varGamma \Vdash \exists x G' \iff$ there is a $\mathcal{C}$-constraint C and a variable y such that:
- y does not occur free in $\varDelta$, $\varGamma$, $\exists x G'$.
- $\varGamma \vdash_{\mathcal{C}} \exists y C$.
- $I_1, \varDelta, \varGamma \cup \{C\} \Vdash G'[y/x]$.

By induction hypothesis $I_2, \varDelta, \varGamma \cup \{C\} \Vdash G'[y/x]$, hence $I_2, \varDelta, \varGamma \Vdash \exists x G'$.

$\forall x G'$ $I_1, \varDelta, \varGamma \Vdash \forall x G' \iff$ there is a variable y such that:
- y does not occur free in $\varDelta$, $\varGamma$, $\forall x G'$
- $I_1, \varDelta, \varGamma \Vdash G'[y/x]$.

By induction hypothesis $I_2, \varDelta, \varGamma \Vdash G'[y/x]$, therefore $I_2, \varDelta, \varGamma \Vdash \forall x G'$. □

Lemma 3. *Let $\{I_i\}_{i \geq 0}$ be a denumerable family of interpretations such that $I_0 \sqsubseteq I_1 \sqsubseteq I_2 \sqsubseteq \ldots$. Then, for any $\varDelta, G$ and $\varGamma$, $\bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash G \iff$ there exists $k \geq 0$ such that $I_k, \varDelta, \varGamma \Vdash G$.*

Proof. The implication to the left is a consequence of Lemma 2, since $I_k \sqsubseteq \bigsqcup_{i \geq 0} I_i$ holds for any k. The converse is proved by induction on the structure of G. We will use that $(\bigsqcup_{i \geq 0} I_i)(\varDelta, \varGamma) = \bigcup_{i \geq 0} I_i(\varDelta, \varGamma)$.

$C \in \mathcal{L}_{\mathcal{C}}$ $\bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash C \iff \varGamma \vdash_{\mathcal{C}} C \iff I_k, \varDelta, \varGamma \Vdash C$ is true independently of $k \geq 0$.

$A \in At$ $\bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash A \iff A \in (\bigsqcup_{i \geq 0} I_i)(\varDelta, \varGamma) = \bigcup_{i \geq 0} I_i(\varDelta, \varGamma)$. Therefore, there exists $k \geq 0$ such that $A \in I_k(\varDelta, \varGamma)$, hence, for that k, $I_k, \varDelta, \varGamma \Vdash A$.

$G_1 \wedge G_2$ $\bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash G_1 \wedge G_2 \iff \bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash G_j$ 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 $I_{k_j}, \varDelta, \varGamma \Vdash G_j$ for each $j \in \{1,2\}$. Let $k = max(k_1, k_2)$. Then $I_k, \varDelta, \varGamma \Vdash G_j$ for each $j \in \{1,2\}$ in virtue of Lemma 2, and therefore $I_k, \varDelta, \varGamma \Vdash G_1 \wedge G_2$.

$G_1 \vee G_2$ $\bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash G_1 \vee G_2 \iff$ there is $j \in \{1,2\}$ such that $\bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash G_j$. The induction hypothesis can be used, so there exist $k \geq 0$ such that $I_k, \varDelta, \varGamma \Vdash G_j$, and therefore $I_k, \varDelta, \varGamma \Vdash G_1 \vee G_2$.

$D \Rightarrow G'$ $\bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash D \Rightarrow G' \iff \bigsqcup_{i \geq 0} I_i, \varDelta \cup \{D\}, \varGamma \Vdash G' \Longrightarrow$ there is $k \geq 0$ such that $I_k, \varDelta \cup \{D\}, \varGamma \Vdash G'$, by induction hypothesis $\Longrightarrow$ there is $k \geq 0$ such that $I_k, \varDelta, \varGamma \Vdash D \Rightarrow G'$.

$C \Rightarrow G'$ $\bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \Vdash C \Rightarrow G' \iff \bigsqcup_{i \geq 0} I_i, \varDelta, \varGamma \cup \{C\} \Vdash G' \Longrightarrow$ there is $k \geq 0$ such that $I_k, \varDelta, \varGamma \cup \{C\} \Vdash G'$, by induction hypothesis $\Longrightarrow$ there is $k \geq 0$ such that $I_k, \varDelta, \varGamma \Vdash C \Rightarrow G'$.

$\exists x G'$ $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash \exists x G' \iff$ 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 $I_k, \Delta, \Gamma \Vdash \exists x G'$ for some $k \geq 0$.

$\forall x G'$ $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash \forall x G' \iff$ 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 $I_k, \Delta, \Gamma \Vdash G'[y/x]$. Hence $I_k, \Delta, \Gamma \Vdash \forall x G'$ for some $k \geq 0$. □

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 Δ, 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 $T(I_1) \sqsubseteq T(I_2)$. □

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.

$\subseteq$) Consider any 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 in Δ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $\bigsqcup_{i\geq 0} I_i, \Delta, \Gamma \Vdash \exists \overline{x}(A \approx A' \wedge G)$. Thanks to Lemma 3, there exists $k \geq 0$ such that $I_k, \Delta, \Gamma \Vdash \exists \overline{x}(A \approx A' \wedge G)$, and therefore $A \in T(I_k)(\Delta, \Gamma)$. As a consequence, $T(\bigsqcup_{i\geq 0} I_i)(\Delta, \Gamma) \subseteq \bigcup_{i\geq 0} T(I_i)(\Delta, \Gamma) = (\bigsqcup_{i\geq 0} T(I_i))(\Delta, \Gamma)$. This happens for every Δ and Γ, thus $T(\bigsqcup_{i\geq 0} I_i) \sqsubseteq \bigsqcup_{i\geq 0} T(I_i)$. □

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, 5 and the Knaster-Tarski fixed point theorem [18]. □

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

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

Fig. 1. Steps leading to $T^4(I_\perp), \Delta, \Gamma \Vdash fib(2,x)$.

Example 5. Let Δ 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 $\Gamma = \{z_1 \approx 1, z_2 \approx 1, x \approx z_1 + z_2\}$, $\Delta' = \Delta \cup \{mf(0,1), mf(1,1)\}$ and $\Delta'' = \Delta \cup \{mf(0,1), mf(1,1), mf(2, z_1 + z_2)\}$.

The chart shows the main steps leading to $T^4(I_\perp), \Delta, \Gamma \Vdash fib(2,x)$. *memfib* is abbreviated with *mf*, and *getfib* with *gf*.

This is not difficult to see, if such forcing relations are verified from left to right. For instance,

$$T(I_\perp), \Delta', \Gamma \Vdash mf(0, z_1)$$

is checked in one step, since $mf(0,1) \in \Delta'$ and $\Gamma \vdash_{\mathcal{C}} z \approx 1$. The forcing relations in each column justify those in the next. In order to verify that an existential quantification is forced, it is required to introduce new fresh variables. However, for the sake of readability, equivalent and more simple expressions have been used instead.

Bear in mind that only some of the formulas forced are gathered in the table, as it is indicated by the ellipses. In particular, since interpretations are monotonous, any formula present in a position of such table is automatically present in the whole rectangle that has that position as top-left corner.

Soundness and completeness The following theorems establish the full connection between the fixed point semantics presented and the calculus $\mathcal{UC}$. The definitions below correspond to technicalities that will be used in the proof of such soundness and completeness result.

Let $\mathcal{S} = \{\langle \Delta, \Gamma, G \rangle \in \mathcal{W} \times \mathcal{P}(\mathcal{L}_{\mathcal{C}}) \times \mathcal{G} \mid lfp(T), \Delta, \Gamma \Vdash G\}$. The function $ord : \mathcal{S} \longrightarrow \mathbb{N}$ is defined as follows. Given any $\langle \Delta, \Gamma, G \rangle \in \mathcal{S}$, Lemma 3 guarantees that the set of natural numbers k such that $T^k(I_\perp), \Delta, \Gamma \Vdash G$ 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 renaming of a strict subformula of 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,*

$$\Delta; \Gamma \vdash_{\mathcal{UC}} G \Longrightarrow \mathit{lfp}(T), \Delta, \Gamma \Vdash G.$$

Proof. Since this is one of the main results presented, the whole proof is included. 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 $\mathit{lfp}(T), \Delta, \Gamma \Vdash C$ holds.

Inductive case. We suppose that $\Delta; \Gamma \mid\!\!\!— G$ has a proof of height h. Let us prove $\mathit{lfp}(T), \Delta, \Gamma \Vdash G$ by case analysis on the $\mathcal{UC}$-rule employed in the bottom of such proof.

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

($\wedge_R$) There must exist goals G_1, G_2 such that $G \equiv G_1 \wedge G_2$ and the sequents $\Delta; \Gamma \mid\!\!\!— G_i$ has a proof of height less than h for each $i \in \{1, 2\}$. By induction hypothesis, $\mathit{lfp}(T), \Delta, \Gamma \Vdash G_i$ is assumed for each $i \in \{1, 2\}$ and, as a consequence, $\mathit{lfp}(T), \Delta, \Gamma \Vdash G$.

($\vee_R$) There must exist goals G_1, G_2 such that $G \equiv G_1 \vee G_2$ and the sequent $\Delta; \Gamma \mid\!\!\!— G_i$ has a proof of height $h - 1$ for some $i \in \{1, 2\}$. By induction hypothesis, $\mathit{lfp}(T), \Delta, \Gamma \Vdash G_i$ holds, which implies $\mathit{lfp}(T), \Delta, \Gamma \Vdash G$.

($\Rightarrow_R$) Then $G \equiv D \Rightarrow G'$ and the sequent $\Delta, D; \Gamma \mid\!\!\!— G'$ has a proof of height $h-1$. By induction hypothesis, $\mathit{lfp}(T), \Delta \cup \{D\}, \Gamma \Vdash G'$. Therefore $\mathit{lfp}(T), \Delta, \Gamma \Vdash D \Rightarrow G'$.

($\Rightarrow_{C_R}$) Now $G \equiv C \Rightarrow G'$ and the sequent $\Delta; \Gamma, C \mid\!\!\!— G'$ has a proof of height $h-1$. By induction hypothesis, $\mathit{lfp}(T), \Delta, \Gamma \cup \{C\} \Vdash G'$, therefore $\mathit{lfp}(T), \Delta, \Gamma \Vdash C \Rightarrow G'$.

($\exists_R$) Then $G \equiv \exists x G'$, and there must exist a constraint C and a variable y not occurring free in Δ, Γ, $\exists x G'$, such that $\Delta; \Gamma, C \mid\!\!\!— G'[y/x]$ has a proof of height $h - 1$ and $\Gamma \vdash_{\mathcal{C}} \exists y C$. Then $\mathit{lfp}(T), \Delta, \Gamma \cup \{C\} \Vdash G'[y/x]$, by induction hypothesis, and therefore $\mathit{lfp}(T), \Delta, \Gamma \Vdash \exists x G'$.

($\forall_R$) 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 $\Delta; \Gamma \mid\!\!\!— G'[y/x]$ has a proof of height $h - 1$. By induction hypothesis, $\mathit{lfp}(T), \Delta, \Gamma \Vdash G'[y/x]$ and, as a consequence, $\mathit{lfp}(T), \Delta, \Gamma \Vdash \forall x G'$.

Theorem 3. *For any Δ, Γ and G,*

$$\mathit{lfp}(T), \Delta, \Gamma \Vdash G \Longrightarrow \Delta; \Gamma \vdash_{\mathcal{UC}} G.$$

Proof. By induction on the structural 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 that $\Delta'; \Gamma' \vdash_{\mathcal{UC}} G'$. Then, let us conclude $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ by case analysis on the structure of G.

$C \in \mathcal{L}_{\mathcal{C}}$ If $\langle \Delta, \Gamma, C\rangle \in \mathcal{S}$ then $\Gamma \vdash_{\mathcal{C}} C$, therefore $\Delta; \Gamma \vdash_{\mathcal{UC}} C$ by (C_R).

$A \in At$ $\langle \Delta, \Gamma, A\rangle \in \mathcal{S}$ implies that $\mathit{lfp}(T), \Delta, \Gamma \Vdash A$. Let $k = \mathit{ord}(\langle \Delta, \Gamma, A\rangle)$, then $T^k(I_\perp), \Delta, \Gamma \Vdash A$, which is equivalent to $A \in (T^k(I_\perp))(\Delta, \Gamma)$. Hence there is a variant $\forall \overline{x}(G' \Rightarrow A')$ of a clause of Δ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $T^{k-1}(I_\perp), \Delta, \Gamma \Vdash \exists \overline{x}(A \approx A' \wedge G')$. In this reason, $\langle \Delta, \Gamma, \exists \overline{x}(A \approx A' \wedge G')\rangle < \langle \Delta, \Gamma, A\rangle$, 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 elaboration $\forall \overline{x}(G' \Rightarrow A')$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} A$.

$G_1 \wedge G_2$ Then $\langle \Delta, \Gamma, G_1 \wedge G_2\rangle \in \mathcal{S}$ implies that $\mathit{lfp}(T), \Delta, \Gamma \Vdash G_i$ for each $i \in \{1, 2\}$. Clearly, $\mathit{ord}(\langle \Delta, \Gamma, G_1 \wedge G_2\rangle) = \mathit{ord}(\langle \Delta, \Gamma, G_1\rangle = \mathit{ord}(\langle \Delta, \Gamma, G_2\rangle)$ and G_1, G_2 are strict subformulas of $G_1 \wedge G_2$, hence $\langle \Delta, \Gamma, G_i\rangle < \langle \Delta, \Gamma, G_1 \wedge G_2\rangle$, for each $i \in \{1, 2\}$. Then, by the induction hypothesis, $\Delta; \Gamma \vdash_{\mathcal{UC}} G_i$, for each $i \in \{1, 2\}$. So $\Delta; \Gamma \vdash_{\mathcal{UC}} G_1 \wedge G_2$, applying the rule $(\wedge_R)$.

$G_1 \vee G_2$ $\langle \Delta, \Gamma, G_1 \vee G_2\rangle \in \mathcal{S}$ implies that $\mathit{lfp}(T), \Delta, \Gamma \Vdash G_i$ for some $i \in \{1, 2\}$. It is true that $\mathit{ord}(\langle \Delta, \Gamma, G_1 \vee G_2\rangle) = \mathit{ord}(\langle \Delta, \Gamma, G_i\rangle)$, G_i is a strict subformula of $G_1 \vee G_2$ and, as a consequence, $\langle \Delta, \Gamma, G_i\rangle < \langle \Delta, \Gamma, G_1 \vee G_2\rangle$. Therefore, by the induction hypothesis we obtain $\Delta; \Gamma \vdash_{\mathcal{UC}} G_i$ for some $i \in \{1, 2\}$. Thanks to the rule $(\vee_R)$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} G_1 \vee G_2$.

$D \Rightarrow G'$ Now $\langle \Delta, \Gamma, D \Rightarrow G'\rangle \in \mathcal{S}$ implies that $\mathit{lfp}(T), \Delta \cup \{D\}, \Gamma \Vdash G'$. Clearly, $\mathit{ord}(\langle \Delta, \Gamma, D \Rightarrow G'\rangle) = \mathit{ord}(\langle \Delta \cup \{D\}, \Gamma, G'\rangle)$ and G' is a strict subformula of $D \Rightarrow G'$, so $\langle \Delta \cup \{D\}, \Gamma, G'\rangle < \langle \Delta, \Gamma, D \Rightarrow G'\rangle$. Therefore, by the induction hypothesis, $\Delta, D; \Gamma \vdash_{\mathcal{UC}} G'$. Thanks to the rule $(\Rightarrow_R)$, it follows that $\Delta; \Gamma \vdash_{\mathcal{UC}} D \Rightarrow G'$.

$C \Rightarrow G'$ Then $\langle \Delta, \Gamma, C \Rightarrow G'\rangle \in \mathcal{S}$ implies that $\mathit{lfp}(T), \Delta, \Gamma \cup \{C\} \Vdash G'$. Clearly, $\mathit{ord}(\langle \Delta, \Gamma, C \Rightarrow G'\rangle) = \mathit{ord}(\langle \Delta, \Gamma \cup \{C\}, G'\rangle)$ and G' is a strict subformula of $C \Rightarrow G'$, so $\langle \Delta, \Gamma \cup \{C\}, G'\rangle < \langle \Delta, \Gamma, C \Rightarrow G'\rangle$. Then, by the induction hypothesis, $\Delta; \Gamma, C \vdash_{\mathcal{UC}} G'$, and $\Delta; \Gamma \vdash_{\mathcal{UC}} C \Rightarrow G'$ due to the rule $(\Rightarrow_{C_R})$.

$\exists x G'$ Then $\langle \Delta, \Gamma, \exists x G'\rangle \in \mathcal{S}$ implies that there is a constraint C and a variable y such that:

- y does not occur free in Δ, Γ, $\exists x G'$.
- $\Gamma \vdash_{\mathcal{C}} \exists y C$.
- $\mathit{lfp}(T), \Delta, \Gamma \cup \{C\} \Vdash G'[y/x]$.

$\mathit{ord}(\langle \Delta, \Gamma, \exists x G'\rangle) = \mathit{ord}(\langle \Delta, \Gamma \cup \{C\}, G'[y/x]\rangle)$ by definition, and $G'[y/x]$ is a renaming of a strict subformula of $\exists x G'$, so $\langle \Delta, \Gamma \cup \{C'\}, G'[y/x]\rangle < \langle \Delta, \Gamma, \exists x G'\rangle$. Therefore $\Delta; \Gamma, C' \vdash_{\mathcal{UC}} G'[y/x]$ by the induction hypothesis. Hence $\Delta; \Gamma \vdash_{\mathcal{UC}} \exists x G'$, by using the rule $(\exists_R)$.

$\forall x G'$ Then $\langle \Delta, \Gamma, \forall x G'\rangle \in \mathcal{S}$ implies that there is a variable y such that:

- y does not occur free in Δ, Γ, $\forall x G'$.
- $\mathit{lfp}(T), \Delta, \Gamma \Vdash G'[y/x]$.

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

This fixed point semantics supplies a framework in which properties of programs can be easily analyzed. For instance, the behavior of 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 6. Let $\Delta =$

```
p(x) :- x >= 0.
p(x) :- x < 0.
```

and $\Delta' =$ `p(x) :- x >= 0; x < 0.`
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 since 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 $\Delta \supseteq$

```
p(x) :- q(x).
p(x) :- q'(x).
```

and if $\Delta' \supseteq$ `p(x) :- q(x) ; q'(x).`
then Δ and Δ' could be equivalent.

4.2 Another fixed point semantics approach

We have just described a fixed point semantics for $HH(\mathcal{C})$. 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 5. This semantics is defined for any general constraint system $\mathcal{C}$. The conditions imposed in Subsection 2.1 are meant as minimal requirements for a $\mathcal{C}$ to be a constraint system, but in many useful cases $\mathcal{C}$ satisfies additional properties, as it was mentioned there. For instance, the entailment relation $\vdash$ referred in such subsection is known to be sound and complete w.r.t. the standard semantic relation $\models_{\approx}$ of first-order logic with equality, hence, for the instance $HH(\mathcal{R})$, requirements like $\Gamma \vdash_{\mathcal{R}} C$ can be directly replaced by $\Gamma \cup Ax_{\mathcal{R}} \models_{\approx} C$, in the definition of the forcing relation.

As in the frame of *CLP*, we are interested in finding general conditions for the constraint systems that would guarantee the existence of semantics for constraints based on a model theory, in order to incorporate it into the fixed point semantics of logic programs.

More precisely, the semantics of constraint logic programs are usually based on the assumption that: the domain of computation (model), which is the structure used to interpret the constraints; the solver, which checks whether constraints are $\mathcal{C}$-satisfiable; and the constraint theory, that describes the logical semantics of the constraints, *agree*. See [9] for details.

From now on we will focus on constraint systems $\mathcal{C}$ for which an additional condition is required: there is a standard structure $\mathcal{A}_{\mathcal{C}}$ such that $\vdash_{\mathcal{C}}$ and $\mathcal{A}_{\mathcal{C}}$ *agree* in a sense, similar to that of [9], that will be specified soon. Additional notation involving standard structures is now introduced for that purpose.

Given a standard structure $\mathcal{A}_{\mathcal{C}}$ over a signature Σ, which interprets the symbols of Σ, and with domain AC, an assignment (for $\mathcal{A}_{\mathcal{C}}$) is a function $\nu : V \rightarrow AC$, where V is a set of variables. $V = dom(\nu)$ is said to be the domain of ν. *Assig* is the set of assignments.

Given $\nu \in Assig$, a variable $y \notin dom(\nu)$ and $a \in AC$, $\nu[y \leftarrow a]$ is the assignment with domain $dom(\nu) \cup \{y\}$ such that

$$\nu[y \leftarrow a](x) \stackrel{\text{def}}{=} \begin{cases} a, & \text{if } x \equiv y \\ \nu(x), & otherwise, \end{cases}$$

and it is said to be an *extension* of ν to y.

Given a first-order formula F over Σ, $[\![F]\!]_{\nu}^{\mathcal{A}_{\mathcal{C}}} \in \{true, false\}$ is the classical truth value of the formula F in the model $\mathcal{A}_{\mathcal{C}}$ under the assignment ν.

Definition 7. Let $\mathcal{C}$ be a constraint system and $\mathcal{A}_{\mathcal{C}}$ be a standard structure over Σ. $\mathcal{A}_{\mathcal{C}}$ and $\vdash_{\mathcal{C}}$ *agree* if for any Γ, ν and C, $\Gamma \vdash_{\mathcal{C}} C$ if and only if $[\![\bigwedge \Gamma \Rightarrow C]\!]_{\nu}^{\mathcal{A}_{\mathcal{C}}} = true$[4].

Intuitively, this means that the entailment in the constraint system can be identified with (the universal closure of) the implication, in that specific structure.

Constraints will be interpreted by the sets of assignments (for such $\mathcal{A}_{\mathcal{C}}$) that make them true. Formally, given a constraint C, the set $[\![C]\!]$ is defined as follows:

$$[\![C]\!] = \{\nu \in Assig \mid dom(\nu) \supseteq free(C) \text{ and } [\![C]\!]_{\nu} = true\}^{5}.$$

Such definition is extended to finite sets of constraints in the natural way, i.e. $[\![\Gamma]\!] = [\![\bigwedge \Gamma]\!]$. Furthermore, in some cases it will be necessary that the domains

[4] Hereafter, the superscript $\mathcal{A}_{\mathcal{C}}$ may be omitted if it is clear from the context.

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

of such assignments include specific sets of variables, and in this reason we define: $[\![\Gamma]\!]_V = \{\nu \in [\![\Gamma]\!] \mid dom(\nu) \supseteq V\}$, where V is any set of variables.

Notice that if $\mathcal{A}_\mathcal{C}$ and $\vdash_\mathcal{C}$ agree, then $\Gamma \vdash_\mathcal{C} C \iff [\![\bigwedge \Gamma]\!]_{free(C)} \subseteq [\![C]\!]$, and that Γ is $\mathcal{C}$-satisfiable iff $[\![\Gamma]\!] \neq \emptyset$.

For instance, let $\mathcal{A}_\mathcal{R}$ be the Σ-structure whose domain is $\mathbb{R}$ and that interprets constants for real numbers and arithmetic symbols in the natural way. Then $\mathcal{A}_\mathcal{R}$ and $\mathcal{R}$ agree.

Example 7. Consider $\mathcal{C} = \mathcal{R}$ and the Σ-structure $\mathcal{A}_\mathcal{R}$ above. If $C \equiv x * x + y * y \approx 1$, then $[\![C]\!] = \{\nu : \{x, y\} \to \mathbb{R}^2 \mid \nu(x)^2 + \nu(y)^2 = 1\}$. Once each variable is associated to a coordinates 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 in the origin of the real plane. Thus, the syntactic object $x * x + y * y \approx 1$ is replaced in the forcing relation by such circle, which is its intended meaning in $\mathcal{A}_\mathcal{R}$.

The new semantics we will present combines a notion of forcing similar to that used in Subsection 4.1 with the classical structure considered for $\mathcal{C}$. New definitions are needed for the concepts of forcing and interpretation.

The notion of $\mathcal{C}$-interpretation of the algebraic semantics provided in [9] associates sets of expressions of the form $p(a_1, \ldots, a_n)$ to programs, where each a_i belongs to the domain of the structure, and p is a program predicate symbol. The approach followed in this paper is close to that, because our $\mathcal{C}$-interpretations associate to each pair $\langle \Delta, \nu\rangle$ a set of not necessarily ground atoms, which can be assigned to elements of the domain of the structure via ν.

$\mathcal{C}$-interpretations and guided forcing relations As in the case of $\Vdash$, we are looking for a model $I^\mathcal{C}$ and a relation $\Vdash^\mathcal{C}$ such that $\Delta; \Gamma \vdash_{\mathcal{UC}} G$ iff $I^\mathcal{C}, \Delta, [\![\Gamma]\!] \Vdash^\mathcal{C} G$. Some technicalities, needed in the proof of such result, will be promptly presented. In fact, the equivalence between $\Vdash^\mathcal{C}$ and $\vdash_{\mathcal{UC}}$ is proved connecting $\Vdash^\mathcal{C}$ with $\Vdash$, and using the equivalence between $\Vdash$ and $\vdash_{\mathcal{UC}}$. But such connection is established defining two *guided* versions of these forcing relations, denoted by $\Vdash^\mathcal{C}_\tau$ and $\Vdash_\tau$, respectively, where τ is an index that play the role of guide.

The introduction of those forcing relations demands the definition and manipulation of several notions of interpretations and fixed point operators. The Figure 2 may help to identify the notation and to understand the connection between the different induced semantics.

The index τ of the guided versions is closely related to the structure of goals. The formal definition is the following:

Definition 8. The set of *structural trees* $\mathcal{T}$, with elements τ, is recursively defined by the rule:
$\tau ::= cst \mid and(\tau_1, \tau_2) \mid or(i, \tau) \mid imp(\tau) \mid impc(\tau) \mid cl(n, \tau) \mid exists(\tau) \mid forall(\tau)$, where $i \in \{1, 2\}$ and $n \in \mathbb{N}$.

$\mathcal{T}^{cl} \subseteq \mathcal{T}$ is the set of trees with the form $cl(n, \tau)$, where $\tau \in \mathcal{T}$ and $n \in \mathbb{N}$.

$$\begin{tikzcd}
 & \text{Interpretations} & & \mathcal{C}\text{-interpretations} \\
\text{non guided} & \langle \mathcal{I}, \Vdash \rangle \arrow[d, leftrightarrow, "\text{Corollary 4}"] & & \langle \mathcal{I}^{\mathcal{C}}, \Vdash^{\mathcal{C}} \rangle \arrow[d, leftrightarrow, "\text{Definition 13}"] \\
\text{guided} & \langle \mathcal{I}_{\mathcal{T}}, \Vdash_{\mathcal{T}} \rangle \arrow[rr, leftrightarrow, "\text{Propositions 1,2}"] & & \langle \mathcal{I}^{\mathcal{C}}, \Vdash^{\mathcal{C}}_{\tau} \rangle
\end{tikzcd}$$

Fig. 2. Different fixed point semantics

A new notion of interpretation is provided, because now context are not pairs $\langle \Delta, \Gamma \rangle$, but pairs $\langle \Delta, \nu \rangle$. On the other hand, another remarkable difference arises: in the range of the interpretations, each atom is tagged with a tree of $\mathcal{T}^{cl}$.

Definition 9. A *$\mathcal{C}$-interpretation* $I^{\mathcal{C}}$ is a function $I^{\mathcal{C}} : \mathcal{W} \times Assig \to \mathcal{P}(At \times \mathcal{T}^{cl})$ that is monotonous, i.e. for any Δ_1, Δ_2 and ν_1, ν_2, if $\Delta_1 \subseteq \Delta_2$ and $\nu_1 = \nu_2|_{dom(\nu_1)}$ then $I^{\mathcal{C}}(\Delta_1, \nu_1) \subseteq I^{\mathcal{C}}(\Delta_2, \nu_2)$. Moreover, $\langle A, cl(n, \tau) \rangle \in I^{\mathcal{C}}(\Delta, \nu)$ implies that $free(\Delta \cup \{A\}) \subseteq dom(\nu)$.

Let $\mathcal{I}^{\mathcal{C}}$ be the set of these $\mathcal{C}$-interpretations.

A preorder $\sqsubseteq$ can be defined for $\mathcal{I}^{\mathcal{C}}$ similar to such of $\mathcal{I}$, $(\mathcal{I}^{\mathcal{C}}, \sqsubseteq)$ is a complete lattice and his infimum, denoted $I_{\perp}{}^{\mathcal{C}}$ is the constant function $\emptyset$.

The guided forcing relation $\Vdash^{\mathcal{C}}_{\tau}$ is defined from the concept of $\mathcal{C}$-interpretation, but such definition requires the notion of accordance below.

Definition 10. Given Δ, G and τ, τ is said to be *in accordance* with $\langle \Delta, G \rangle$ if one of the cases below applies:

- $\tau = cst$ and $G \equiv C$.
- $\tau = cl(n, \tau')$, $G \equiv A$ and there is a variant $\forall \overline{x}(G' \Rightarrow A')$ of the n^{th} clause of Δ such that $\overline{x}$ do not occur in Δ, A, and τ' is in accordance with $\langle \Delta, \exists \overline{x}(A \approx A' \wedge G') \rangle$
- $\tau = and(\tau_1, \tau_2)$, $G \equiv G_1 \wedge G_2$ and τ_i is in accordance with $\langle \Delta, G_i \rangle$ for each $i \in \{1, 2\}$.
- $\tau = or(i, \tau')$, $G \equiv G_1 \vee G_2$ and τ' is in accordance with $\langle \Delta, G_i \rangle$.
- $\tau = impd(\tau')$, $G \equiv D \Rightarrow G'$ and τ' is in accordance with $\langle \Delta \cup \{D\}, G' \rangle$.
- $\tau = impc(\tau')$, $G \equiv C \Rightarrow G'$ and τ' is in accordance with $\langle \Delta, G' \rangle$.
- $\tau = exists(\tau')$, $G \equiv \exists x G'$ and τ' is in accordance with $\langle \Delta, G'[y/x] \rangle$ for any y not free in Δ, G.
- $\tau = forall(\tau')$, $G \equiv \forall x G'$ and τ' is in accordance with $\langle \Delta, G'[y/x] \rangle$ for any y not free in Δ, G.

Definition 11. Given G, $I^{\mathcal{C}} \in \mathcal{I}^{\mathcal{C}}$, Δ and ν such that $free(\Delta \cup \{G\}) \subseteq dom(\nu)$, G is said to be *forced by $I^{\mathcal{C}}, \Delta$ and ν with the guide* τ, written $I^{\mathcal{C}}, \Delta, \nu \Vdash^{\mathcal{C}}_{\tau} G$, if τ is in accordance with $\langle \Delta, G \rangle$ and $\Vdash^{\mathcal{C}}_{\tau}$ satisfies the recursive rules below.

$I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{cst} C \overset{\text{def}}{\Longleftrightarrow} \nu \in [\![C]\!]$.
$I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{cl(n,\tau)} A \overset{\text{def}}{\Longleftrightarrow} \langle A, cl(n,\tau)\rangle \in I^{\mathcal{C}}(\varDelta, \nu)$.
$I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{and(\tau_1,\tau_2)} G_1 \wedge G_2 \overset{\text{def}}{\Longleftrightarrow} I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{\tau_i} G_i$ for each $i \in \{1, 2\}$.
$I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{or(i,\tau)} G_1 \vee G_2 \overset{\text{def}}{\Longleftrightarrow} I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{\tau} G_i$.
$I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{imp(\tau)} D \Rightarrow G \overset{\text{def}}{\Longleftrightarrow} I^{\mathcal{C}}, \varDelta \cup \{D\}, \nu \Vdash^{\mathcal{C}}_{\tau} G$.
$I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{impc(\tau)} C \Rightarrow G \overset{\text{def}}{\Longleftrightarrow} \nu \notin [\![C]\!]$ or $I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{\tau} G$.
$I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{exists(\tau)} \exists x G \overset{\text{def}}{\Longleftrightarrow}$ given a variable y such that $y \notin dom(\nu)$, there is a ν' extension of ν to y such that $I^{\mathcal{C}}, \varDelta, \nu' \Vdash^{\mathcal{C}}_{\tau} G[y/x]$.
$I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{forall(\tau)} \forall x G \overset{\text{def}}{\Longleftrightarrow}$ given a variable $y \notin dom(\nu)$, for any ν' extension of ν to y, $I^{\mathcal{C}}, \varDelta, \nu' \Vdash^{\mathcal{C}}_{\tau} G[y/x]$.

From this definition it is followed that the label τ is narrowly connected with the structure of the goal and with the clauses used to prove it.

Intuitively, the subscript τ in $\Vdash^{\mathcal{C}}_{\tau}$ plays the role of guide, fixing the choice for the case when the goal is a disjunction or an atom.

This notion must be extended to sets of assignments, which henceforth are denoted by $\varTheta$, as follows.

Definition 12. Given $I^{\mathcal{C}}, \varDelta, \tau$ and $\varTheta \subseteq Assig$, a goal G is said to be *forced by* $I^{\mathcal{C}}, \varDelta$ *and* $\varTheta$*,with the guide* τ, written $I^{\mathcal{C}}, \varDelta, \varTheta \Vdash^{\mathcal{C}}_{\tau} G$, if $I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{\tau} G$ for each $\nu \in \varTheta$.

Now the non guided forcing relation $\Vdash^{\mathcal{C}}$ can be defined:

Definition 13. Given $\varDelta$, $I^{\mathcal{C}}$ and $\varTheta \subseteq Assig$, a goal G is said to be *forced by* $I^{\mathcal{C}}, \varDelta$ *and* $\varTheta$, written $I^{\mathcal{C}}, \varDelta, \varTheta \Vdash^{\mathcal{C}} G$, if there exists τ such that $I^{\mathcal{C}}, \varDelta, \varTheta \Vdash^{\mathcal{C}}_{\tau} G$.

The particular model we are looking for, fixing the connection between $\Vdash^{\mathcal{C}}$ and $\vdash_{\mathcal{C}}$, will be defined as the least fixed point of the operator over $\mathcal{C}$-interpretations defined below. In this definition, and in the rest of the paper, let us assume that, any program $\varDelta$ has its clauses ordered by an enumeration, and when a clause is added to $\varDelta$, it will be the last one in the order. We will frequently refer to the n^{th} clause of $\varDelta$ according to that enumeration.

Definition 14. The *operator* $T^{\mathcal{C}} : \mathcal{I}^{\mathcal{C}} \longrightarrow \mathcal{I}^{\mathcal{C}}$ transforms $\mathcal{C}$-interpretations as follows. For any $I^{\mathcal{C}} \in \mathcal{I}^{\mathcal{C}}$, $\varDelta$, ν, τ and A such that *free*$(\varDelta \cup \{A\}) \subseteq dom(\nu)$, $\langle A, cl(n,\tau)\rangle \in T^{\mathcal{C}}(I^{\mathcal{C}})(\varDelta, \nu)$ if

- $\forall \overline{x}(G \Rightarrow A')$ is a variant D of the n^{th} clause of $\varDelta$ and, for each $x \in \overline{x}$, $x \notin dom(\nu)$.
- $I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{\tau} \exists \overline{x}(A \approx A' \wedge G)$.

Notice that the subscript τ fixes which clause may be used to prove that an atom is forced. Therefore, in order to check $I^{\mathcal{C}}, \varDelta, \nu \Vdash^{\mathcal{C}}_{\tau} G$ for some $I^{\mathcal{C}}, \varDelta, \nu, \tau$ and G, no choice is possible, since all of them are gathered in τ.

The operator $T^{\mathcal{C}}$ is proved to be monotonous and continuous. Such proofs are analogous to those for T (Lemmas 4 and 5). Therefore, $T^{\mathcal{C}}$ has a least fixed point, which is $\bigsqcup_{i\geq 0} {T^{\mathcal{C}}}^i({I_\bot}^{\mathcal{C}})$, and we denote it by $\mathit{lfp}(T^{\mathcal{C}})$.

The following examples intend to motivate and illustrate the behavior of the operator $T^{\mathcal{C}}$, as well as the meaning of programs w.r.t. $\Vdash^{\mathcal{C}}_{\tau}$.

Example 8. This is a very simple example showing the necessity of τ in the definition of $\mathcal{I}^{\mathcal{C}}$ and $\Vdash^{\mathcal{C}}_{\tau}$. Choosing the constraint system $\mathcal{R}$ and the structure $\mathcal{A}_{\mathcal{R}}$, let us consider the program Δ in Example 6 and the goal $G \equiv \forall y\, p(y)$. For such program and goal $\Delta; \emptyset \not\vdash_{\mathcal{UC}} G$. Let us suppose that no τ was used, and so interpretations map pairs $\langle \Delta, \nu\rangle$ to sets of atoms. Such hypothetical interpretations and the correspondent operator will be overlined. Let $r \in \mathbb{R}$, then $\overline{T}(\overline{I}_\bot)(\Delta, [y \leftarrow r])$ would contain the atom $p(y)$ if $r \geq 0$, thanks to the first clause. But using to the second one, it would be also happen when $r < 0$. Therefore, if the forcing for a goal $\forall x G$ is defined only in terms of the forcing for G, it seems impossible to avoid that $\overline{T}(\overline{I}_\bot), \Delta, \emptyset$ would force $\forall y\, p(y)$. The intuitionism imposes that, in order to force $\forall y\, p(y)$ from Δ and any Γ, the atom $p(y)$ must be forced by all the assignments $[y \leftarrow r]_{r\in\mathbb{R}}$ and using the same clause. Therefore, it is necessary to store information regarding how atoms have been forced. That is why atoms A have been replaced by pairs $\langle A, \mathit{cl}(n,\tau)\rangle$. For this example, let $\tau_i = \mathit{forall}(\mathit{cl}(i, \mathit{cst}))$ for $i \in \{1, 2\}$. It is easy to check that $\langle p(y), \tau_1\rangle \in T^{\mathcal{C}}({I_\bot}^{\mathcal{C}})(\Delta, [y \leftarrow r])$ if $r \geq 0$, and $\langle p(y), \tau_2\rangle \in T^{\mathcal{C}}({I_\bot}^{\mathcal{C}})(\Delta, [y \leftarrow r])$ if $r < 0$, but that does not lead to the fact that, for some τ, $T^{\mathcal{C}}({I_\bot}^{\mathcal{C}}), \Delta, \emptyset \Vdash^{\mathcal{C}}_{\tau} \forall y\, p(y)$.

Example 9. Let us consider the program Δ of the instance $\mathit{HH}(\mathcal{R})$:

```
circle(X, Y) :- X * X + Y * Y < 1.
parab(X, Y)  :- X > 0, Y > 0, Y * Y < X.
sector(X, Y) :- circle(X, Y), parab(X, Y), X > 0.5.
```

Let us try to find for which τ and assignments $[x \leftarrow r]$, $r \in \mathbb{R}$,
$\mathit{lfp}(T^{\mathcal{C}}), \Delta, [x \leftarrow r] \Vdash^{\mathcal{C}}_{\tau} \forall y((0.1 < y \wedge y < 0.2) \Rightarrow \mathit{sector}(x, y))$.
Let $\tau = \tau_0$. That happens iff $\tau_0 = \mathit{forall}(\tau_1)$ and
$\mathit{lfp}(T^{\mathcal{C}}), \Delta, [x \leftarrow r, y \leftarrow s_0] \Vdash^{\mathcal{C}}_{\tau_1} ((0.1 < y \wedge y < 0.2) \Rightarrow \mathit{sector}(x, y))$
for each $s_0 \in \mathbb{R} \iff \tau_1 = \mathit{impc}(\tau_2)$ and
$\mathit{lfp}(T^{\mathcal{C}}), \Delta, [x \leftarrow r, y \leftarrow s_0] \Vdash^{\mathcal{C}}_{\tau_2} \mathit{sector}(x, y)$
for each $s_0 \in (0.1, 0.2) \iff \tau_2 = \mathit{cl}(3, \tau_3)$ and
$\mathit{lfp}(T^{\mathcal{C}}), \Delta, [x \leftarrow r, y \leftarrow s_0] \Vdash^{\mathcal{C}}_{\tau_3} \exists x_1, y_1(x \approx x_1 \wedge y \approx y_1 \wedge \mathit{circle}(x, y) \wedge$
$\mathit{parab}(x, y) \wedge x > 0.5)$
for each $s_0 \in (0.1, 0.2) \iff \tau_3 = \mathit{exists}(\mathit{exists}(\tau_4))$ and for each $s_0 \in (0.1, 0.2)$ there are $s_1, s_2 \in \mathbb{R}$ such that
$\mathit{lfp}(T^{\mathcal{C}}), \Delta, [x \leftarrow r, y \leftarrow s_0, x_1 \leftarrow s_1, y_1 \leftarrow s_2] \Vdash^{\mathcal{C}}_{\tau_4} x \approx x_1 \wedge y \approx y_1 \wedge$
$\mathit{circle}(x_1, y_1) \wedge \mathit{parab}(x_1, y_1) \wedge x_1 > 0.5$.
This can be easily simplified into:

$lfp(T^{\mathcal{C}}), \Delta, [x \leftarrow r, y \leftarrow s_0] \Vdash^{\mathcal{C}}_{\tau_4} circle(x,y) \wedge parab(x,y) \wedge x > 0.5$
for each $s_0 \in (0.1, 0.2)$.

If this process is carried through, the only suitable τ is obtained, together with the following condition obtained over r: for each $s_0 \in (0.1, 0.2)$, $r > 0.5$, $s_0^2 < r$ and $r^2 + s_0^2 < 1$. So, if
$\Theta = \{[x \leftarrow r] \mid \forall s(0.1 < s < 0.2 \Rightarrow (r > 0.5 \wedge s^2 < r \wedge r^2 + s^2 < 1))\}$, then
$lfp(T^{\mathcal{C}})(I_{\perp}{}^{\mathcal{C}}), \Delta, \Theta \Vdash^{\mathcal{C}}_{\tau} \forall y((0.1 < y \wedge y < 0.2) \Rightarrow sector(x,y))$.
In fact, Θ is the largest set of assignments for which this holds.

Remember that we are interested in having two guided forcing relations because, once a connection between them has been fixed, another connection is derived between the non guided versions. The guided version of $\Vdash$ is now defined, as in the previous cases, for a new notion of interpretation:

Definition 15. A *guided interpretation* $I_{\mathcal{T}}$ is a function $I_{\mathcal{T}} : \mathcal{W} \times \mathcal{P}(\mathcal{L}_{\mathcal{C}}) \rightarrow \mathcal{P}(At \times \mathcal{T}^{cl})$ that is monotonous, i.e. for any Δ_1, Δ_2 and Γ_1, Γ_2, if $\Delta_1 \subseteq \Delta_2$ and $\Gamma_1 \subseteq \Gamma_2$, then $I_{\mathcal{T}}(\Delta_1, \Gamma_1) \subseteq I_{\mathcal{T}}(\Delta_2, \Gamma_2)$. Let $\mathcal{I}_{\mathcal{T}}$ be the set of guided interpretations.

A partial order $\sqsubseteq$ can be defined for $\mathcal{I}_{\mathcal{T}}$, similarly to that for $\mathcal{I}$. $(\mathcal{I}_{\mathcal{T}}, \sqsubseteq)$ is a complete lattice and has an infimum, denoted $I_{\perp\mathcal{T}}$, the constant function $\emptyset$.

Definition 16. Given $I_{\mathcal{T}} \in \mathcal{I}_{\mathcal{T}}$, Δ, Γ, G and τ, the goal G *is forced* by $I_{\mathcal{T}}, \Delta$ and Γ with the guide τ, which is written $I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{\tau} G$, if τ is in accordance with $\langle \Delta, G \rangle$ and $\Vdash_{\tau}$ is the relation recursively defined depending on the structure of G, as follows:

$I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{cst} C \overset{\text{def}}{\Longleftrightarrow} \Gamma \vdash_{\mathcal{C}} C$.
$I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{cl(n,\tau)} A \overset{\text{def}}{\Longleftrightarrow} \langle A, cl(n,\tau) \rangle \in I_{\mathcal{T}}(\Delta, \Gamma)$.
$I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{and(\tau_1,\tau_2)} G_1 \wedge G_2 \overset{\text{def}}{\Longleftrightarrow} I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{\tau_i} G_i$ for each $i \in \{1, 2\}$.
$I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{or(i,\tau)} G_1 \vee G_2 \overset{\text{def}}{\Longleftrightarrow} I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{\tau} G_i$.
$I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{imp(\tau)} D \Rightarrow G \overset{\text{def}}{\Longleftrightarrow} I_{\mathcal{T}}, \Delta \cup \{D\}, \Gamma \Vdash_{\tau} G$.
$I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{impc(\tau)} C \Rightarrow G \overset{\text{def}}{\Longleftrightarrow} I_{\mathcal{T}}, \Delta, \Gamma \cup \{C\} \Vdash_{\tau} G$.
$I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{exists(\tau)} \exists x G \overset{\text{def}}{\Longleftrightarrow}$ there is a constraint C and a variable y such that:
- y does not occur free in Δ, Γ, $\exists x G$.
- $\Gamma \vdash_{\mathcal{C}} \exists y C$.
- $I_{\mathcal{T}}, \Delta, \Gamma \cup \{C\} \Vdash_{\tau} G[y/x]$.

$I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{forall(\tau)} \forall x G \overset{\text{def}}{\Longleftrightarrow}$ there is a variable y such that:
- y does not occur free in Δ, Γ, $\forall x G$.
- $I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{\tau} G[y/x]$.

Now the corresponding operator, whose least fixed point will help us to establish the equivalence between $\Vdash^{\mathcal{C}}_{\tau}$ and $\Vdash_{\tau}$, is defined.

Definition 17. The *operator* $T_{\mathcal{T}} : \mathcal{I}_{\mathcal{T}} \longrightarrow \mathcal{I}_{\mathcal{T}}$ transforms interpretations as follows. For any $I_{\mathcal{T}} \in \mathcal{I}_{\mathcal{T}}$, Δ, Γ, τ and $A \in At$, $\langle A, cl(n,\tau)\rangle \in T_{\mathcal{T}}(I_{\mathcal{T}})(\Delta,\Gamma)$ if there is a variant $\forall\overline{x}(G \Rightarrow A')$ of the n^{th} clause of Δ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $I_{\mathcal{T}}, \Delta, \Gamma \Vdash_{\tau} \exists\overline{x}(A \approx A' \wedge G)$.

The operator $T_{\mathcal{T}}$ is proved to be monotonous and continuous. The proofs are analogous to those for T (Lemmas 4 and 5). Therefore, $T_{\mathcal{T}}$ has a least fixed point, which is $\bigsqcup_{i\geq 0}(T_{\mathcal{T}})^i(I_{\perp\mathcal{T}})$, and we denote it by $lfp(T_{\mathcal{T}})$.

Below we enunciate a property of this semantics that will be useful in the proofs of several technical lemmas.

Lemma 6. *For any $I_{\mathcal{T}}, \Delta, G, \Gamma_1, \Gamma_2$ and τ, if $\Gamma_1 \vdash_{\mathcal{C}} \Gamma_2$ and $I_{\mathcal{T}}, \Delta, \Gamma_2 \Vdash_{\tau} G$ then $I_{\mathcal{T}}, \Delta, \Gamma_1 \Vdash_{\tau} G$.*

Proof. It is straightforward by induction on the structure of G.

The following lemma and corollary justify why the interpretations $\mathcal{I}_{\mathcal{T}}$ were said to be a guided version of those in $\mathcal{I}$.

Lemma 7. *Given Δ, Γ, G and $n \geq 0$, $T^n(I_{\perp}), \Delta, \Gamma \Vdash G \iff$ there exists τ such that $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau} G$.*

Proof. The proof is inductive on the order relation between pairs $\langle m, G\rangle$ defined below, where $m \geq 0$. $\langle m_1, G_1\rangle < \langle m_2, G_2\rangle$ iff *i)* $m_1 < m_2$ or *ii)* $m_1 = m_2$ and G_1 is an strict subformula of G_2 up to renaming of free variables. So, assuming the claim for every pair $\langle n', G'\rangle < \langle n, G\rangle$, it must be proved for $\langle n, G\rangle$, by case analysis on the structure of G.

$C \in \mathcal{L}_{\mathcal{C}}$

- $\Rightarrow$) $T^n(I_{\perp}), \Delta, \Gamma \Vdash C$ implies $\Gamma \vdash_{\mathcal{C}} C$. So, $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau} C$, taking $\tau = cst$.
- $\Leftarrow$) There exists τ for which $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau} C$. Due to the way in which $\Vdash_{\tau}$ was defined, such τ is bound to be *cst*, and so $\Gamma \vdash_{\mathcal{C}} C$. Therefore, $T^n(I_{\perp}), \Delta, \Gamma \Vdash C$ holds.

$A \in At$

- $\Rightarrow$) $T^n(I_{\perp}), \Delta, \Gamma \Vdash A$ implies $A \in T^n(I_{\perp})(\Delta, \Gamma)$. So, $n \geq 1$ and there is a variant $\forall\overline{x}(G' \Rightarrow A')$ of a clause of Δ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $T^{n-1}(I_{\perp}), \Delta, \Gamma \Vdash \exists\overline{x}(A \approx A' \wedge G')$. Let m be the order number of such clause in Δ. Since $\langle n-1, \exists\overline{x}(A \approx A' \wedge G')\rangle < \langle n, A\rangle$, by induction hypothesis we can assume that there is τ' such that $(T_{\mathcal{T}})^{n-1}(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau'} \exists\overline{x}(A \approx A' \wedge G')$. So, defining $\tau = cl(m, \tau')$, it is true that $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau} A$.
- $\Leftarrow$) There exists τ for which $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau} A$. Due to the way in which $\Vdash_{\tau}$ was defined, such τ must be of the form $cl(m, \tau')$, where $\forall\overline{x}(G' \Rightarrow A')$ is a variant of the m^{th} clause of Δ such that the variables $\overline{x}$ do not occur free

in Δ, Γ, A, and $(T_{\mathcal{T}})^{n-1}(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau'} \exists\overline{x}(A \approx A' \wedge G')$. By induction hypothesis, $T^{n-1}(I_{\perp}), \Delta, \Gamma \Vdash \exists\overline{x}(A \approx A' \wedge G')$. Therefore, $T^{n}(I_{\perp}), \Delta, \Gamma \Vdash A$ is satisfied.

$G_1 \wedge G_2$ $T^n(I_\perp), \Delta, \Gamma \Vdash G_1 \wedge G_2 \iff T^n(I_\perp), \Delta, \Gamma \Vdash G_i$ for each $i \in \{1,2\} \iff$ there exist τ_i' for each $i \in \{1,2\}$, for which $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau_i'} G_i$ for each $i \in \{1,2\}$, by induction hypothesis $\iff$ there exists $\tau = and(\tau_1', \tau_2')$ such that $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G_1 \wedge G_2$.

$G_1 \vee G_2$ $T^n(I_\perp), \Delta, \Gamma \Vdash G_1 \vee G_2 \iff T^n(I_\perp), \Delta, \Gamma \Vdash G_i$, for some $i \in \{1,2\} \iff$ there exist $i \in \{1,2\}$ and τ' for which $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau'} G_i$, by induction hypothesis $\iff (T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G_1 \vee G_2$, where $\tau = or(i, \tau')$.

$D \Rightarrow G'$ $T^n(I_\perp), \Delta, \Gamma \Vdash D \Rightarrow G' \iff T^n(I_\perp), \Delta \cup \{D\}, \Gamma \Vdash G' \iff$ there exists τ' for which $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta \cup \{D\}, \Gamma \Vdash_{\tau'} G'$, by induction hypothesis $\iff$ there exists $\tau = imp(\tau')$ such that $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_\tau D \Rightarrow G'$.

$C \Rightarrow G'$ $T^n(I_\perp), \Delta, \Gamma \Vdash C \Rightarrow G' \iff T^n(I_\perp), \Delta, \Gamma \cup \{C\} \Vdash G' \iff$ there exists τ' for which $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \cup \{C\} \Vdash_{\tau'} G'$, by induction hypothesis $\iff$ there exists $\tau = impc(\tau')$ such that $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_\tau C \Rightarrow G'$.

$\exists x G'$ $T^n(I_\perp), \Delta, \Gamma \Vdash \exists x G' \iff$ there is a constraint C and a variable y such that:
- y does not occur free in Δ, Γ, $\exists x G'$.
- $\Gamma \vdash_{\mathcal{C}} \exists y C$.
- $T^n(I_\perp), \Delta, \Gamma \cup \{C\} \Vdash G'[y/x]$.

$\iff$ there exists τ' for which
- y does not occur free in Δ, Γ, $\exists x G'$.
- $\Gamma \vdash_{\mathcal{C}} \exists y C$.
- $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \cup \{C\} \Vdash_{\tau'} G'[y/x]$,

by induction hypothesis $\iff (T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_\tau \exists x G'$, for $\tau = exists(\tau')$.

$\forall x G'$ $T^n(I_\perp), \Delta, \Gamma \Vdash \forall x G' \iff$ there is a variable y such that:
- y does not occur free in Δ, Γ, $\forall x G'$.
- $T^n(I_\perp), \Delta, \Gamma \Vdash G'[y/x]$.

$\iff$ there exists τ' for which
- y does not occur free in Δ, Γ, $\forall x G'$.
- $(T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau'} G'[y/x]$,

by induction hypothesis $\iff (T_{\mathcal{T}})^n(I_{\perp\mathcal{T}}), \Delta, \Gamma \Vdash_\tau \forall x G'$, for $\tau = forall(\tau')$.
□

Example 10. In the Example 1, ${T_{\mathcal{T}}}^2(I_{\perp\mathcal{T}}), \Delta, \{C\} \Vdash_\tau G$, where
$\tau = imp(forall(impc(cl(2, ex(and(cst, cl(1, ex(cst))))))))$.

Lemma 8. *Let $\{I_{\mathcal{T}i}\}_{i \geq 0}$ be a denumerable family of interpretations such that $I_{\mathcal{T}0} \sqsubseteq I_{\mathcal{T}1} \sqsubseteq I_{\mathcal{T}2} \sqsubseteq \ldots$, and let G be a goal. Then, for any Δ, Γ and τ, $\bigsqcup_{i \geq 0} I_{\mathcal{T}i}, \Delta, \Gamma \Vdash_\tau G \iff$ there exists $k \geq 0$ such that $I_{\mathcal{T}k}, \Delta, \Gamma \Vdash_\tau G$.*

Proof. Analogous to that for Lemma 3. □

Theorem 4. *Given Δ, Γ and G, $\mathit{lfp}(T), \Delta, \Gamma \Vdash G \iff$ there exists τ such that $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G$.*

Proof. $\mathit{lfp}(T), \Delta, \Gamma \Vdash G \iff$ there is $k > 0$ such that $T^k(I_\bot), \Delta, \Gamma \Vdash G$, by Lemma 3 $\iff$ there are $k > 0$ and τ such that ${T_{\mathcal{T}}}^k(I_{\mathcal{T}\bot}), \Delta, \Gamma \Vdash_\tau G$, by Lemma 7 $\iff$ there is τ such that $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G$, by Lemma 8.

Connecting the forcing relations Our next task is to establish the connection between the guided semantics, and finally the non guided ones.

The following proposition states one of the implications of the particular equivalence between $\Vdash_\tau$ and $\Vdash^{\mathcal{C}}_\tau$.

Proposition 1. *Given Δ, Γ, G, τ,*

$$\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G \Longrightarrow \mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \Vdash^{\mathcal{C}}_\tau G.$$

Proof. Let $V = \mathit{free}(\Delta \cup \{G\})$ and $\nu \in [\![\Gamma]\!]_V$. We prove $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_\tau G$ by induction on the structure of τ:

- $\tau \equiv \mathit{cst}$ and $G \equiv C$. $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\mathit{cst}} C \iff \Gamma \vdash_{\mathcal{C}} C \iff [\![\Gamma]\!]_{\mathit{free}(C)} \subseteq [\![C]\!]$, because $\mathcal{A}_{\mathcal{C}}$ agrees with $\mathcal{C}$. Then, since $[\![\Gamma]\!]_V \subseteq [\![\Gamma]\!]_{\mathit{free}(C)}$, $\nu \in [\![C]\!]$ holds, and hence $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{\mathit{cst}} C$.
- $\tau \equiv \mathit{cl}(n, \tau')$ and $G \equiv A$. $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\mathit{cl}(n,\tau')} A \iff A \in \mathit{lfp}(T_{\mathcal{T}})(\Delta, \Gamma)$ $\iff$ there is variant $\forall\overline{x}(G' \Rightarrow A')$ of the n^{th} clause in Δ such that the variables $\overline{x}$ do not occur free in Δ, Γ, A, and $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau'} \exists\overline{x}(A \approx A' \wedge G')$. $\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{\exists\overline{x}(A\approx A'\wedge G')\})} \Vdash^{\mathcal{C}}_{\tau'} \exists\overline{x}(A \approx A' \wedge G')$, by induction hypothesis. $[\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{A\})} \subseteq [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{\exists\overline{x}(A\approx A'\wedge G')\})}$, because $\mathit{free}(\Delta \cup \{\exists\overline{x}(A \approx A' \wedge G')\}) \supseteq \mathit{free}(\Delta \cup \{A\})$, so $\nu \in [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{\exists\overline{x}(A\approx A'\wedge G')\})}$ and $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{\tau'} \exists\overline{x}(A \approx A' \wedge G')$. Now, using the definition of $\Vdash^{\mathcal{C}}_\tau$ we obtain that $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_\tau A$.
- $\tau \equiv \mathit{and}(\tau_1, \tau_2)$ and $G \equiv G_1 \wedge G_2$. $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G \iff$ for each $i \in \{1, 2\}$, $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau_i} G_i$. Then, by induction hypothesis, for each $i \in \{1, 2\}$, $\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G_i\})} \Vdash^{\mathcal{C}}_{\tau_i} G_i$. In addition $\mathit{free}(\Delta \cup \{G_i\}) \subseteq \mathit{free}(\Delta\cup\{G\})$ so $[\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \subseteq [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G_i\})}$, then $\nu \in [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G_i\})}$, for each $i \in \{1, 2\}$. Therefore $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{\tau_i} G_i$ for each $i \in \{1, 2\}$ and hence $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_\tau G$.
- $\tau \equiv \mathit{or}(i, \tau')$ and $G \equiv G_1 \vee G_2$. Similar to the previous case.
- $\tau \equiv \mathit{imp}(\tau')$ and $G \equiv D \Rightarrow G'$. $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G \iff \mathit{lfp}(T_{\mathcal{T}}), \Delta \cup \{D\}, \Gamma \Vdash_{\tau'} G'$. Then $\mathit{lfp}(T^{\mathcal{C}}), \Delta \cup \{D\}, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{D,G'\})} \Vdash^{\mathcal{C}}_{\tau'} G'$, due to the induction hypothesis. Since $\mathit{free}(\Delta \cup \{D, G'\}) = \mathit{free}(\Delta \cup \{G\})$, it follows that $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_\tau G$.
- $\tau \equiv \mathit{impc}(\tau')$ and $G \equiv C' \Rightarrow G'$. $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\mathit{impc}(\tau')} C' \Rightarrow G' \iff \mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \cup \{C'\} \Vdash_{\tau'} G'$. Then, the induction hypothesis can be applied, obtaining $\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma \cup \{C'\}]\!]_{\mathit{free}(\Delta\cup\{G'\})} \Vdash^{\mathcal{C}}_{\tau'} G'$. Notice that $\mathit{free}(C') \subseteq \mathit{dom}(\nu)$. There are two cases:

i) $\nu \in [\![C']\!]$. Then $\nu \in [\![\Gamma \cup \{C'\}]\!]_{\mathit{free}(\Delta \cup \{G'\})}$, so $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{\tau'} G'$ and so $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{\tau} G$.

ii) $\nu \notin [\![C']\!]$. Then, it is immediately true that $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{impc(\tau')} G$.

- $\tau \equiv exists(\tau')$ and $G \equiv \exists x G'$. $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{exists(\tau')} \exists x G' \iff$ for any variable y not free in Δ, Γ nor G, there is a constraint C such that $\Gamma \vdash_{\mathcal{C}} \exists y C$ and $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \cup \{C\} \Vdash_{\tau'} G'[y/x]$[6]. By induction hypothesis, $\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma \cup \{C\}]\!]_{\mathit{free}(\Delta \cup \{G'[y/x]\})} \Vdash^{\mathcal{C}}_{\tau'} G'[y/x] (\dagger)$. Now we use the fact that, since $\vdash_{\mathcal{C}}$ agrees with $\mathcal{A}_{\mathcal{C}}$, $\Gamma \vdash_{\mathcal{C}} \exists y C$ and $\nu \in [\![\Gamma]\!]_{\mathit{free}(\Delta \cup \{G\})}$ imply $\nu \in [\![\exists y C]\!]$. As a consequence, there is an extension ν' of ν to y that makes C true, so $\nu' \in [\![\Gamma \cup \{G\}]\!]_{\mathit{free}(\Delta \cup \{G\}) \cup \{y\}}$ and, thanks to $(\dagger)$ and the fact $\mathit{free}(\Delta \cup \{G'[y/x]\}) = \mathit{free}(\Delta \cup \{G\}) \cup \{y\}$, $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu' \Vdash^{\mathcal{C}}_{\tau'} G'[y/x]$, hence $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{\tau} \exists x G'$.
- $\tau \equiv forall(\tau')$ and $G \equiv \forall x G'$. $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{forall(\tau')} \forall x G' \iff$ for any variable y not free in Δ, Γ nor G, $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau'} G'[y/x]$. Then, by induction hypothesis, $\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta \cup \{G'[y/x]\})} \Vdash^{\mathcal{C}}_{\tau'} G'[y/x]$. Any ν' extension of ν to y belongs to $[\![\Gamma]\!]_{\mathit{free}(\Delta \cup \{G'[y/x]\})}$, hence $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu' \Vdash^{\mathcal{C}}_{\tau'} G[y/x]$ for any such ν'. Thus $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{forall(\tau')} \forall x G$. $\square$

In order to prove the remaining implication, some particular constraints $c(\Delta, \tau, G)$ are introduced. c is defined as a partial function such that for any Δ, G and τ, $c(\Delta, \tau, G)$ is only defined if τ is in an accordance with $\langle \Delta, G \rangle$. In addition, if $\mathit{lfp}(T^{\mathcal{C}}), \Delta, \nu \Vdash^{\mathcal{C}}_{\tau} G$ for some ν, then $c(\Delta, \tau, G)$ is defined, ν satisfies it, and it is the weakest constraint that, together with Δ, forces G guided by τ, when there is any.

Definition 18. The partial function $c : \mathcal{W} \times \mathcal{T} \times \mathcal{G} \to \mathcal{L}_{\mathcal{C}}$ is recursively defined by the rules below.

- $c(\Delta, cst, C) = C$.
- $c(\Delta, cl(n, \tau'), A) = c(\Delta, \tau', \exists \overline{x}(A \approx A' \wedge G'))$, where $\forall \overline{x}(G' \Rightarrow A')$ is a variant of the n^{th} clause of Δ and for each $x \in \overline{x}$, $x \notin \mathit{free}(\Delta \cup \{A\})$.
- $c(\Delta, \mathit{and}(\tau_1, \tau_2), G_1 \wedge G_2) = c(\Delta, \tau_1, G_1) \wedge c(\Delta, \tau_2, G_2)$.
- $c(\Delta, or(i, \tau'), G_1 \vee G_2) = c(\Delta, \tau', G_i)$.
- $c(\Delta, imp(\tau'), D' \Rightarrow G') = c(\Delta \cup \{D'\}, \tau', G')$.
- $c(\Delta, impc(\tau'), C' \Rightarrow G') = C' \Rightarrow c(\Delta, \tau', G')$.
- $c(\Delta, exists(\tau'), \exists x G') = \exists y\, c(\Delta, \tau', G'[y/x])$ where $y \notin \mathit{free}(\Delta \cup \{\exists x G'\})$.
- $c(\Delta, forall(\tau'), \forall x G') = \forall y\, c(\Delta, \tau', G'[y/x])$ where $y \notin \mathit{free}(\Delta \cup \{\forall x G'\})$.

Such partial function is well defined, in the sense that given any Δ, τ and G, if $c(\Delta, \tau, G)$ is defined it is unique up to renaming of bounded variables.

Lemma 9. *Given Δ, G and τ,*

[6] Without loss of generality, we may assume that $\mathit{free}(\exists y C) \subseteq \mathit{free}(\Delta \cup \{G\})$.

$$c(\mathit{\Delta},\tau,G) \textit{ is defined} \iff \tau \textit{ is in accordance with } \langle\mathit{\Delta},G\rangle.$$

In that case, $\mathit{free}(c(\mathit{\Delta},\tau,G)) = \mathit{free}(\mathit{\Delta}\cup\{G\})$.

Proof. Straightforward by induction on the structure of τ. $\square$

Example 11. Let $\mathit{\Delta},\tau$ and G be those in Example 9. Then $c(\mathit{\Delta},\tau,G)$ is defined and $[\![c(\mathit{\Delta},\tau,G)]\!] = [\![\forall s(0.1 < s \wedge s < 0.2 \Rightarrow (s * s < x \wedge x * x + s * s < 1))]\!]$.

The following lemmas correspond to the technicalities we have announced in order to prove that, in the sense of Proposition 1, $\Vdash^{\mathcal{C}}_{\tau}$ implies $\Vdash_{\tau}$.

The lemma below states the essential property of the constraint $c(\mathit{\Delta},\tau,G)$ w.r.t. the semantics $\Vdash^{\mathcal{C}}_{\tau}$.

Lemma 10. *Given* $\mathit{\Delta},G,\tau$ *and* ν *such that* τ *is in accordance with* $\langle\mathit{\Delta},G\rangle$ *and* $\mathit{free}(\mathit{\Delta}\cup\{G\}) \subseteq \mathit{dom}(\nu)$,

$$\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau} G \iff \nu \in [\![c(\mathit{\Delta},\tau,G)]\!].$$

Proof. We analyze cases according to the structure of τ. Applying Lemma 9, we may assume, in any case, that $c(\mathit{\Delta},\tau,G)$ is defined because τ is in accordance with $\langle\mathit{\Delta},G\rangle$ by hypothesis.

- $\tau \equiv \mathit{cst}$, $G \equiv C$. In this case $c(\mathit{\Delta},\tau,G) = C$ and $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\mathit{cst}} C \iff \nu \in [\![C]\!] = [\![c(\mathit{\Delta},\tau,G)]\!]$.
- $\tau \equiv \mathit{and}(\tau_1,\tau_2)$, $G \equiv G_1 \wedge G_2$. $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau} G \iff \mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau_i} G_i$ for each $i \in \{1,2\} \iff \nu \in [\![c(\mathit{\Delta},\tau_i,G_i)]\!]$ for $i \in \{1,2\}$, by induction hypothesis $\iff \nu \in [\![c(\mathit{\Delta},\tau,G)]\!]$.
- $\tau \equiv \mathit{or}(i,\tau')$, $G \equiv G_1 \vee G_2$. $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau} G \iff \mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau'} G_i \iff \nu \in [\![c(\mathit{\Delta},\tau',G_i)]\!]$, by induction hypothesis $\iff \nu \in [\![c(\mathit{\Delta},\tau,G)]\!]$.
- $\tau \equiv \mathit{imp}(\tau')$, $G \equiv D \Rightarrow G'$. $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau} G \iff \mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta}\cup\{D\},\nu \Vdash^{\mathcal{C}}_{\tau'} G' \iff \nu \in [\![c(\mathit{\Delta}\cup\{D\},\tau',G')]\!]$, by induction hypothesis $\iff \nu \in [\![c(\mathit{\Delta},\tau,G)]\!]$.
- $\tau \equiv \mathit{impc}(\tau')$, $G \equiv C \Rightarrow G'$. $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau} C \Rightarrow G' \iff \nu \notin [\![C]\!]$ or $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau'} G' \iff \nu \notin [\![C]\!]$ or $\nu \in [\![c(\mathit{\Delta},\tau',G')]\!]$, by induction hypothesis $\iff \nu \in [\![C \Rightarrow c(\mathit{\Delta},\tau',G')]\!] = [\![c(\mathit{\Delta},\tau,G)]\!]$.
- $\tau \equiv \mathit{exists}(\tau')$, $G \equiv \exists x G'$. $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\tau} G \iff$ there is an extension ν' of ν to y such that $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu' \Vdash^{\mathcal{C}}_{\tau'} G'[y/x] \iff$ there is an extension ν' of ν to y such that $\nu' \in [\![c(\mathit{\Delta},\tau',G'[y/x])]\!]$, by induction hypothesis $\iff$ there is $a \in AC$ such that $[\![c(\mathit{\Delta},\tau',G'[y/x])]\!]_{\nu[y\leftarrow a]} = \mathit{true}$ $\iff [\![\exists y\, c(\mathit{\Delta},\tau',G'[y/x])]\!]_{\nu} = \mathit{true} \iff \nu \in [\![\exists y\, c(\mathit{\Delta},\tau',G'[y/x])]\!] = [\![c(\mathit{\Delta},\tau,G)]\!]$.
- $\tau \equiv \mathit{forall}(\tau')$, $G \equiv \forall x G'$. $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu \Vdash^{\mathcal{C}}_{\mathit{forall}(\tau')} \forall x G' \iff$ given a variable $y \notin \mathit{dom}(\nu)$, for any extension ν' of ν to the variable y, $\mathit{lfp}(T^{\mathcal{C}}),\mathit{\Delta},\nu' \Vdash^{\mathcal{C}}_{\tau'} G'[y/x]$ holds $\iff \nu' \in [\![c(\mathit{\Delta},\tau',G'[y/x])]\!]$ for any extension ν' of ν, by induction hypothesis $\iff$ for any $a \in AC$, $[\![c(\mathit{\Delta},\tau',G'[y/x])]\!]_{\nu[y\leftarrow a]} = \mathit{true} \iff [\![\forall y\, c(\mathit{\Delta},\tau',G'[y/x])]\!]_{\nu} = \mathit{true} \iff \nu \in [\![\forall y\, c(\mathit{\Delta},\tau',G'[y/x])]\!] = [\![c(\mathit{\Delta},\tau,G)]\!]$. $\square$

Now we establish the connection between $c(\Delta, \tau, G)$ and the semantics $\Vdash_\tau$.

Lemma 11. *Given Δ, Γ, G and τ,*

$$lfp(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G \iff \Gamma \vdash_{\mathcal{C}} c(\Delta, \tau, G).$$

Proof. Both implications are proved below.

$\Rightarrow$) $lfp(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_\tau G$ implies that $c(\Delta, \tau, G)$ is defined, applying Lemma 9, and in virtue of Proposition 1, $lfp(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{free(\Delta \cup \{G\})} \Vdash_\tau G$. Then from Lemma 10, $[\![\Gamma]\!]_{free(\Delta \cup \{G\})} \subseteq [\![c(\Delta, \tau, G)]\!]$, but again from Lemma 9, $free(\Delta \cup \{G\}) = free(c(\Delta, \tau, G))$. So $[\![\Gamma]\!]_{free(c(\Delta,\tau,G))} \subseteq [\![c(\Delta, \tau, G)]\!]$ which is equivalent to $\Gamma \vdash_{\mathcal{C}} c(\Delta, \tau, G)$.

$\Leftarrow$) First of all, let us prove that $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_\tau G$, by induction of the structure of τ. In any case, it is verified that τ is in accordance with $\langle \Delta, G \rangle$, because we are assuming that $c(\Delta, \tau, G)$ is defined and Lemma 9.

- $\tau \equiv cst$ and $G \equiv C$. Then $c(\Delta, \tau, G) = C$, and $lfp(T_{\mathcal{T}}), \Delta, C \Vdash_\tau C$ holds by definition of $\Vdash_\tau$.
- $\tau \equiv cl(n, \tau')$ and $G \equiv A$. Let $\forall \overline{x}(G' \Rightarrow A')$ be a variant of the n^{th} clause of Δ such that no $x \in \overline{x}$ is free in $\Delta, c(\Delta, cl(n, \tau'), A)$ nor A. In that case, $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, cl(n, \tau'), A) \Vdash_\tau A$ happens if and only if $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, cl(n, \tau'), A) \Vdash_{\tau'} \exists \overline{x}(A \approx A' \wedge G')$, which is true by induction hypothesis, because $c(\Delta, cl(n, \tau'), A) \equiv c(\Delta, \tau', \exists \overline{x}(A \approx A' \wedge G'))$.
- $\tau \equiv and(\tau_1, \tau_2)$ and $G \equiv G_1 \wedge G_2$. Now $c(\Delta, \tau, G) = c(\Delta, \tau_1, G_1) \wedge c(\Delta, \tau_2, G_2)$. By induction hypothesis, $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau_i, G_i) \Vdash_{\tau_i} G_i$ for each $i \in \{1, 2\}$. Since $c(\Delta, \tau, G) \vdash_{\mathcal{C}} c(\Delta, \tau_i, G_i)$ for each $i \in \{1, 2\}$, thanks to Lemma 6 it follows that $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_{\tau_i} G_i$ for $i = 1, 2$, and hence $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_\tau G$.
- $\tau \equiv or(i, \tau')$ and $G \equiv G_1 \vee G_2$. Applying the induction hypothesis, $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau', G_i) \Vdash_{\tau'} G_i$, so $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_{\tau'} G_i$, because $c(\Delta, \tau, G) = c(\Delta, \tau', G_i)$, therefore $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_\tau G$.
- $\tau \equiv imp(\tau')$ and $G \equiv D \Rightarrow G'$. $lfp(T_{\mathcal{T}}), \Delta \cup \{D\}, c(\Delta \cup \{D\}, \tau', G') \Vdash_{\tau'} G'$, by the induction hypothesis. Then $lfp(T_{\mathcal{T}}), \Delta \cup \{D\}, c(\Delta, \tau, G) \Vdash_{\tau'} G'$, since $c(\Delta, \tau, G) = c(\Delta \cup \{D\}, \tau', G')$, hence $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_\tau D \Rightarrow G'$.
- $\tau \equiv impc(\tau')$ and $G \equiv C \Rightarrow G'$. $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau', G') \Vdash_{\tau'} G'$, by the induction hypothesis. Therefore, $lfp(T_{\mathcal{T}}), \Delta, \{C, C \Rightarrow c(\Delta, \tau', G')\} \Vdash_{\tau'} G'$, by Lemma 6. Then $lfp(T_{\mathcal{T}}), \Delta, \{C, c(\Delta, \tau, G)\} \Vdash_{\tau'} G'$, since $c(\Delta, \tau, G) = C \Rightarrow c(\Delta, \tau', G')$. Hence $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_\tau C \Rightarrow G'$, by definition of $\Vdash_\tau$.
- $\tau \equiv exists(\tau')$ and $G \equiv \exists x G'$. $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, exists(\tau'), \exists x G') \Vdash_\tau \exists x G'$ $\iff$ there is a constraint C' and a variable y, not free in $\Delta, c(\Delta, \tau, G)$ nor G, such that:
  i) $c(\Delta, exists(\tau'), \exists x G') \vdash_{\mathcal{C}} \exists y C'$,
  ii) $lfp(T_{\mathcal{T}}), \Delta, c(\Delta, exists(\tau'), \exists x G') \cup \{C'\} \Vdash_{\tau'} G'[y/x]$.

Let us choose C' to be $c(\Delta, \tau', G'[y/x])$. Since $c(\Delta, exists(\tau'), \exists x G') \equiv \exists y\ c(\Delta, \tau', G'[y/x])$, *i)* boils down to the evidence $\exists y C' \vdash_{\mathcal{C}} \exists y C'$. On the other hand, by induction hypothesis, $\mathit{lfp}(T_{\mathcal{T}}), \Delta, C' \Vdash_{\tau'} G'[y/x]$, and thanks to Lemma 6 this implies *ii)*.

- $\tau \equiv forall(\tau')$ and $G \equiv \forall x G'$. Let $y \notin \mathit{free}(\Delta \cup \{G\})$. Applying the induction hypothesis, $\mathit{lfp}(T_{\mathcal{T}}), \Delta, c(\Delta, \tau', G'[y/x]) \Vdash_{\tau'} G'[y/x]$. By definition $c(\Delta, \tau, G) \equiv \forall y\, c(\Delta, \tau', G'[y/x])$, so it is clear that $c(\Delta, \tau, G) \vdash_{\mathcal{C}} c(\Delta, \tau', G'[y/x])$. Hence it follows $\mathit{lfp}(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_{\tau'} G'[y/x]$, in virtue of Lemma 6. Since $y \notin \mathit{free}(c(\Delta, \tau, G))$ and $y \notin \mathit{free}(\Delta \cup \{G\})$, $\mathit{lfp}(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_{\tau} G$.

Therefore, $\mathit{lfp}(T_{\mathcal{T}}), \Delta, c(\Delta, \tau, G) \Vdash_{\tau} G$ is satisfied for any Δ, G and τ. Then $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau} G$, thanks to Lemma 6 and $\Gamma \vdash_{\mathcal{C}} c(\Delta, \tau, G)$. □

Finally, we are ready to prove the counterpart of Proposition 1.

Proposition 2. *Given Δ, Γ, G and τ,*

$$\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \Vdash^{\mathcal{C}}_{\tau} G \Longrightarrow \mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau} G.$$

Proof. $\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \Vdash^{\mathcal{C}}_{\tau} G$ implies $[\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \subseteq [\![c(\Delta, \tau, G)]\!]$, thanks to Lemma 10. But $\mathit{free}(\Delta \cup \{G\}) = \mathit{free}(c(\Delta, \tau, G))$, by Lemma 9. Then $[\![\Gamma]\!]_{\mathit{free}(c(\Delta,\tau,G))} \subseteq [\![c(\Delta, \tau, G)]\!])$. Thus $\Gamma \vdash_{\mathcal{C}} c(\Delta, \tau, G)$, because $\mathcal{A}_{\mathcal{C}}$ and $\vdash_{\mathcal{C}}$ agree, and finally, from Lemma 11, $\mathit{lfp}(T), \Delta, \Gamma \Vdash_{\tau} G$ is obtained. □

The main theorem below, which establishes the relation between the non guided semantics, is a consequence of the previous results.

Theorem 5. *For any Δ, Γ and G,*

$$\mathit{lfp}(T), \Delta, \Gamma \Vdash G \iff \mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \Vdash^{\mathcal{C}} G.$$

Proof. $\mathit{lfp}(T), \Delta, \Gamma \Vdash G \iff$ there exists τ such that $\mathit{lfp}(T_{\mathcal{T}}), \Delta, \Gamma \Vdash_{\tau} G$ by Theorem 4 $\iff$ exists τ such that $\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \Vdash^{\mathcal{C}}_{\tau} G$ by Propositions 1 and 2 $\iff$ $\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \Vdash^{\mathcal{C}} G$ by definition of the forcing relation $\Vdash^{\mathcal{C}}$. □

Corollary 1. *(Soundness and completeness) For any Δ, Γ and G,*

$$\mathit{lfp}(T^{\mathcal{C}}), \Delta, [\![\Gamma]\!]_{\mathit{free}(\Delta\cup\{G\})} \Vdash^{\mathcal{C}} G \iff \Delta; \Gamma \vdash_{\mathcal{UC}} G.$$

Proof. The claim is a simple combination of Theorems 2, 3 and 5. □

5 Conclusions

In previous papers [11,10] 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 [14]. 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 [6,7] we defined semantics for *HH*($\mathcal{C}$) based on fixed point constructions as is usually done in the *LP* and *CLP* fields [12,1,2,9,4].

As far as we know, our works have been the first attempts 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 [13] 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 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 and constraints in goals. The universal quantifier is also handled in [3], but the presence of universal constraints involves further difficulties that we have solved.

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 [20]. 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 [9] fixed point semantics constitutes 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.

Formalizing the two fixed point semantics we have introduced for *HH*($\mathcal{C}$) requires a lot of intermediate technical definitions and results, mainly for the second one for which, assignments for some structure in accordance with the constraint system are incorporated to the forcing relation to interpret constraints. In this paper we have included every lemma and definition used to prove the foundations of the two semantics and their connection between they and with the proof system.

References

1. 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.
2. M. Comini, G. Levi, and M. C. Meo. A theory of observables for logic programs. *Information and Computation*, 69(1–2):23–80, 2001.
3. M. de Marco. *Intuitionistic Semantics for Hereditarily Harrop Logic Programming.* PhD thesis, Wesleyan University, 1999.
4. M. Gabbrielli, G. M. Dore, and G. Levi. Observable semantics for constraint logic programs. *Journal of Logic and Computation*, 5(2):133–171, 1995.
5. M. Garcia-Diaz and S. Nieva. Solving constraints for an instance of an extended CLP language over a domain based on real numbers and herbrand terms. *Journal of Functional and Logic Programming*, 2003(2), September 2003.
6. M. García-Díaz and S. Nieva. A fixed point semantics for an extended CLP language In 13th International Workshop on Functional and (Constraint) Logic Programming (WFLP'04) Herbert Kuchen editor. Technical Report AIB-2004-05 pages 118–136, RWTH Aachen, jun 2004.
7. M. García-Díaz and S. Nieva. Providing declarative semantics for HH extended constraint logic programs. In *Proc. Sixth ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP04)*, pages 55–66. ACM Press, 2004.
8. J. Jaffar and M. J. Maher. Constraint logic programming: a survey. *Journal of Logic Programming*, 19/20:503–581, 1994.
9. 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.
10. 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.
11. 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.
12. J. W. Lloyd. *Foundations of logic programming.* Springer-Verlag, 1987.
13. D. Miller. A logical analysis of modules in logic programming. *Journal of Logic Programming*, 6(1-2):79–108, 1989.
14. 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.
15. G. Nadathur. A proof procedure for the logic of hereditary Harrop formulas. *Journal of Automated Reasoning*, 11:111–145, 1993.
16. G. Smolka and R. Treinen. Records for logic programming. *Journal of Logic Programming*, 18(3):229–258, 1994.
17. A. Tarski. *A decision method for elementary algebra and geometry.* University of California Press, 1951.
18. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.
19. 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.
20. D. A. Wolfram. A semantics for λ-prolog. *Theoretical Computer Science*, 136:277–288, 1994.