Logic Programming in Tau Categories *

Stacy E. Finkelstein
McGill University
sef@triples.math.mcgill.ca

Peter Freyd
University of Pennsylvania
pjf@saul.cis.upenn.edu

James Lipton
Wesleyan University
lipton@allegory.cs.wesleyan.edu

Abstract

Many features of current logic programming languages are not captured by conventional semantics. Their fundamentally non-ground character, and the uniform way in which such languages have been extended to typed domains, subject to constraints, suggest that a categorical treatment of constraint domains, of programming *syntax* and of semantics may be closer in spirit to what declarative programming is really about, than conventional set-theoretic semantics.

We generalize the notion of a (many-sorted) logic program and of a resolution proof, by defining them both over a (not necessarily free) τ-category $\mathcal{C}$, a category with products enriched with a mechanism for canonically manipulating n-ary relations [8]. Computing over this domain includes computing over the Herbrand Universe, and over equationally presented constraint domains as special cases. We give a categorical treatment of the fix-point semantics of Kowalski and van Emden, which establishes completeness in a very general setting.

1 Introduction

Tau Categories are categories enriched with a mechanism for canonically manipulating n-ary relations. This τ-structure creates, through canonical limits, an ideal framework for formalizing an abstract syntax for logic programming with constraints. By defining the notions of resolution and unification over such a category, one is able to capture a quite general notion of constraint logic programming and establish a completeness theorem, based on a non-ground categorical generalization of fixed point semantics.

In previous work connecting category theory and logic programming, interpretation of logic in categories has followed a general pattern described by M. Makkai and G. Reyes [15]. The work by A. Corradini and U. Montanari [5] focused on the abstract computations of logic programs and the view of logic programs as structured transition systems. The work by A. Corradini and A. Asperti [4] builds on [5] and defines a model for a logic program as a family of monoidal categories indexed by sets of variables. In both of these works, the category-theoretic setting is semantical and is not used to describe details of the resolution process. The work of Asperti and Martini [2] involves a translation of some of the syntactic mechanisms of resolution along with the interpretation of logic programs in cartesian categories based on the concept of using

*From: *Computer Science Logic '94*, Leszek Pacholski and Jerzy Tiuryn, eds., Lecture Notes in Computer Science 933, Springer, 1995.

projections as variables, and was a starting point for our work. The categorical framework can be further exploited to cover more general notions of programming syntax and resolution as well as a cleaner semantics that exploits and strengthens the fixed-point approach of Kowalski and van Emden [13].

Categorical interpretations are often limited by the fact that the main constructs such as products and pullbacks are only defined up to isomorphism, forcing arbitrary choices of representatives of a given isomorphism class. τ-categories, introduced by Peter Freyd in [7] and further explored in *Categories, Allegories* [8] are a uniform setting in which to work with n-ary relations and thus describe logic programs. They are finite limit categories enriched with τ-structure, a distinguished class of diagrams which provides canonical choices for limits, and guarantees associative equality and a strict unit for the product.

We begin by describing a categorical syntax for logic programs in an arbitrary τ-category as in [6].

2 τ-categories

We begin with an outline of the theory of τ-categories, described in detail in [8]. We depart slightly from the cited reference in that in this paper, τ-categories need only have all finite products, not all finite limits. The τ-structure guarantees that when finite limits do exist they are canonical. Also note that consideration of this class of categories is not overly restrictive since every small finite limit category is equivalent to a τ-category([8]).

The relevant definitions (of table, short column, compositions and τ-categories) are given in table 1.

An isomorphism class of tables is called a *relation*. any table is equivalent to a monic composed with a product table. In fact, this is a strong equivalence with regard to the τ-structure, as a τ-monic composed with a product τ-table is *exactly* its equivalent τ-table.

Axiom 1 of the definition of a τ-category says that the class τ is a set of representatives for each isomorphism class of tables. It is this property which allows for a canonical choice of limits. We say that the product diagram $\langle A \times B; p_1, p_2 \rangle$ is canonical if it is a τ-table. Similarly, a pullback diagram is canonical if $\langle P; p_1, p_2 \rangle \in \tau$ (where $p_1 : P \to A$ and $p_2 : P \to B$ are the pullback arrows of $f : A \to C$ and $g : B \to C$ respectively.) These axioms also imply that the category has a unique terminal object.

Canonical limits in τ-categories enjoy several important properties ([8]). Canonical products are strictly associative. The terminator is strictly a two-sided unit. Also, if each of two horizontally adjacent pullback squares is a canonical pullback then so is the outer pullback rectangle they describe.

3 Logical Structure and Interpretation

The information contained in this section describes a τ-categorical syntax for logic programming and is given in greater detail in [6].

The basic translation of the first-order language $\mathcal{L}$ into the τ-category follows the usual interpretation as in [15], slightly augmented with the canonical choices allowed by the τ-structure. The relevant definitions are given in table 2. We first define an interpretation of terms, inductively, as arrows in $\mathbf{A}^\tau$, relative to a sequence of distinct variables containing all of the free variables in the term. We then extend this interpretation to predicates.

Table 1: Tables and τ-categories

Definition 2.1 *A* **table** *is an object T with a monic finite sequence of morphisms $x_1, \ldots, x_n$ with T as a common source. We will denote such a table by $\langle T; x_1, \ldots, x_n \rangle$.*

Definition 2.2 *Given a table $\langle T; x_1, \ldots, x_n \rangle$ we say that x_j is a* **short column** *it does not contribute to the monic character of the table, in other words if for every $f, g : X \to T$ such that $fx_j \neq gx_j$ there exists $i < j$ such that $fx_i \neq gx_i$.*

If x_j is a short column, then $\langle T; x_1, \ldots, \hat{x}_j, \ldots, x_n \rangle$ will denote the table obtained by removing the arrow x_j from the sequence $x_1, \ldots, x_n$.

Definition 2.3 *Given tables $\langle T; x_1, \ldots, x_m \rangle$ and $\langle T'; y_1, \ldots, y_n \rangle$ where $x_j : T \to T'$, we define their* **composition at j** *as $\langle T; x_1, \ldots, x_m \rangle \circ_j \langle T'; y_1, \ldots, y_n \rangle = \langle T; x_1, \ldots, x_{j-1}, x_j y_1, \ldots, x_j y_n, x_{j+1}, \ldots, x_m \rangle$.*

Definition 2.4 *A* **τ-category** *is a category with all finite limits and a distinguished class of tables, denoted τ, such that*

1. *Every table is isomorphic to a unique table in τ.*
2. *$\langle T; 1_T \rangle \in \tau$, all T.*
3. *If $\langle T; x_1, \ldots, x_m \rangle \in \tau$ and $\langle T'; y_1, \ldots, y_n \rangle \in \tau$ and $x_j : T \to T'$, then*
$$\langle T; x_1, \ldots, x_m \rangle \circ_j \langle T'; y_1, \ldots, y_n \rangle \in \tau$$
4. *If $\langle T; x_1, \ldots, x_n \rangle \in \tau$ and x_j is a short column, then $\langle T; x_1, \ldots, \hat{x}_j, \ldots, x_n \rangle \in \tau$*

In the table, we will let $\mathcal{L}$ be a first order language, $\vec{x} = (x_1, \ldots, x_m)$ be a sequence of distinct variables of sorts $\rho_1, \rho_2, \ldots, \rho_m$ and let $M(\vec{x})$ denote the canonical product $M(\rho_1) \times \cdots \times M(\rho_m)$ in $\mathbf{A}^\tau$.

It is important to note in the definition of $M_{\vec{x}}(Pt_1 \cdots t_m)$ that not only is the arrow p a monic arrow, by a property of pullbacks, but it is also a τ-monic arrow since the second leg of this canonical pullback is short.

3.1 Substitution and Unification in $\mathbf{A}^\tau$

Let $\theta = \{x_1 := t_1, \ldots, x_n := t_n\}$ be a substitution with the x_i's all distinct. In addition, let $\vec{y} = (y_1, \ldots, y_m)$ be a sequence of distinct variables containing all of the free variables of the terms t_i.

Then we may define a categorical substitution $\Theta_{\vec{y}}$ in $\mathbf{A}^\tau$ to be the morphism:

$$M(\vec{y}) \xrightarrow{\langle M_{\vec{y}}(t_1), \ldots, M_{\vec{y}}(t_n) \rangle} M(\vec{x})$$

One may prove that for any term s, $M_{\vec{y}}(s\theta) = M_{\vec{x}}(s) \circ \Theta_{\vec{y}}$ and thus that a categorical substitution may be applied to a term in this structure via ordinary composition of arrows in $\mathbf{A}^\tau$.

Definition 3.4 *Assume θ and $\vec{y}$ are as described above. Given an atomic formula φ, we may define the application of a substitution to φ to be the arrow q which is the canonical pullback of $M_{\vec{x}}(\varphi) \rightarrowtail M(\vec{x})$ along the arrow $\Theta_{\vec{y}}$.*

For nonatomic formulae $\varphi := \wedge_{i=1}^m A_i$, one may extend the definition for atomic formulae by taking canonical pullbacks along each τ-monic of φ.

Table 2: Term structure in τ-categories

Terms
Definition 3.1 *For any τ-category $\mathbf{A}^\tau$, an $\mathbf{A}^\tau$-structure M on the language $\mathcal{L}$ is a function $M : \mathcal{L} \to \mathbf{A}^\tau$ such that for every sort σ in $\mathcal{L}$, $M(\sigma)$ is an object of $\mathbf{A}^\tau$, and for every operation symbol f in $\mathcal{L}$ of arity n, input sorts $\sigma_1, \sigma_2, \ldots, \sigma_n$, and of sort ρ, $M(f) : M(\vec{\sigma}) \to M(\rho)$ is a morphism in $\mathbf{A}^\tau$ where $M(\vec{\sigma})$ denotes the canonical product $M(\sigma_1) \times \cdots \times M(\sigma_n)$. (Constants will be thought of as functions of arity 0, and so for each constant c of sort ρ, $M(c) : \mathbf{1} \to M(\rho)$ is a morphism in $\mathbf{A}^\tau$.)*
Definition 3.2 *For a term t of sort ρ, having all of its free variables among $\vec{x}$, $M_{\vec{x}}(t)$ is defined to be a morphism $M(\vec{x}) \to M(\rho)$ in $\mathbf{A}^\tau$ as follows:*
t:= x$_i$ *$M_{\vec{x}}(x_i)$ is defined to be the canonical projection $M(\vec{x}) \to M(\rho_i)$*
t:= c *For a constant c of sort ρ, $M_{\vec{x}}(c)$ is defined as the composition:* $$M(\vec{x}) \xrightarrow{!_{M_{\vec{x}}}} \mathbf{1} \xrightarrow{M(c)} M(\rho)$$
t:= ft$_1 \cdots$ **t**$_m$ *If each t_i is of sort σ_i, then $M_{\vec{x}}(ft_1 \cdots t_m)$ is defined as the following composition:* $$M(\vec{x}) \xrightarrow{\langle M_{\vec{x}}(t_1), \ldots, M_{\vec{x}}(t_m)\rangle} M(\vec{\sigma}) \xrightarrow{M(f)} M(\rho)$$
Predicates
Definition 3.3 *For every predicate symbol R in $\mathcal{L}$ with arity n and sorts $\sigma_1, \sigma_2, \ldots, \sigma_n$, define $M(R)$ to be a τ-monic $M(R) \to M(\vec{\sigma})$. For a formula φ with all its free variables among $\vec{x}$, $M_{\vec{x}}(\varphi)$ is defined as follows:*
φ**:= Pt**$_1 \cdots$ **t**$_m$ *If each t_i is of sort σ_i, then $M_{\vec{x}}(Pt_1 \cdots t_m)$ is defined to be the τ-arrow p which is the canonical pullback of $M(P) \rightarrowtail M(\vec{\sigma})$ along the arrow $\langle M_{\vec{x}}(t_1), \ldots, M_{\vec{x}}(t_m)\rangle$.*
φ**:=** $\wedge_{i=1}^m A_i$ *If each A_i is an atom which has been interpreted as described above, then $M_{\vec{x}}(\varphi)$ is defined to be the m-tuple of τ-monic arrows $M_{\vec{x}}(A_i) \rightarrowtail M(\vec{x})$.*

As in the definition of $M_{\vec{x}}(Pt_1 \cdots t_m)$, q will automatically be a τ-monic arrow in $\mathbf{A}^\tau$. Additionally, one may show for $\varphi := Pu_1 \cdots u_k$, $M_{\vec{y}}(\varphi\theta) = M_{\vec{y}}(P(u_1\theta) \cdots (u_k\theta))$.

In a logic programming language, the unifiability of formulae depends on the equality of the predicate symbols and the existence of a *unifier*, a substitution which unifies the terms appearing in the formulae. In a logic programming τ-category, the unifiability of formulae depends on the equality of the interpretation of the predicate symbol and the existence of a pullback for the terms. It is this ability to define formulae in terms of different lists of variables and then use a pullback to unify them that removes the need for any renaming of variables to avoid clashes.

Definition 3.5 *Let $\varphi := Pt_1 \cdots t_n$ and $\psi := Rs_1 \cdots s_m$. φ and ψ are said to be* **unifiable** *if the τ-monic arrows $M(P) \to M(\vec{\sigma})$ and $M(R) \to M(\vec{\rho})$ are equal and there exists a pair of substitution arrows $\langle \Theta_{\vec{y}}, \Phi_{\vec{y}}\rangle$ forming a commutative square, i.e., such that $\langle M_{\vec{z}}(s_1), \ldots, M_{\vec{z}}(s_n)\rangle \circ \Phi_{\vec{y}} = \langle M_{\vec{x}}(t_1), \ldots, M_{\vec{x}}(t_n)\rangle \circ \Theta_{\vec{y}}$*

In this case the pair $\langle \Theta_{\vec{y}}, \Phi_{\vec{y}}\rangle$ are said to unify the formulae φ and ψ. Note that the first condition implies in particular that n=m, $M(\vec{\sigma}) = M(\vec{\rho})$, and $M(P) = M(R)$. Also note that the occur check is internal to this setting. Since all interpretations are made with respect to a list of variables and, in particular, the substitution arrows are made with respect to an ostensibly different list of variables than the terms to be unified, there can be no substitution of a variable with a term containing that variable unless a separate diagram specifies these to be equal.

In Horn clause logic programming, if unifiers of formulae exist, there are most general unifiers. The corresponding notion in a τ-category is that of canonical pullbacks.

Definition 3.6 *Unifiers* $\langle \Theta_{\vec{y}}, \Phi_{\vec{y}} \rangle$ *are* **most general unifiers** *(mgu's) iff the commutative square described in definition 3.5 is a pullback.*

In many constraint domains (e.g. the set of closed terms in the lambda calculus), terms may be unifiable without having mgu's. But if unifiers exist in a category, it is straightforward to add the appropriate pullback squares and form a new τ-category in which mgu's exist. This is discussed in the last section of the paper.

Lemma 3.7 *The two conditions in Definition 3.5 imply that the arrows* $M_{\vec{y}}(\varphi\theta) \to M(\vec{y})$ *and* $M_{\vec{y}}(\psi\phi) \to M(\vec{y})$ *are equal when* $\langle \Theta_{\vec{y}}, \Phi_{\vec{y}} \rangle$ *is taken to be most general.*

3.2 Resolution

Definition 3.8 *Let C be the definite clause $A_1, A_2, \ldots, A_n \vdash B$, (for $n \geq 0$). Then a* **definite clause diagram**, *$M_{\vec{z}}(C)$, is the following (n+1)-tuple of τ-monic arrows:*

$$\begin{tikzcd}
M_{\vec{z}}(A_1) \arrow[dr, tail] & \cdots & M_{\vec{z}}(A_n) \arrow[dl, tail] & \\
& M(\vec{z}) & & M_{\vec{z}}(B) \arrow[ll, tail]
\end{tikzcd}$$

Definition 3.9 *A* **definite τ-logic program** *P^{τ} is a finite set of definite clause diagrams $M_{\vec{z}}(C)$.*

Definition 3.10 *Let G be the definite goal $B_1 \wedge B_2 \wedge \ldots \wedge B_m$, (for $m \geq 1$). Then a* **definite goal diagram**, *$M_{\vec{x}}(G)$ will be an m-tuple of τ-monic arrows $M_{\vec{x}}(B_i) \rightarrowtail M(\vec{x})$ to the base $M(\vec{x})$.*

Definition 3.11 *Let $M_{\vec{x}}(G)$ be a definite goal diagram, and $M_{\vec{z}}(C)$ be a definite clause diagram. Then a new definite goal diagram $M_{\vec{y}}(G')$ is* **derived** *from $M_{\vec{x}}(G)$ and $M_{\vec{z}}(C)$ using $\langle \Theta_{\vec{y}}, \Phi_{\vec{y}} \rangle$ if the following conditions hold:*

1. *$M_{\vec{x}}(B_i) \to M(\vec{x})$ is the* **selected** *τ-monic arrow of $M_{\vec{x}}(G)$.*
2. *$\langle \Theta_{\vec{y}}, \Phi_{\vec{y}} \rangle$ is an mgu of $M_{\vec{x}}(B_i) \to M(\vec{x})$ and $M_{\vec{z}}(B) \to M(\vec{z})$.*
3. *$M_{\vec{y}}(G')$ is the following definite goal diagram:*

$$\begin{tikzcd}
M_{\vec{y}}(B_1\theta) \arrow[drrr, tail] & \cdots M_{\vec{y}}(B_{i-1}\theta) \arrow[drr, tail] & M_{\vec{y}}(A_1\phi) \arrow[dr, tail] & \cdots & M_{\vec{y}}(A_n\phi) \arrow[dl, tail] & M_{\vec{y}}(B_{i+1}\theta) \arrow[dll, tail] & \cdots M_{\vec{y}}(B_m\theta) \arrow[dlll, tail] \\
& & & M(\vec{y}) & & &
\end{tikzcd}$$

where recall $M_{\vec{y}}(B_j\theta) \rightarrowtail M(\vec{y})$ is defined by a substitution pullback of $M_{\vec{x}}(B_j) \rightarrowtail M(\vec{x})$ against $\Theta_{\vec{y}}$.

Definition 3.12 *Let $\mathbf{A}^\tau$ be a τ-category, P^τ a definite τ-logic program and $M_{\vec{x}}(G)$ a definite goal diagram. A* τ**SLD-derivation** *of $P^\tau \vdash_{A^\tau} M_{\vec{x}}(G)$ consists of a sequence (finite or infinite) $M_{\vec{x}}(G)$, $M_{\vec{x}'}(G_1)$, $M_{\vec{x}''}(G_2)$, $\ldots$ of definite goal diagrams, a sequence $M_{\vec{z}}(C_1)$, $M_{\vec{z}'}(C_2)$, $\ldots$ of definite program clauses of P^τ and a sequence $\langle \Theta^1, \Phi^1 \rangle$, $\langle \Theta^2, \Phi^2 \rangle$, $\ldots$ of most general unifiers such that each $M_{\vec{a}}(G_{i+1})$ is derived from $M_{\vec{b}}(G_i)$ and $M_{\vec{c}}(C_{i+1})$ using $\langle \Theta^{i+1}_{\vec{a}}, \Phi^{i+1}_{\vec{a}} \rangle$. A* τ**SLD-proof** *of $\{M_{\vec{x}}(G)\}$ from P^τ is a finite τSLD-derivation of $P^\tau \vdash_{A^\tau} M_{\vec{x}}(G)$ which has the empty diagram as the last definite goal diagram in the derivation.*

Definition 3.13 *Let P^τ be a definite τ-logic program and let $M_{\vec{x}}(G)$ be a definite goal diagram. A* τ**-computed answer** *for $P^\tau \vdash_{A^\tau} M_{\vec{x}}(G)$ is the substitution arrow $\Theta = \Theta^1 \circ \Theta^2 \circ \cdots \circ \Theta^r$ where $\langle \Theta^1, \Phi^1 \rangle$, $\ldots$, $\langle \Theta^r, \Phi^r \rangle$ is the sequence of most general unifiers used in a τSLD - proof of $\{M_{\vec{x}}(G)\}$ from P^τ.*

3.3 Models

Definition 3.14 *An* **interpretation** *for a logic programming structure $\mathbf{A}^\tau$ is a cartesian category $\mathcal{D}$ with an associated cartesian functor $H : \mathbf{A}^\tau \to \mathcal{D}$.*

In the following section we shall only be considering semantic categories $\mathcal{D}$ which have the syntactic category as a subcategory. In this case, the functor H will be the obvious inclusion functor which preserves the logical structure.

Definition 3.15 *The notion of* **validity** *in $\mathcal{D}$ (under H) for a formula φ is defined as follows:*

1. *A conjunction $M_{\vec{x}}(\varphi)$ is valid in $\mathcal{D}$ iff $H\,[l]$, where l: $Lim \to M(\vec{x})$ is the canonical limit of the diagram $M_{\vec{x}}(\varphi)$, is an isomorphism.*
2. *A definite clause diagram $M_{\vec{z}}(A_1, A_2, \ldots, A_n \vdash B)$ is valid in $\mathcal{D}$ (under H) iff there exists a monic arrow m such that:*

$$\begin{tikzcd} H\,[Lim] \arrow[rr, tail, "m"] \arrow[dr, "H\,[l]"'] & & H\,[M_{\vec{x}}(B)] \arrow[dl] \\ & H\,[M(\vec{x})] & \end{tikzcd}$$

   where l is the limit of the diagram $M_{\vec{x}}(\psi)$ for $\psi = A_1 \wedge A_2 \wedge \ldots \wedge A_n$.

3. *A logic program P^τ is valid in $\mathcal{D}$ iff every clause of P^τ is valid in $\mathcal{D}$. In this case we say that $\mathcal{D}$ (under H) is a* **model** *of P^τ.*

Notice that, for atomic formulas, validity in a model just means it is interpreted as an isomorphism. Also, this definition of validity does not distinguish between ground and non-ground terms (the former being terms with source **1**).

Theorem 3.16 (Soundness of τSLD-Resolution) *Let P^τ be a definite τ-logic program and $M_{\vec{x}}(G)$ a definite goal diagram. If $\Theta_{\vec{y}}$ is a τ-computed answer for $P^\tau \vdash_{A^\tau} M_{\vec{x}}(G)$, then it is the case that $M_{\vec{y}}(G\theta)$ is valid in all models of P^τ.*

4 Completeness of τSLD-Resolution

Let $\mathcal{L}$ be the language of the Horn-clause program P and $\mathcal{L}_{\mathcal{R}} = \{R_1, \ldots, R_n\}$ the set of relation symbols in the language, of sorts $\{\sigma_1, \ldots, \sigma_n\}$. In order to establish completeness we begin with a category $\mathcal{C}$ with terminator and products, in which all sorts, constants and function symbols of $\mathcal{L}$ have been interpreted by a $\mathcal{C}$-structure M. On this base *category of constraints* we create a syntactic category into which we translate logic programs and over which SLD resolution will take place. The syntactic category, (previously denoted $\mathbf{A}^{\tau}$in preceding sections) is obtained by *freely adjoining indeterminate monics* $X_i \rightarrowtail B$ to $\mathcal{C}$, so as to associate to each predicate in $\mathcal{L}_{\mathcal{R}}$ an indeterminate monic, through which no arrows from $\mathcal{C}$ representing terms will factor.

In general, for any category $\mathcal{C}$ with terminator and products, and any object B in $\mathcal{C}$, the category $\mathcal{C}_B$ (also called $\mathcal{C}[X_i]$ below) obtained by adjoining an indeterminate subobject $X_i \rightarrowtail B$ is defined as follows. The objects of $\mathcal{C}_B$ are pairs $\langle A, s\rangle$ where A is an object of $\mathcal{C}$ and s is a finite set of morphisms $A \to B$ of $\mathcal{C}$. We will let the morphisms $\langle A, s\rangle \to \langle A', s'\rangle$ in $\mathcal{C}_B$ be those morphisms $f : A \to A'$ of $\mathcal{C}$ such that for every morphism $g : A' \to B$ in s' it is the case that $f;g : A \to B$ is in s.

$\mathcal{C}_B$ has a terminal object, $\langle \mathbf{1}, \emptyset\rangle$, and products for every pair of objects $\langle A, s\rangle$ and $\langle A', s'\rangle$ given by $\langle A \times A', \pi_1 s \cup \pi_2 s'\rangle$, where $A \times A'$ is a product in $\mathcal{C}$ with projections π_1 and π_2, and $\pi_1 s \cup \pi_2 s'$ is the union of the sets $\pi_1 s = \{\pi_1 f : f \in s\}$ and $\pi_2 s' = \{\pi_1 f : f \in s'\}$. In addition, the embedding $\mathcal{C} \overset{\iota}{\rightarrowtail} \mathcal{C}_B$ which sends $f : A \to A'$ in $\mathcal{C}$ to $f : \langle A, \emptyset\rangle \to \langle A', \emptyset\rangle$ in $\mathcal{C}_B$ is full and faithful and preserves existing limits. We may also consider an underlying functor $\mathcal{U} : \mathcal{C}_B \to \mathcal{C}$ which sends $f : \langle A, s\rangle \to \langle A', s'\rangle$ in $\mathcal{C}_B$ to its underlying arrow $f : A \to A'$. It is through this functor that $\mathcal{C}_B$ inherits a τ-structure from $\mathcal{C}$. We may define $\tau_{\mathcal{D}_B}$ by letting a table be in $\tau_{\mathcal{D}_B}$ iff its image under $\mathcal{U}$ is in $\tau_{\mathcal{D}}$. It is straightforward to verify that this definition of $\tau_{\mathcal{D}_B}$ satisfies the axioms of a τ-category (definition 2.4).

The category $\mathcal{C}_B$ also has a generic subobject $id_{\mathcal{C}} : \langle B, \{id\}\rangle \to \langle B, \emptyset\rangle$. This is our indeterminate monic $X \rightarrowtail \iota(B)$, and it is important to note that all pullbacks of this monic along arrows t from $\mathcal{C}$ (so-called "term arrows") exist in $\mathcal{C}_B$ and are of the form $\langle A, \{t\}\rangle \overset{id_{\mathcal{C}}}{\rightarrowtail} \langle A, \emptyset\rangle$ which cannot be an isomorphism in $\mathcal{C}_B$.

To emphasize the fact that we are interpreting predicate letters as indeterminate monics, we use the notation $\mathcal{C}[X] \equiv \mathcal{C}\,[X_1, \ldots, X_n]$ for the category obtained from $\mathcal{C}$ by freely adjoining indeterminate monics $X_i \rightarrowtail M(\sigma_i)$ one for each $R_i \in \mathcal{L}_{\mathcal{R}}$. We extend M to an interpretation (also called M) of the predicates in $\mathcal{L}_{\mathcal{R}}$, by defining $M(R_i) = X_i$ (that is to say, $M(R_i)$ is the monic $X_i \rightarrowtail M(\sigma_i)$). Such a category, along with such a structure map, corresponds to the category $\mathbf{A}^{\tau}$and structure map (also denoted M) over which the proof theory has been discussed in previous sections. We shall call the category $\mathcal{C}[X]$ with its structure map a *logic programming category* (an LP-category). We now use the notation $P^{\tau}\vdash_{\mathcal{C}} G$ to mean there is a τSLD-proof of G from P^{τ} in the category $\mathcal{C}[X]$.

Our ambient semantic category will be the topos $\widehat{\mathcal{C}}$ given by the contravariant Yoneda embedding from $\mathcal{C}$ (ie: $\mathcal{C} \rightarrowtail Set^{\mathcal{C}^{op}}$.) In $\widehat{\mathcal{C}}$ we are able to take unions and images. Hence, in particular, any predicate defined by resolution over $\mathcal{C}$ is representable as a monic in $\widehat{\mathcal{C}}$. We will use this framework to define initial models for logic programs over $\mathcal{C}$, as subcategories of $\widehat{\mathcal{C}}$.

We will need to make use of a few elementary facts about the Yoneda embedding whose proofs can be found in e.g. [10, 8]. The embedding shown above is full and faithful. The objects B in $\mathcal{C}$ appear in $\widehat{\mathcal{C}}$ as representable functors $Hom(\_, B)$, although we will often abuse notation and refer to them by their original names B. All representable functors are coprime and projective. Thus, in particular, arrows from representables into unions of functors in $\widehat{\mathcal{C}}$ must factor through one of them. Also, up to isomorphism, we may take the subobject relation $A \subset B$ between subobjects A, B of some object C to be pointwise inclusion of functors. That is to say, given a chain of subobjects of b in $\widehat{\mathcal{C}}$ such as the diagram on the left,

$$\begin{tikzcd} \cdots & E_i \arrow[rr, tail] \arrow[dr, "g_1^*"'] & & E_{i+1} \arrow[dl, "g_r^*"] & \cdots \\ & & b & & \end{tikzcd} \qquad \begin{tikzcd} \cdots & F_i \arrow[rr, "\subseteq"] \arrow[dr, "\subseteq"'] & & F_{i+1} \arrow[dl, "\subseteq"] & \cdots \\ & & b & & \end{tikzcd}$$

there are isomorphic subobjects $F_i \simeq E_i$ such that the diagram on the right commutes, with all arrows the canonical inclusions $\xrightarrow{\subseteq}$ induced by pointwise set-theoretic containment of functors: for each object X in $\mathcal{C}$, $F_i(X) \subseteq F_{i+1}(X)$.

To show completeness, we shall develop a category-theoretic generalization of the Kowalski-Van Emden T_P-operator ([13]). We will define an operator $T_{P,\mathcal{C}}$ on semantic categories $\mathcal{D}$ which builds a new category from $\mathcal{D}$ generated by the clauses of the program. By a process of iteration we construct a fixed point for $T_{P,\mathcal{C}}$, an initial model for the program from which we may show completeness.

Throughout, we will denote by $(f)^{\#} : Sub(a) \longrightarrow Sub(b)$ the functor from the lattice of sub-objects of a in $\widehat{\mathcal{C}}$ to that of b induced by pulling back along $f : b \to a$, and by $\Im(f)$ its left adjoint in $\widehat{\mathcal{C}}$. Notice that $\Im(id)(g)$ is the image of g in $\widehat{\mathcal{C}}$.

Definition 4.1 *A* **semantic category** *is a full subcategory $\mathcal{D}$ of $\widehat{\mathcal{C}}$ which contains $\mathcal{C}$ and which is equipped with an extended structure (denoted $[\![\,]\!]^{\mathcal{D}}$) based on the structure already defined on $\mathcal{C}$ (previously denoted M). $[\![\,]\!]^{\mathcal{D}}$ will interpret sorts, constants and function symbols as prescribed by the $\mathcal{C}$ -structure M, and will assign to each predicate symbol a monic arrow of $\mathcal{D}$.*

Note that one may show using a straightforward induction on terms that for any term t, $[\![t]\!]^{\mathcal{D}}_{\vec{x}} = [\![t]\!]^{\mathcal{C}}_{\vec{x}}$. Thus we will often drop the superscripted categories from our notation when discussing interpretations of terms.

Given a program P let us denote by cl a clause $A_1, A_2, \ldots, A_n \vdash B(t)$. We will then mean by hd_{cl} the predicate symbol B (the *head* of cl), by tm_{cl} the term t (associated with hd_{cl}), and by tl_{cl} the list $A_1, \ldots, A_n$ (the *tail* of cl). We then make the following categorical notational definitions.

Definition 4.2 *Let P^{τ} be a definite τ-logic program in a semantic category $\mathcal{D}$ with extended structure $[\![\,]\!]^{\mathcal{D}}$. Then denote by $[\![cl]\!]^{\mathcal{D}}$ the definite clause diagram associated with $A_1, A_2, \ldots, A_n \vdash B(t)$. We will define $[\![hd_{cl}]\!]^{\mathcal{D}}$ to be $[\![B]\!]^{\mathcal{D}}$, $[\![tm_{cl}]\!]^{\mathcal{D}}_{\vec{x}}$ to be $[\![t]\!]^{\mathcal{D}}_{\vec{x}}$, and $[\![tl_{cl}]\!]^{\mathcal{D}}_{\vec{x}}$ to be the monic from the limit of the (definite goal) diagram consisting of the fan of arrows $[\![A_i]\!]^{\mathcal{D}}_{\vec{x}} \rightarrowtail [\![\vec{x}]\!]^{\mathcal{D}}$ from the clause diagram above.*

Definition 4.3 *If $[\![t]\!]_{\vec{x}} : M(\vec{x}) \to M(\vec{\sigma})$ and $[\![r]\!]_{\vec{y}} : M(\vec{y}) \to M(\vec{\sigma})$ are terms, then $[\![t]\!]_{\vec{x}}$ is said to* **factor through** *$[\![r]\!]_{\vec{y}}$ in $\mathcal{D}$ if there exists an arrow θ in $\mathcal{D}$ such that the arrows id and θ are pullbacks of the arrows $[\![t]\!]_{\vec{x}}$ and $[\![r]\!]_{\vec{y}}$ respectively.*

We remind the reader that the Yoneda embedding reflects factorizations, since it is full: if $[\![t]\!]_{\vec{x}}$ factors through $[\![r]\!]_{\vec{y}}$ in $\widehat{\mathcal{C}}$, then it factors through in $\mathcal{C}$.

In favt we have the following, which will be useful in proving completeness of resolution.

Proposition 4.4 *Let $C_1 \xrightarrow{r} C_2$ and $C_0 \xrightarrow{t} C_2$ be arrows in $\mathcal{C}$. If $\llbracket t \rrbracket$ factors through $\Im(\llbracket r \rrbracket)(F)$ for some subobject F of $\llbracket C_1 \rrbracket$ in $\widehat{\mathcal{C}}$, then t is an instance of r in $\mathcal{C}$ and $\llbracket t \rrbracket$ factors through $F\llbracket r \rrbracket$. In particular, there is an arrow $C_0 \xrightarrow{\varphi} C_1$ such that $t = \varphi r$ and $\llbracket \varphi \rrbracket$ factors through F.*

Proof. Consider the diagram

$$\begin{tikzcd}
 & & \llbracket C_0 \rrbracket \arrow[dll, dotted, "\varrho"'] \arrow[dl, "u"'] \arrow[d, "\llbracket t \rrbracket"] \\
\bar{F} \arrow[r, "e"] \arrow[dr, "F"'] & \overline{\Im(\llbracket r \rrbracket)(F)} \arrow[r, tail, "\Im(\llbracket r \rrbracket)(F)"] & \llbracket C_2 \rrbracket \\
 & \llbracket C_1 \rrbracket \arrow[ur, "\llbracket r \rrbracket"'] &
\end{tikzcd}$$

where $e; \Im(\llbracket r \rrbracket)(F)$ is an epi-monic image factorization of $F\llbracket r \rrbracket$ and each $\llbracket C_i \rrbracket$ is a representable functor in $\widehat{\mathcal{C}}$, and therefore projective. The projectivity of $\llbracket C_0 \rrbracket$ implies that the arrow u (which exists by the hypothesis that $\llbracket t \rrbracket$ factors through $\Im(\llbracket r \rrbracket)(F)$) lifts to an arrow ϱ making the diagram commute. Let $f = \varrho F$. The source and target of f are representable, so by the fullness of the Yoneda embedding, $f = \llbracket \varphi \rrbracket$ for some substitution φ, and $\llbracket \varphi \rrbracket \llbracket r \rrbracket = \llbracket t \rrbracket$. By faithfulness of the Yoneda embedding, $\varphi r = t$. $\Box$

Van Emden and Kowalski introduced an elegant semantics for Prolog in terms of fixed points of a continuous operator T_P on the power set of the Herbrand base B_P of a prolog program P [13]. The action of T_p on a subset S of B_P is given by:

$$\begin{aligned} T_P(S) \quad = \quad & S \cup \{Q(t') : A'_1, \ldots, A'_n \vdash Q(t') \text{ is a ground} \\ & \qquad \text{instance of a clause in } P \ \& \ A'_1, \ldots, A'_n \in S\}. \end{aligned}$$

The least term model of P is precisely the least fixed point of T_P which we can think of as building the model as a (co-)limit of a chain. If we fix a predicate letter Q, we can think of each clause $A_i, \ldots, A_n \vdash Q(t)$ in P as contributing to the extent of Q all ground instances of t corresponding to terms in the interpretation of the A_i. Because of the non-ground nature of categorical semantics we can define analogous operators E and $T_{P,\mathcal{C}}$ over our more general categorical models in many respects simpler their set-theoretic counterpart. The operator E adjoins *finitely* many arrows to the arrow interpreting the predicate Q in a category. The fact that this is done over an arbitrary category $\mathcal{C}$ with terminator and certain limits poses no difficulties. The main idea is as follows. We assume the predicate letter B has the interpretation $\llbracket B \rrbracket^{\mathcal{D}}$ in some category $\mathcal{D}$. Then if $A_1, \ldots, A_n \vdash B(t)$ is a clause in P, let $\llbracket A_1, \ldots, A_n \rrbracket^{\mathcal{D}}_{\vec{x}}$ be the limit of the clause diagram for $A_1, \ldots, A_n$ in $\mathcal{D}$. Then we define E by

$$E(\llbracket\,\rrbracket^{\mathcal{D}}, B) = \llbracket B \rrbracket^{\mathcal{D}} \cup \Im(\llbracket t \rrbracket^{\mathcal{D}}_{\vec{x}})(\llbracket A_1, \ldots, A_n \rrbracket^{\mathcal{D}}_{\vec{x}}) \cup \cdots$$

where the remaining terms of the union, not shown, correspond to any remaining clauses in P whose head predicate letter is B, and where, we recall, $\Im(\llbracket t \rrbracket^{\mathcal{D}}_{\vec{x}})$ is the image functor along t. We now give this definition in its full generality.

Definition 4.5 *Let $\mathcal{D}$ be a semantic category as defined in definition 4.1 with extended structure $[\![\,]\!]^{\mathcal{D}}$. Then for each predicate letter $B \in \mathcal{L}_{\mathcal{R}}$ define the following*

$$E([\![\,]\!]^{\mathcal{D}}, B) = [\![B]\!]^{\mathcal{D}} \cup \left[\bigcup_{\substack{cl \in P \\ hd_{cl} = B}} \{\Im([\![tm_{cl}]\!]^{\mathcal{D}}_{\vec{x}})([\![tl_{cl}]\!]^{\mathcal{D}}_{\vec{x}})\} \right]$$

For each term s in $\mathcal{C}$, let

$$E([\![\,]\!]^{\mathcal{D}}, B, s) = ([\![s]\!]^{\mathcal{D}}_{\vec{y}})^{\#}(E([\![\,]\!]^{\mathcal{D}}, B))$$

Note that these arrows are in $\widehat{\mathcal{C}}$, but not, in general, in $\mathcal{D}$.

Now let $T_{P,\mathcal{C}}(\mathcal{D})$ be the full subcategory of $\widehat{\mathcal{C}}$ generated by $\mathcal{D}$, $\mathbf{0} \in \widehat{\mathcal{C}}$, $E([\![\,]\!]^{\mathcal{D}}, B)$ for each predicate letter $B \in \mathcal{L}_{\mathcal{R}}$, and closed under finite limits. $T_{P,\mathcal{C}}(\mathcal{D})$ is then also a semantic category with extended structure defined by $[\![B]\!]^{T_{P,\mathcal{C}}(\mathcal{D})} = E([\![\,]\!]^{\mathcal{D}}, B)$ for predicate symbols $B \in \mathcal{L}_{\mathcal{R}}$.

In particular, let $\mathcal{C}_0$ be the full subcategory of $\widehat{\mathcal{C}}$ obtained by adjoining to the original category $\mathcal{C}$ the co-terminator $\mathbf{0}$ of $\widehat{\mathcal{C}}$ and defining an extended interpretation $[\![\,]\!]^0$ given by $[\![B]\!]^0$ = the unique map $0 \to [\![b]\!]^0$ for predicate symbols $B \in \mathcal{L}_{\mathcal{R}}$, where b = sort of B. We may then define a sequence of categories $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n, \ldots$ built by letting $\mathcal{C}_{i+1} = T_{P,\mathcal{C}}(\mathcal{C}_i)$, each with the extended interpretation $[\![B]\!]^n$ given in the definition of $T_{P,\mathcal{C}}(\mathcal{C}_i)$.

Construct for each predicate symbol $B \in \mathcal{L}_{\mathcal{R}}$ the filtered colimit in $\widehat{\mathcal{C}}$ of the following diagram of inclusions

$$[\![B]\!]^0 \rightarrowtail [\![B]\!]^1 \rightarrowtail \cdots \rightarrowtail [\![B]\!]^k \rightarrowtail \cdots$$

Lemma 4.6 $\varinjlim\{[\![B]\!]^n\} \cong \bigcup_n \{[\![B]\!]^n\}$

This follows easily from the facts about the Yoneda embedding at the beginning of the section. In particular, we may take the preceding diagram of inclusions to be pointwise containment of functors F_n. The pointwise union of this diagram is easily seen to be its colimit.

Definition 4.7 *We define $\mathcal{C}^*$ as the full subcategory of $\widehat{\mathcal{C}}$ generated by the categories $\mathcal{C}_k$, $\varinjlim\{[\![B]\!]^n\}$ for each predicate symbol $B \in \mathcal{L}_{\mathcal{R}}$, and closed under finite limits. For this category we then define an extended structure by $[\![B]\!]^* = \varinjlim\{[\![B]\!]^n\}$ for each predicate symbol $B \in \mathcal{L}_{\mathcal{R}}$.*

Lemma 4.8 $[\![tl_{cl}]\!]^*_{\vec{x}} = \varinjlim\{[\![tl_{cl}]\!]^k_{\vec{x}}\}$

Proof. Consider the clause diagram $[\![cl]\!]^k$ in $\mathcal{C}_k$ as defined in definition 4.2. Let $[\![A_i]\!]^*_{\vec{x}}$ be given by $[\![A_i]\!]^*_{\vec{x}} = ([\![u]\!]^*_{\vec{x}})^{\#}[\![A'_i]\!]^*$, where $A_i = A'_i(u)$. By definition $[\![A'_i]\!]^* = \varinjlim\{[\![A'_i]\!]^k\}$. Hence, $[\![A_i]\!]^*_{\vec{x}} = ([\![u]\!]^*_{\vec{x}})^{\#}(\varinjlim\{[\![A'_i]\!]^k\})$. So since $(-)^{\#}$ is also a left adjoint (in addition to being a right adjoint), we have $[\![A_i]\!]^*_{\vec{x}} = \varinjlim\{([\![u]\!]^*_{\vec{x}})^{\#}[\![A'_i]\!]^k\}_k$. But then, $[\![A_i]\!]^*_{\vec{x}} = \varinjlim\{([\![u]\!]^k_{\vec{x}})^{\#}[\![A'_i]\!]^k\}_k = \varinjlim\{[\![A_i]\!]^k\}_k$. And so

$$\begin{aligned}
[\![tl_{cl}]\!]^*_{\vec{x}} &= \varprojlim\{[\![A_i]\!]^*_{\vec{x}}\}_i \\
&= \varprojlim\{\varinjlim\{[\![A_i]\!]^k\}_k\}_i \\
&= \varinjlim\{\varprojlim\{[\![A_i]\!]^k\}_i\}_k \\
&= \varinjlim\{[\![tl_{cl}]\!]^k_{\vec{x}}\}_k
\end{aligned}$$

because filtered colimits commute with finite limits in a topos (eg [3]). □

Lemma 4.9 *The category $\mathcal{C}^*$ with interpretation $H : \mathcal{C}[X] \longrightarrow \mathcal{C}^*$ induced by evaluation $H(X_i) = [\![B]\!]^*$ is a model of the program P.*

Proof. We show that $T_{P,\mathcal{C}}(\mathcal{C}^*) = \mathcal{C}^*$, and, in particular, that E preserves colimits of subobject chains, ie.

$$[\![B]\!]^* = [\![B]\!]^* \cup \left[\bigcup_{\substack{cl \in P \\ hd_{cl} = B}} \{\Im([\![tm_{cl}]\!]_{\vec{x}})([\![tl_{cl}]\!]^*_{\vec{x}})\} \right] \tag{1}$$

for each predicate letter $B \in \mathcal{L}_{\mathcal{R}}$. It will suffice to show that for each clause such that $hd_{cl} = B$, $\Im([\![tm_{cl}]\!]_{\vec{x}})([\![tl_{cl}]\!]^*_{\vec{x}}) \subseteq [\![B]\!]^*$. Then since $\Im([\![tm_{cl}]\!]_{\vec{x}})$ is the left adjoint of $([\![tm_{cl}]\!]_{\vec{x}})^{\#}$, it suffices to show $[\![tl_{cl}]\!]^*_{\vec{x}} \subseteq ([\![tm_{cl}]\!]_{\vec{x}})^{\#} [\![B]\!]^*$. We know by the definition of $[\![B]\!]^{k+1}$ that $\Im([\![tm_{cl}]\!]_{\vec{x}}) [\![tl_{cl}]\!]^k_{\vec{x}} \subseteq [\![B]\!]^{k+1}$. So since $(\text{-})^{\#}$ is also a left adjoint, we have that for each k

$$\begin{aligned} [\![tl_{cl}]\!]^k_{\vec{x}} &\subseteq ([\![tm_{cl}]\!]_{\vec{x}})^{\#} [\![B]\!]^{k+1} \\ &\subseteq \varinjlim\{([\![tm_{cl}]\!]_{\vec{x}})^{\#} [\![B]\!]^k\} \\ &= ([\![tm_{cl}]\!]_{\vec{x}})^{\#} \varinjlim\{[\![B]\!]^k\} \\ &= ([\![tm_{cl}]\!]_{\vec{x}})^{\#} [\![B]\!]^* \end{aligned}$$

By lemma 4.8 we know that $[\![tl_{cl}]\!]^*_{\vec{x}} = \varinjlim\{[\![tl_{cl}]\!]^k_{\vec{x}}\}$. Thus we may conclude by the universal property of colimits that

$$[\![tl_{cl}]\!]^*_{\vec{x}} = \varinjlim\{[\![tl_{cl}]\!]^k_{\vec{x}}\} \subseteq ([\![tm_{cl}]\!]_{\vec{x}})^{\#} [\![B]\!]^* \tag{2}$$

In fact it is the case that for any semantic category $\mathcal{D}$, being a fixed point of $T_{P,\mathcal{C}}$ is equivalent to being a model of P. In the proof above, one sees that to show $\mathcal{C}^*$ a fixed point of $T_{P,\mathcal{C}}$ (equation 1 for each predicate letter B), one may equivalently show that for each clause such that $hd_{cl} = B$, $\mathcal{C}^*$ is a model of P (equation 2). □

Lemma 4.10 ($[\![\,]\!]$-monotonicity) *If for every predicate letter A in $\mathcal{L}_{\mathcal{R}}$, $[\![A]\!]^{\mathcal{D}} \subseteq [\![A]\!]^{\mathcal{D}'}$ then for any predicate letter $B \in \mathcal{L}_{\mathcal{R}}$ and any term t, $E([\![\,]\!]^{\mathcal{D}}, B, t) \subseteq E([\![\,]\!]^{\mathcal{D}'}, B, t)$.*

Proof. Since limits preserve containment, it suffices to show that for any predicate letter $B \in \mathcal{L}_{\mathcal{R}}$, $E([\![\,]\!]^{\mathcal{D}}, B) \subseteq E([\![\,]\!]^{\mathcal{D}'}, B)$. By hypothesis, $[\![B]\!]^{\mathcal{D}} \subseteq [\![B]\!]^{\mathcal{D}'}$. Also, since $[\![A_i]\!]^{\mathcal{D}}_{\vec{x}} \subseteq [\![A_i]\!]^{\mathcal{D}'}_{\vec{x}}$ for each $[\![A_i]\!]^{\mathcal{D}}_{\vec{x}}$ in $[\![tl_{cl}]\!]^{\mathcal{D}}_{\vec{x}}$ and each $[\![A_i]\!]^{\mathcal{D}'}_{\vec{x}}$ in $[\![tl_{cl}]\!]^{\mathcal{D}'}_{\vec{x}}$, we have that $[\![tl_{cl}]\!]^{\mathcal{D}}_{\vec{x}} \subseteq [\![tl_{cl}]\!]^{\mathcal{D}'}_{\vec{x}}$ for each clause $cl \in P$ such that $hd_{cl} = B$. But since in any regular category the functor $\Im(s)$, for any arrow s, preserves containment of monics, it is the case that for each clause $cl \in P$ such that $hd_{cl} = B$, $\Im([\![tm_{cl}]\!]^{\mathcal{D}}_{\vec{x}})([\![tl_{cl}]\!]^{\mathcal{D}}_{\vec{x}}) \subseteq \Im([\![tm_{cl}]\!]^{\mathcal{D}'}_{\vec{x}})([\![tl_{cl}]\!]^{\mathcal{D}'}_{\vec{x}})$. But now the required containment follows directly from elementary properties of unions. □

Lemma 4.11 *$\mathcal{C}^*$ is an initial model of P.*

Proof. We want to show that for any model $\mathcal{D}$ of P (ie any semantic category $\mathcal{D}$ which is a fixed point of $T_{P,\mathcal{C}}$), that validity in $\mathcal{C}^*$ implies validity in $\mathcal{D}$. In particular, it suffices to show that for any predicate letter $B \in \mathcal{L}_{\mathcal{R}}$, $[\![B]\!]^* \subseteq [\![B]\!]^{\mathcal{D}}$.

We first show by induction on k that for all k, $[\![B]\!]^k \subseteq [\![B]\!]^{\mathcal{D}}$. Since $[\![B]\!]^0 =$ the unique map $0 \to [\![b]\!]^0$ for $b =$ sort of B, it is clear that $[\![B]\!]^0 \subseteq [\![B]\!]^{\mathcal{D}}$. Now suppose that $[\![B]\!]^k \subseteq [\![B]\!]^{\mathcal{D}}$. By

Lemma 4.10 and the fact that $\mathcal{D}$ is a fixed point of $T_{P,\mathcal{C}}$, $[\![B]\!]^{k+1} = E([\![\,]\!]^k, B) \subseteq E([\![\,]\!]^{\mathcal{D}}, B) = [\![B]\!]^{\mathcal{D}}$.

It now follows by the definition of $[\![B]\!]^*$ and Lemma 4.6 that $[\![B]\!]^* = \varinjlim\{[\![B]\!]^n\} \cong \bigcup_n\{[\![B]\!]^n\} \subseteq [\![B]\!]^{\mathcal{D}}$. □

Definition 4.12 *A clause $A_1, A_2, \ldots, A_m \vdash B(r)$ is said to* **admit** *Q(t) if $B = Q$ and $[\![t]\!]_{\vec{x}}$ factors through $[\![r]\!]_{\vec{y}}$.*

Lemma 4.13 *If an atomic formula Q(t) is valid in all models of the program P, (i.e. is interpreted as an isomorphism) then some clause of P admits Q(t).*

Proof. Consider the following model $\mathcal{D}$ of the program P. Adjoin to $\mathcal{C}$ the arrows (in $\widehat{\mathcal{C}}$):

$$\bigcup_{\substack{cl \in P \\ hd_{cl} = Q}} Im([\![tm_{cl}]\!]_{\vec{y}})$$

for each predicate letter $Q \in P$ and close under finite limits. Let $\mathcal{D}$ be the full subcategory of $\widehat{\mathcal{C}}$ generated by this. Now let the extended structure $[\![\,]\!]^{\mathcal{D}}$ be defined by

$$[\![B]\!]^{\mathcal{D}} = \bigcup_{\substack{cl \in P \\ hd_{cl} = Q}} Im([\![tm_{cl}]\!]_{\vec{y}})$$

where by convention, if $B \notin P$ then $[\![B]\!]^{\mathcal{D}}$ (the empty union) shall be the unique arrow $\mathbf{0} \to [\![b]\!]$ for $b =$ sort of B.

The category $\mathcal{D}$ with interpretation $H : \mathcal{C}[X] \longrightarrow \mathcal{D}$ induced by evaluation $H(X_i) = [\![B]\!]^{\mathcal{D}}$ is a model of the program P since if $A_1, A_2, \ldots, A_m \vdash B(r)$ is a clause *cl* of P, then $[\![r]\!]_{\vec{y}}$ factors through $Im([\![r]\!]_{\vec{y}})$ and so through $[\![B]\!]^{\mathcal{D}}$. Thus $[\![B(r)]\!]_{\vec{y}}^{\mathcal{D}}$ is an isomorphism through which $[\![tl_{cl}]\!]_{\vec{y}}^{\mathcal{D}}$ factors automatically.

Now, since $Q(t)$ is valid in all models, $[\![Q(t)]\!]_{\vec{x}}^{\mathcal{D}}$ is an isomorphism. In other words, $[\![t]\!]_{\vec{x}}$ factors through $[\![Q]\!]^{\mathcal{D}} = \bigcup_{\substack{cl \in P \\ hd_{cl} = Q}} Im([\![tm_{cl}]\!]_{\vec{y}})$. But then, by coprimeness of representable objects of $\widehat{\mathcal{C}}$ (i.e. images of objects of $\mathcal{C}$ under the Yoneda embedding) $[\![t]\!]_{\vec{x}}$ must factor through $Im([\![r]\!]_{\vec{y}})$ for some particular clause $A_1, A_2, \ldots, A_m \vdash Q(r)$ in P. By proposition 4.4 then, $[\![t]\!]_{\vec{x}}$ must factor through $[\![r]\!]_{\vec{y}}$ and fullness of the Yoneda embedding, the factoring arrow must be in $\mathcal{C}$. Thus, this clause of P admits $Q(t)$. □

Lemma 4.14 *Let G be an atomic formula. If $[\![G]\!]_{\vec{x}}^k$ is an isomorphism, then there is a τSLD-proof of $M_{\vec{x}}^{\mathcal{C}[X]}(G)$.*

Proof. The proof will proceed by induction on k.

If $k = 0$, then the lemma holds vacuously since for all predicate letters $B \in \mathcal{L}_{\mathcal{R}}$, $[\![B]\!]^0 : \mathbf{0} \to [\![b]\!]$ where b is the sort of B, which cannot be an isomorphism. Now assume that the lemma is true for $k = n$. And suppose that $[\![G]\!]_{\vec{x}}^{n+1}$ is an isomorphism, where $G = Q(t)$ for some predicate letter Q and term t. If $[\![Q(t)]\!]^n_{\vec{x}}$ is also an isomorphism, then the lemma holds by the induction hypothesis. So assume that $[\![Q(t)]\!]^n_{\vec{x}}$ is not an isomorphism. By definition

$$[\![Q(t)]\!]^{n+1}_{\vec{x}} = E([\![\,]\!]^n, Q, t) = ([\![t]\!]_{\vec{x}})^{\#}\left([\![Q]\!]^n \cup \left[\bigcup_{\substack{cl\in P\\ hd_{cl}=Q}} \{\Im([\![tm_{cl}]\!]_{\vec{y}})([\![tl_{cl}]\!]^n_{\vec{y}})\}\right]\right)$$

Since this is an isomorphism, then $[\![t]\!]_{\vec{x}}$ factors through the union

$$\left([\![Q]\!]^n \cup \left[\bigcup_{\substack{cl\in P\\ hd_{cl}=Q}} \{\Im([\![tm_{cl}]\!]_{\vec{y}})([\![tl_{cl}]\!]^n_{\vec{y}})\}\right]\right)$$

But then, since $M(\vec{x})$ (or, more precisely, its image in $\widehat{\mathcal{C}}$) is coprime, it must be the case that $[\![t]\!]_{\vec{x}}$ factors through **one** of the members of the union. Since we have assumed that $[\![t]\!]_{\vec{x}}$ does not factor through $[\![Q]\!]^n$, it must be the case that for some clause $cl \in P$ such that $hd_{cl} = Q$ (say the clause $A_1, A_2, \ldots, A_m \vdash Q(r)$) $([\![t]\!]_{\vec{x}})^{\#}\Im([\![r]\!]_{\vec{y}})([\![tl_{cl}]\!]^n_{\vec{y}})$ is an isomorphism. In fact, by Lemmas 4.13 and 4.11 and 4.4, there is a substitution arrow $\Theta_{\vec{x}}$ in $\mathcal{C}$, corresponding to a substitution θ on terms, such that $\Theta_{\vec{x}}[\![r]\!]_{\vec{y}} = [\![t]\!]_{\vec{x}}$ and such that $\Theta_{\vec{x}}$ factors through $[\![tl_{cl}]\!]^n_{\vec{x}}$, i.e. $[\![tl_{cl}\theta]\!]^n_{\vec{x}}$ is an isomorphism.

Thus, we have that for each i, $[\![A_i\theta]\!]^n_{\vec{x}}$ is an isomorphism. But then, by our induction hypothesis, there are τSLD-proofs of $M^{\mathcal{C}[X]}_{\vec{x}}(A_i\theta)$ for each i.

We may then build a τSLD-proof of $M^{\mathcal{C}[X]}_{\vec{x}}$(Q(t)) as follows. Using the clause diagram on the left below, we may take one τSLD-derivation step using $\langle id, \theta\rangle$ as mgu. Then our new goal would be the diagram on the right below. But, as mentioned above, each of these has a (finite) τSLD-proof.

$$\begin{tikzcd}
M^{\mathcal{C}[X]}_{\vec{y}}(A_1) \arrow[dr] & \cdots & M^{\mathcal{C}[X]}_{\vec{y}}(A_m) \arrow[dl] & & M^{\mathcal{C}[X]}_{\vec{x}}(A_1\theta) \arrow[dr] & \cdots & M^{\mathcal{C}[X]}_{\vec{x}}(A_m\theta) \arrow[dl] \\
& M(\vec{y}) & M^{\mathcal{C}[X]}_{\vec{y}}(Q(r)) \arrow[l] & & & M(\vec{x}) &
\end{tikzcd}$$

□

Lemma 4.15 *Suppose the pullback $[\![Bt]\!]^*_{\vec{x}} \rightarrowtail [\![\vec{x}]\!]^*$ of the monic $[\![B]\!]^* \rightarrowtail b$ interpreting Bt in $\mathcal{C}^*$ is an isomorphism. Then, for some k so is the corresponding pullback $[\![Bt]\!]^k_{\vec{x}} \rightarrowtail [\![\vec{x}]\!]^*$ interpreting Bt in $\mathcal{C}_k$.*

Proof. Suppose pulling back $[\![B]\!]^*$ along t in $\mathcal{C}^*$ gives rise to an isomorphism γ:

$$\begin{tikzcd}
{[\![Bt]\!]^*_{\vec{x}}} \arrow[r, "\beta"] \arrow[d, tail, "\gamma"'] \arrow[dr, phantom, "\lrcorner", very near start] & {[\![B]\!]^*} \arrow[d, tail] & \textit{or equivalently} & {[\![\vec{x}]\!]} \arrow[r, "\gamma^{-1}\beta"] \arrow[d, tail, "id"'] \arrow[dr, phantom, "\lrcorner", very near start] & {[\![B]\!]^*} \arrow[d, tail] \\
{[\![\vec{x}]\!]} \arrow[r, "{[\![t]\!]_{\vec{x}}}"'] & b & & {[\![\vec{x}]\!]} \arrow[r, "{[\![t]\!]_{\vec{x}}}"'] & b
\end{tikzcd}$$

Then $[\![\vec{x}]\!] \xrightarrow{\gamma^{-1}\beta} [\![B]\!]^*$ is an arrow from an object in $\mathcal{C}$, i.e. a representable functor in $\widehat{\mathcal{C}}$ to $[\![B]\!]^* \simeq \bigcup [\![B]\!]^n$. By the coprimeness of $\mathcal{C}$-objects in $\widehat{\mathcal{C}}$, $\gamma^{-1}\beta$ factors through some $[\![B]\!]^k$ as $[\![\vec{x}]\!] \xrightarrow{\delta} [\![B]\!]^k \subset [\![B]\!]^*$. It is then straightforward to show that the arrows id and δ are pullbacks of the arrows $[\![t]\!]_{\vec{x}}$ and $[\![B]\!]^k \rightarrowtail b$ respectively. □

Theorem 4.16 (Completeness of τSLD-Resolution) *Let $\mathcal{C}$ be a category and $\mathcal{C}[X]$ a logic programming category with extended structure $M : \mathcal{L} \to \mathcal{C}[X]$ as described previously. Let P^τ be a τ-logic program in the language $\mathcal{L}$ and G a definite goal diagram. If G holds in every model of P^τ, then $P \vdash_{\mathcal{C}} G$ (ie: there is a τSLD-proof of G from P^τ in the category $\mathcal{C}[X]$).*

Proof. Let P^τ as described above have clause diagrams $M^{\mathcal{C}[X]}(cl^1), \ldots, M^{\mathcal{C}[X]}(cl^m)$ and let G be a definite goal diagram (ie diagram at left below) such that for every model of P^τ, G is true.

$$
\begin{tikzcd}
M^{\mathcal{C}[X]}_{\vec{y}}(G_1) \arrow[dr] & \cdots & M^{\mathcal{C}[X]}_{\vec{y}}(G_r) \arrow[dl] \\
 & M(\vec{y}) &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
[\![G_1]\!]^*_{\vec{y}} \arrow[dr, "g_1^*"'] & \cdots & [\![G_r]\!]^*_{\vec{y}} \arrow[dl, "g_r^*"] \\
 & [\![\vec{y}]\!] &
\end{tikzcd}
$$

In particular then, for the model $\mathcal{C}^*$ of P^τ, we have that $H[\ell]$, where ℓ is the limit of the diagram G at left above, is an isomorphism. Since by definition, $H[\ell]$ is the limit of the diagram at right above, if it is an isomorphism, then g_j^* is an isomorphism for each j. But now by Lemma 4.15, g_j^k is an isomorphism for each j, so by Lemma 4.14 there is a τSLD-proof of $M^{\mathcal{C}[X]}_{\vec{y}}(\mathrm{G}_j)$ for each j. $\square$

5 Some Examples of Constraint Categories

Let $\mathcal{L}$ be the language of a one-sorted Horn clause program. Define $\mathcal{H}$, the free category associated with $\mathcal{L}$ (also known as the *empty algebraic theory* for $\mathcal{L}$) to be the small category with *objects* the natural numbers $0, 1, \ldots$, and *arrows* given by the following. For each nonzero pair of objects m, n with $m \leq n$ a projection map $\pi^n_m : n \to m$, a diagonal arrow $\delta^m_n : m \to n$ and, for each function symbol f of arity n in $\mathcal{L}$ an arrow $\boldsymbol{f}: n \to 1$. These arrows satisfy the equations for associativity as well as

$$\pi^n_m \pi^m_k = \pi^n_k \quad \text{and} \quad \delta^m_n \pi^n_m = id_m = \delta^m_m = \pi^m_m$$

This category has products and τ structure, as the reader can check. Furthermore there is a one-to-one correspondence between arrows in $\mathcal{H}$ and (tuples) of terms in the Herbrand universe over $\mathcal{L}$. If we now give $\mathcal{H}$ the canonical interpretation that associates with each f in $\mathcal{L}$ the arrow $\boldsymbol{f}$ and let $H[X_1, \ldots, X_n]$ be the category obtained by adjoining indeterminates for each of the predicate letters in $\mathcal{L}$, then $H[X_1, \ldots, X_n]$ captures logic programming in the sense made precise in the theorem below. Terminology is from [13].

Theorem 5.1 *Let P be a Horn clause program, G a (possibly open) definite goal. Then there is an SLD proof $P \vdash_H G$ over this category if and only if there is a computed answer for $P \cup \{G\}$*

This example is essentially the case studied in [2]. A straightforward adaptation of this construction gives the corresponding free category for many-sorted logic.

We can modify the preceding construction to capture equational constraints. Let $\mathcal{L}$ be a language, possibly many-sorted, and $\mathcal{E}$ an equational theory over $\mathcal{L}$. By taking a quotient of the Herbrand category $\mathcal{H}/\mathcal{E}$ we obtain the algebraic theory associated with $\mathcal{E}$, containing the same objects as $\mathcal{H}$ and congruence classes of arrows of $\mathcal{H}$ as morphisms. The construction is well known (see e.g. [18]). We then *add mgu's* by selectively completing this category by adjoining terminal cones (i.e. pullbacks) to all commutative *syntactic* squares, that is to say, to all diagrams consisting of interpretations of terms in $\mathcal{L}$ and their corresponding unifiers. We obtain a category in which a pullback of the arrows corresponding to two terms exists precisely when the terms are $\mathcal{E}$-unifiable, that is to say, when there is a substitution θ such that $\mathcal{E} \models s\theta = t\theta$. SLD-resolution over this category corresponds to finding a computed answer with equational constraints from $\mathcal{E}$.

Note that for domains with pairing or projection operations (see e.g. [19, 12]), first order constraints are equivalent to (variable-free) equational set constraints using the allegory-theoretic connectives $\cap, \cup, \cdot/., (\_)^o$ (and complementation in the case of classical logic). Thus *first-order constraints* can be captured with many-sorted equational categories of the type just discussed.

6 Conclusion

We have shown how a general notion of many-sorted, *non-ground* constraint logic programming can be captured by a categorical abstract syntax in which resolution takes place via pullbacks in a τ-category with canonical products, monics and limits. A categorical analogue of the T_P operator of Kowalski and Van Emden gives a completeness theorem for this notion in semantic categories defined via the Yoneda embedding.

This categorical framework opens the way for a categorical approach to the semantics of uniform proof systems and higher-order logic programming [16] and as well as for the formalization of control. It would be interesting to study connections with the hyperdoctrinal constraints of [17], and with the π domains and non-ground semantics of [1]. Also worth exploring in this context is a categorical theory of abstract interpretation based on adjunctions over the Yoneda category $\widehat{\mathcal{C}}$.

References

[1] G. Levi A. Bossi, M. Gabbrielli and M. Martelli. The s-semantics approach: Theory and applications. *Journal of Logic Programming*, 19-20, 1994.

[2] A. Asperti and S. Martini. Projections instead of variables, a category theoretic interpretation of logic programs. In *Proceedings of the 6^{th} International Conference on Logic Programming*, pages 337–352. MIT Press, 1989.

[3] Michael Barr and Charles Wells. *Toposes, Triples, and Theories.* Springer-Verlag, 1985.

[4] A. Corradini and A. Asperti. A categorical model for logic programs: Indexed monoidal categories. In *Proceedings REX Workshop '92.* Springer Lecture Notes in Computer Science, 1992.

[5] A. Corradini and U. Montanari. An algebraic semantics of logic programs as structured transition systems. In *Proceedings of the North American Conference on Logic Programming (NACLP '90).* MIT Press, 1990.

[6] Stacy E. Finkelstein. *Tau Categories and Logic Programming.* PhD thesis, University of Pennsylvania, 1994.

[7] Peter Freyd. On canonizing category theory or on functorializing model theory. A Pamphlet, March 1974.

[8] Peter Freyd and Andre Scedrov. *Categories, Allegories.* North-Holland, 1990.

[9] Joxan Jaffar and Jean-Louis Lassez. Constraint logic programming. In *Proceedings of Symposium on Principles of Programming Languages.* ACM, 1987.

[10] J. Lambek and P.J. Scott. *Introduction to Higher Order Categorical Logic.* Cambridge University Press, 1986.

[11] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: a first introduction to topos theory.* Springer-Verlag, 1992.

[12] James Lipton and Paul Broome. Combinatory logic programming: Computing in relation algebras and allegories. In *Proceedings of 1994 the International Logic Programming Symposium.* MIT Press, 1994.

[13] J. W. Lloyd. *Foundations of Logic Programming.* Springer Verlag, New York, 1987.

[14] M. Martelli M. Falaschi, G. Levi and C. Palamidessi. Declarative modeling of the operational behavior of logic languages. *TCS*, 69(3), 1989.

[15] Michael Makkai and Gonzalo Reyes. *First Order Categorical Logic*, volume 611 of *Lecture Notes in Mathematics.* Springer-Verlag, 1977.

[16] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 1990.

[17] P. Panangaden, V. Saraswat, P.J. Scott, and R.A.G. Seely. A hyperdoctrinal view of constraint systems. In *Lecture Notes in Computer Science 666*. Springer-Verlag, 1993.

[18] Axel Poigne. Algebra categorically. In *Category Theory and Computer Programming, LNCS 240*. Springer-Verlag, 1986.

[19] Alfred Tarski and Steven Givant. *A formalization of Set Theory without variables*. American Mathematical Society, Providence, R.I., 1987.