A New Framework for Declarative Programming

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

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

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

Contents

Abstract

We propose a new framework for the syntax and semantics of Weak Hereditarily Harrop logic programming with constraints, based on resolution over τ-categories: finite product categories with canonical structure.

Constraint information is directly built-in to the notion of signature via categorical syntax. Many-sorted equational constraints are a special case of the formalism which combines features of uniform logic programming languages (modules and hypothetical implication) with those of constraint logic programming. Using the canonical structure supplied by τ-categories, we define a diagrammatic generalization of formulas, goals, programs and resolution proofs up to equality (rather than just up to isomorphism).

We extend the Kowalski-van Emden fixed point interpretation, a cornerstone of declarative semantics, to an operational, non-ground, categorical semantics based on indexing over sorts and programs.

We also introduce a topos-theoretic declarative semantics and show soundness and completeness of resolution proofs and of a sequent calculus over the categorical signature. We conclude with a discussion of semantic perspectives on uniform logic programming.

1 Introduction

Logic programming is based, in principle if not in practice, on the declarative paradigm of programming with *executable specifications*, that is to say, with code whose text has independent mathematical meaning consistent with its input-output behavior. Originally the specifications were taken to be first-order Horn theories, with a restricted proof search, resolution, as the computational engine.

However, the demand for more expressive power and efficiency has led language designers to consider logical extensions to the original Horn Clause core [26, 48, 51, 50, 46, 28]) and to add control features and constructs drawn from other language paradigms (e.g. types [55, 32, 49], partial evaluation [31] and constraints [10, 29, 30], to name just a few). The effect has been to expand the boundaries of the subject and of the very notion of declarative content of a program. A more general syntax and semantics is needed to model these new features and provide criteria for good language design.

Furthermore, there is a substantial gap between semantic methods in the functional, imperative and declarative programming communities. It is therefore hard to evaluate the effectiveness of proposals to add imported features to logic programming within an agreed-upon common framework. A categorical foundation for logic programming seems to be an essential tool in this endeavor. This paper is an effort to provide such a foundation.

Categories and Logic Programming Categorical tools in programming syntax and semantics have enjoyed widespread use for over a decade [56]. Categorical models have been used to give clean, implementation-independent approaches to side effects and state [47, 62, 68, 52], non-determinism [53], type disciplines [15, 33] and other logics for computation [7, 63]. The mathematical treatment of some features, such as parametricity and polymorphism, have *required* categorical tools [57, 21].

Logic programming, however, has developed within a different semantic tradition than that of functional or imperative programming. The divide has narrowed in the last ten years, with the development of new denotational, operational, and abstract interpretations for logic programs [11, 39, 40, 14, 31, 38]. There has been a growing interest in a categorical formulation of these ideas, essentially for the same reasons they proved of interest in other programming paradigms.

Categorical approaches to logic programming features appeared in the mid 1980's in Rydeheard and Burstall's treatment of unification [61], also developed independently by Goguen. In 1989, Asperti and Martini [4] formalized the syntax of Horn clause programming over the Herbrand Universe using first-order categorical logic as developed in e.g. [41] and gave a topos-theoretic semantics. In 1992, Corradini, Asperti and Montanari gave a categorical analysis of logic program transitions [13] and of logic program structure using indexed monoidal categories [12]. In [54] Panangaden, Scott, Seely, Saraswat, gave a categorical formulation of concurrent constraints for logic programming. In 1994 Diaconescu [19] formalized equational constraint Horn Clause programming using categories and institutions. In 1995 the authors gave a new syntax and a categorical generalization of the fixed point semantics for the Horn case. The following year Power and Kinoshita [59] gave an indexed category description of Horn Clause resolution, and more recently, Pym [60] has developed a categorical formulation of Horn clause program evaluation within a typed, realizability-style metamathematical framework.

Our contribution in this paper is a new categorical syntax with canonical structure supplied by finite product τ-categories, a definition of resolution over such categories for Weak Hereditarily Harrop programs, and an operational(fixed point) and declarative categorical semantics, with respect to which resolution and a categorical sequent calculus are shown sound and complete.

Our operational model theory is a generalization of the Kowalski-Van Emden fixed-point semantics based on indexing not just over sorts but programs. We show how indexing of local logic programming fibers over a state category places substitution, program augmentation and state change within a single framework.

Outline of the Paper The following section is devoted to a description of categorical syntax for logic programming. We begin with a definition of formula diagrams over a categorical signature that *builds-in* constraint information. We then define finite-product τ-categories and state some of their properties. We define categorical substitution and unification along the lines already familiar from the literature. In the next sections we give a new construction for adjoining predicates freely to a categorical signature and define programs and clauses over generic atomic predicates. The last two sections are devoted to a treatment of operational (fixed point) interpretations and declarative (topos-theoretic) semantics. We give soundness and completeness of two notions of proof for both and conclude with some examples and a discussion of the implications of our work for a semantics-based notion of the declarative paradigm.

Notational Conventions

The categorical background required in this paper can be found in the first few chapters of most category theory texts. The reader should know what limits, colimits and adjoints are. Some notational conventions are taken from [23]. Compositions of morphisms are given in *diagrammatic order*: the composition

$$A \xrightarrow{f} B \xrightarrow{g} C$$

is denoted fg. *Cartesian Categories* are categories with all finite limits. Categories with all finite products (and terminator) are called FP categories. The pullback of a monic arrow $A \xrightarrow{t} B$ along a co-terminal arrow $C \xrightarrow{f} B$ (to an arrow whose target is C) is denoted $f^{\#}(t)$:

$$\begin{tikzcd} D \arrow[r] \arrow[d, "f^{\#}(t)"'] & A \arrow[d, "t"] \\ C \arrow[r, "f"'] & B \end{tikzcd}$$

We sometimes denote the *source* D of the pulled back arrow $f^{\#}(t)$ by $f^{\#}(A)$.

A *subobject* of an object B is the equivalence class of all monics with target B under the equivalence relation

$$A \xrightarrow{t} B \sim A' \xrightarrow{t'} B \text{ iff there is an iso } A \xrightarrow{i} A'.$$

We will introduce the notion of τ-categories (fully treated in [23]) in order to obtain a notion of canonical representatives of subobjects.

The basic notions of indexed categories required here are introduced in the paper.

2 Syntax over a categorical signature

2.1 Categorical Signatures and Categorical Formulae

In order to combine constraint information with proof search in uniform systems, we wish to generalize the notion of signature to an arbitrary finite product category.

We begin by defining the notion of a *categorical signature* and continue with the notion of *categorical formulae.*

Definition 2.1 *A* **categorical signature** *is a triple* $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ *where* $\mathbb{K}$ *is a category with finite products,* $\mathcal{D}$ *a family of arrows in* $\mathbb{K}$ *and* B *a distinguished class of* monics *in* $\mathbb{K}$ *satisfying the following property: the pullback of any* m *in* B *along any coterminal arrow in* $\mathcal{D}$ *exists.*

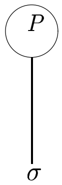
Objects in (occurring as sources and targets of) $\mathcal{D}$ *are* sorts, *and arrows* $f : \sigma \longrightarrow \rho$ *of* $\mathcal{D}$ *are called* terms *of* insort σ *and* outsort ρ. *An arrow whose source is the terminal object and whose target is a sort* σ *is called a* constant *of sort* σ. *Members of* B *are called* predicate tokens. *The target of a predicate token is its sort.*

The maps in $\mathcal{D}$, sometimes called display maps, are a distinguished class of arrows that formalize terms of interest.

When this class of maps is clear from context, we may omit its mention. It will sometimes be useful, for finite sets of predicate tokens, to replace B by a specific sequence of its members.

Predicate tokens are used to build first-order categorical formulae or *formula diagrams* over the signature.

Definition 2.2 *Let* $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ *be a categorical signature. A* **formula diagram** P *of sort* σ *over* $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ *is a labelled diagram with a distinguished object* σ *of* $\mathcal{D}$. *For the purposes of this definition,* *such diagrams will be displayed as a bubble over a distinguished sort, as follows*



to remind the reader that they are not necessarily individual arrows. The class $\mathcal{F}(\mathbb{K}, \mathcal{D}, \mathsf{B})$ *of formula diagrams over* $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ *is given by the following inductive definition.*

- *For any object* σ *in* $\mathcal{K}$*, the identity arrow* $\sigma = \sigma$ *is a formula diagram of sort* σ*, called* $\top_\sigma$*.*
- *The pullback along an arrow* $\sigma \xrightarrow{t} \rho$ *in* $\mathcal{D}$ *of any predicate token of sort* ρ *is a formula diagram of sort* σ*. It is (a fortiori) monic, and is called an* **atomic** *formula diagram.*
- *If* P *and* Q *are formula diagrams of sort* σ *(shown on the left), then so is the labelled diagram* $P \circledast Q$ *(shown on the right) below,*

where $*$ *is either the label* $\Rightarrow$*, or* $\vee$ *or* $\wedge$*.*

- *If* P *is a formula diagram of sort* ρ *and* $\rho \xrightarrow{f} \sigma$ *is an arrow in* $\mathcal{D}$*, then the diagrams*

are formula diagrams of sort σ*, referred to as* $\exists_f P$ *and* $\forall_f P$ *respectively.*

- *If P is a formula diagram of sort ρ and if $\alpha \xrightarrow{t} \rho$ is an arrow in $\mathcal{D}$, then the* **formal pullback** *$(t)^{\#}(P)$ is a formula diagram of sort σ, given by the following inductive definition:*

  1. *if A is a predicate token then $(t)^{\#}(A)$ is just the normal pullback of A along t in $\mathbb{K}$.*
  2. *$(t)^{\#}(P \circledast Q) = (t)^{\#}(P) \circledast (t)^{\#}(Q)$*
  3. *$(t)^{\#}(\exists_f P) = \exists_g((u)^{\#}(P))$ where the bottom square in the following diagram is a (labelled) pullback:*

$$\begin{tikzcd}
u^{\#}P \arrow[d, dash] & P \arrow[d, dash] \\
\eta \arrow[r, "u"] \arrow[d, "{g, \exists}"] & \rho \arrow[d, "{f, \exists}"] \\
\alpha \arrow[r, "t"'] & \sigma
\end{tikzcd}$$

  In other words, to pull back (along t) a quantified diagram $\exists_f P$, first pull back the quantifier arrow f along t, creating the pair of arrows u, g shown, then pull back the formula P along u, and quantify the resulting formula diagram along g (by adding the label $\exists$ to g).

  The $\forall$ definition is similar.

*The formula diagram $(t)^{\#}(P)$ will also be denoted $P(t)$, and called the t-***instantiation** *of the formula diagram P.*

If the quantifying arrow $\rho \xrightarrow{f} \sigma$ in $\forall_f P$ or $\exists_f P$ is of the form $\sigma \times \alpha \xrightarrow{\pi} \sigma$, i.e. a projection out of a product, then the quantifier $\forall_f$ is often written $\forall_{x:\alpha}$, as it coincides with the conventional notion of quantification in logic.

Henceforth, we shall drop the bubble representation of formula diagrams and display them as monics over a distinguished sort.

Definition 2.3 (Theories) *Let σ be an object of $\mathbb{K}$. We define a* **σ-theory** *to be a set of formula diagrams of the sort σ.*

If $\rho \xrightarrow{t} \sigma$ is an arrow in $\mathbb{K}$, and P is a σ-theory, then the formal pullback $t^{\#}(P)$ is the ρ-theory obtained by taking the pullback along t of each formula diagram in P.

When it is not necessary to name σ explicitly, we will simply refer to such a set as a theory. We can think of a σ-theory as a labelled diagram consisting of formula diagrams of sort σ all displayed over a single occurrence of the distinguished sort σ with commas as labels in between playing the role of an extra top-level binary connective. As our use of the word *set* indicates, we do not distinguish between theories displayed in different order, or those with repetitions of formula diagrams.

Remark 2.4 (sort extension) *Any formula diagram P over sort ρ may be thought of as a formula diagram $(\pi)^{\#}(P)$ over (the larger) sort $\rho \times \sigma$ by taking its formal pullback (if it exists) along the projection map $\rho \times \sigma \xrightarrow{\pi} \rho$. This* extension of sort *(denoted $\pi^{\#}P$ or $P(\pi)$) which corresponds in logic to the addition of "dummy" variables to a formula, will be used repeatedly and tacitly in our resolution rules and semantics in order to guarantee that a program and a goal always have a common sort.*

Definition 2.5 *A formula diagram is called* **ground** *if its sort is* $\mathbf{1}$*, the terminal object. Otherwise it is called* **open**.

Predicate diagrams are, in a sense, *variable-free* entities. For example, the formula diagram $P(f)$ for $\rho \xrightarrow{f} \sigma$ an arrow in $\mathbb{K}$ and P a predicate (formula diagram) of sort σ corresponds to the formula $P(f(x))$ in conventional syntactic logic. Only the sort allows us to distinguish between open and ground formulas. Ground predicates play no special role in our treatment. This is a fundamental characteristic of categorical logic, and it leads us in a natural way to a non-ground semantics of logic programming.

2.2 τ-Categories

In developing a categorical notion of syntax, one wishes to interpret sorts and formulas in the category in such a manner as to represent them with *particular* objects or arrows of the category. In general, however, categorical constructions are only defined up to isomorphism, so natural proof-theoretic operations, such as instantiation, may involve arbitrary choices at each step. Such a problem would occur, for example, in the interpretation of product sorts since n-ary products in a category are defined only up to isomorphism, and so are not strictly associative (the associativity map is only required to be an isomorphism instead of an identity).

As a solution, one may work in τ-categories. τ-categories are a special class of finite-product categories in which there is a canonical cartesian structure. This canonical structure allows one to make non-arbitrary choices when interpreting sorts, formulas and proofs. In such categories, since products are strictly associative, there is a unique, *canonical* n-fold product. There is also a unique terminator, as well as a canonical set of monic arrows which are closed under (canonical) pullbacks.

τ-categories are described in detail in [23], but we present here the relevant definitions and properties for use in categorical syntax. Note, however, that 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 τ-categories is not especially restrictive since *every small finite limit category is equivalent to a τ-category* ([23]), thus one can always take a finite-limit completion of a base category and pass to the equivalent τ-category.

We begin by defining the notion of *table*, which is a certain family of arrows with a common source. One should think of a table (up to isomorphism) as denoting a relation. Any table is equivalent to a monic composed with a product diagram.

Definition 2.6 *A* **table** *is an object T together 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 $\langle T; x_1, \ldots, x_n \rangle$.*

Or diagramatically, a table is given by:

$$\begin{tikzcd}
 & & T \arrow[dll, "x_1"'] \arrow[dl, "x_2"] \arrow[drr, "x_n"] & & \\
A_1 & A_2 & & \cdots & A_n
\end{tikzcd}$$

The following two technical definitions are necessary for the definition of τ-categories. The first defines what it means for an arrow to not contribute to the monic nature of the table and the second defines a composition of tables.

Definition 2.7 *Given a table $\langle T; x_1, \ldots, x_n \rangle$ we say that x_j is a* **short column** *if for every $f, g : X \to T$ such that $fx_j \neq gx_j$ there exists $i < j$ such that $fx_i \neq gx_i$.*

In other words, $\langle x_1, \ldots, x_{j-1} \rangle$ is as monic as $\langle x_1, \ldots, x_{j-1}, x_j \rangle$. 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 2.8 *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 2.9 *A* **τ-category** *is a category with all finite products and a distinguished class of tables, denoted τ, such that*

1. *Every table is isomorphic to a unique table in τ.*
2. *$\langle T; id_T \rangle \in \tau$, 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 $\langle T; x_1, \ldots, \hat{x}_j, \ldots, x_n \rangle \in \tau$*

Axiom 1 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 $\langle A \times B; \pi_l, \pi_r \rangle$ is canonical if it is a τ-table. Similarly, a pullback diagram

$$\begin{tikzcd}
P \arrow[r, "p_2"] \arrow[d, "p_1"'] \arrow[dr, phantom, very near start, "\lrcorner"] & B \arrow[d] \\
A \arrow[r] & C
\end{tikzcd}$$

is canonical if $\langle P; p_1, p_2\rangle \in \tau$. These axioms also imply that the category has a unique terminal object.

In addition to guaranteeing a canonical choice of limits, these axioms ensure that those canonical limits satisfy several important properties ([23]).

Lemma 2.10

1. *Canonical products are strictly associative.*
2. *The terminator is strictly a two-sided unit.*
3. *If each square in*

$$\begin{tikzcd} A \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & C \arrow[d] \\ D \arrow[r] & E \arrow[r] & F \end{tikzcd}$$

   is a canonical pullback then so is the rectangle.

The following properties are essential for formalizing syntax in a canonical way in a τ-category.

Lemma 2.11

1. *The canonical pullback of a τ-monic along any arrow is a τ-monic.*
2. *The canonical pullback of a left projection from a τ-product table along any arrow is also a left projection from a τ-product table.*

In other words, the canonical nature of formula diagrams is preserved under canonical pullbacks. *For the remainder of the paper we assume all syntax is defined over a τ-category.*

2.3 Interpreting Conventional Logical Structure over a Category

The notion of categorical signature given in definition 2.1, generalizes the conventional notion of a first-order, many-sorted language. By combining this notion with that of τ-categories, or more precisely, by requiring that a categorical signature $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ consist of a τ-category $\mathbb{K}$ and a distinguished class B of τ-monics, we may give a canonical generalization of a language, as well as a strict interpretation of conventional logic into this τ-categorical signature.

Definition 2.12 *Let $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ be a τ-categorical signature. A pre-interpretation of the language $\mathcal{L}$ into $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ is a function $M : \mathcal{L} \to \mathbb{K}$ such that:*

1. *for every sort σ in $\mathcal{L}$, $\mathcal{M}(\sigma)$ is an object of $\mathcal{D}$*
2. *for every operation symbol f in $\mathcal{L}$of input sort $\sigma_1\sigma_2\cdots\sigma_n$, and of output sort ρ,*

$$M(f) : M(\vec{\sigma}) \ \to M(\rho)$$

   is a morphism in $\mathcal{D}$. Constants are to 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 $\mathcal{D}$.

3. M maps each relation symbol R of sort σ of $\mathcal{L}$ to a morphism $M(R) \rightarrowtail M(\sigma)$ of B. $M(\vec{\sigma})$ denotes the canonical product $M(\sigma_1) \times \cdots \times M(\sigma_n)$.

2.3.1 Interpretation of the Terms and Formulae

Given a $(\mathbb{K}, \mathcal{D}, \mathsf{B})$-pre-interpretation M defining the symbols of the language $\mathcal{L}$ in the category $\mathbb{K}$, we may then interpret the terms inductively as described below. The definition describes each term as an arrow in $\mathbb{K}$ relative to a sequence of distinct variables containing all of the free variables in the term. M is called an *interpretation* if the image of every term lies in the class of display maps $\mathcal{D}$.

Let $\vec{x} = (x_1, \ldots, x_m)$ be a sequence of distinct variables of sorts $\rho_1, \rho_2, \ldots, \rho_m$ (respectively). In order to remind us which variable sequence is under consideration we will adopt the convention of denoting by $M(\vec{x})$ the product $M(\rho_1) \times \cdots \times M(\rho_m)$ in $\mathbb{K}$. Thus, we will drop specific reference to the sorts of the x_i's unless needed in a particular instance.

Definition 2.13 *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 $\mathbb{K}$ 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**$_n$ *If each t_i is of sort σ_i, then $M_{\vec{x}}(ft_1 \cdots t_n)$ is defined as the following composition:*

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

Definition 2.14 *Let $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ be a categorical signature, and M an interpretation of the language $\mathcal{L}$ into $(\mathbb{K}, \mathcal{D}, \mathsf{B})$. We say M is* **surjective** *if every object of $\mathbb{K}$ is the image of a sort in $\mathcal{L}$,* **full** *if every arrow in $\mathbb{K}$ between images of sorts in $\mathcal{L}$ is the image of some term in $\mathcal{L}$, and* **faithful** *if distinct terms are interpreted as distinct arrows.*

The interpretation M also extends to a (partial) mapping from $\mathcal{L}$-formulas to formula diagrams over $(\mathbb{K}, \mathcal{D}, \mathsf{B})$.

Definition 2.15 *Let A be an $\mathcal{L}$-formula of sort σ with free variables among $\vec{x} = (x_1, \ldots, x_m)$. Then $M_{\vec{x}}(A)$ is the following diagram in $\mathcal{F}(\mathbb{K}, \mathcal{D}, \mathsf{B})$:*

$\underline{A = B(t)}$ **$\underline{\text{for } B \text{ a relation symbol in } \mathcal{L}\text{:}}$** *$M_{\vec{x}}(A)$ is the pullback $(t)^{\#}(M(B))$ provided it exists.*

$\underline{A = B * C\text{:}}$ *$M_{\vec{x}}(A)$ is the formula diagram $M_{\vec{x}}(B) \circledast M_{\vec{x}}(C)$.* $\quad$ *$(* \in \{\vee, \wedge \Rightarrow\})$.*

$\underline{A = \exists_{y:\rho} B}$ *$M_{\vec{x}}(A) = \exists_{\pi_\ell} M_{\vec{x}\cdot y}(B)$.*

$\underline{A = \forall_{y:\rho} B}$ *$M_{\vec{x}}(A) = \forall_{\pi_\ell} M_{\vec{x}\cdot y}(B)$.*

where $\cdot$ denotes concatenation and π_ℓ is the left projection

$$M(\sigma) \times M(\rho) \longrightarrow M(\sigma).$$

Notice that since $M(B)$ is a member of Π, and so a τ-monic, $M_{\vec{x}}(B(t))$ will also be a τ-monic by lemma 2.11.

2.3.2 Notation and Shared Data

In the preceding section we saw how conventional syntactic formulae are interpreted into a categorical signature and, in particular, how variables are replaced by projection arrows. It is important to note therefore how *sharing* of variables is captured in such a framework. It is formalized entirely through projections and sort information. If P is a predicate token, the diagrams $P(x)$ and $P(y)$ are indistinguishable, since they are both pullbacks of the predicate P along the identity arrow. However, in any context in which the two formulas appear together, such as their conjunction, sort information will tell us whether the arguments are shared or distinct. In such a case, if x and y are distinct, the sorts of $P(x)$ and $P(y)$ must be extended to $\sigma \times \sigma$ along the left and right projections into σ since the sort information must name all the free variables included in a discussion. If we wish to combine $P(x)$ and $P(x)$ (or $P(x)$ and $Q(x)$ for any other predicate Q of sort σ) we will retain the original sort σ. The following diagram illustrates the way the two notions are captured in formula diagrams, where π_o and π_1 are the left and right projections along $\sigma \times \sigma \longrightarrow \sigma$ respectively.

$$
\begin{tikzcd}
P(id) \arrow[dr] & \circledast & Q(id) \arrow[dl] \\
& \sigma &
\end{tikzcd}
\qquad
\begin{tikzcd}
P(\pi_o) \arrow[dr] & \circledast & Q(\pi_1) \arrow[dl] \\
& \sigma \times \sigma &
\end{tikzcd}
$$

$$P(x) * Q(x) \qquad\qquad P(x) * Q(y)$$

2.4 Substitution

Let $\theta = \{x_1 := t_1, \ldots, x_n := t_n\}$ be a substitution for the distinct variables in the sequence $\vec{x} = (x_1, \ldots, x_n)$. 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 $\mathbb{K}$ to be the morphism:

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

The following lemma illustrates the fact that a categorical substitution may be applied to the interpretation of a term via ordinary composition of arrows in $\mathbb{K}$.

Lemma 2.16 *Assume θ, $\vec{x}$ and $\vec{y}$ are as described above. Then $M_{\vec{y}}(s\theta) = M_{\vec{x}}(s) \circ \Theta_{\vec{y}}$.*

The application of a substitution to the interpretation of a formula may now be defined to be the result of taking a (canonical) pullback. As in the definition of formulae, this resulting arrow will automatically be a τ-monic arrow. This definition is then shown to satisfy an intuitive notion of substitution for formulas. Explicitly, for atomic formulae, the application of the substitution via pullback should result in the same arrow as that given by the interpretation of the previously instantiated formula.

Definition 2.17 *Assume θ and $\vec{y}$ are as described above. Given an atomic formula φ, define $M_{\vec{y}}(\varphi\theta)$ to be the arrow q in the following canonical pullback:*

$$\begin{tikzcd}
M_{\vec{y}}(\varphi\theta) \arrow[r] \arrow[d, tail, "q"'] \arrow[dr, phantom, "\lrcorner", very near start] & M_{\vec{x}}(\varphi) \arrow[d, tail] \\
M(\vec{y}) \arrow[r, "\Theta_{\vec{y}}"'] & M(\vec{x})
\end{tikzcd}$$

Lemma 2.18 *For $\varphi := Pu_1 \cdots u_k$, $M_{\vec{y}}(\varphi\theta) = M_{\vec{y}}(P(u_1\theta)\cdots(u_k\theta))$.*

3 Horn Logic Programming Categories

3.1 Generic Predicates and unordered goals

Any distinguished family of predicates (monics) can play the role of predicates in a logic program, but such a collection should be carefully chosen. If the predicates are not stable under pullbacks of arrows designated as program terms (the display maps $\mathcal{D}$) then certain instances may not even exist in the syntax, which is a rather unusual form of failure for a logic program query.

More importantly, in logic programming, atomic queries $A(t)$ should be true or false in relation to the ambient program. Unless one wishes to constrain it in advance of its use in any program a program predicate A should not be true at any instance $A(t)$ in the syntactic category.

For example, consider the program

```
conn(X,X).
conn(X,Y) :- edge(X,Z),conn(Z,Y).

edge(a,b).
edge(a,c).
```

A category of predicates should supply the syntactic raw material of the program: predicate tokens $\mathsf{B} = \{\mathsf{conn}, \mathsf{edge}\}$ of some sort $\sigma \times \sigma$, and a subcategory of display maps containing $\mathbf{1} \xrightarrow{a} \sigma, \mathbf{1} \xrightarrow{b} \sigma, \mathbf{1} \xrightarrow{c} \sigma$ and closed under products (so that the predicates, e.g. $\mathsf{conn}(X, a)$ may be formed). It should not make any instance (pullback) of the predicate true (i.e. an isomorphism onto its sort), since no predicate should evaluate to true *a priori*, that is to say, prior to execution of the program.

We thus seek a notion of *generic* or *freely adjoined predicate*. We will now make this idea precise, and show how to construct them.

Definition 3.1 (Generic Predicates) *Let X be a subobject of some object b in a finite product category $\mathbb{C}$, and let $\mathcal{D}$ be a family of arrows in $\mathbb{C}$.*

We say X is a **generic subobject** *of b with respect to the (display) maps $\mathcal{D}$ if*

- *For every arrow t in $\mathcal{D}$ targeted at b the pullback $t^{\#}(X)$ exists.*
- *No such pullback is an isomorphism.*

Definition 3.2 (The category $\mathbb{C}[X_1,\ldots,X_n]$) *Let $\mathbb{C}$ be an FP category and $\vec{b} = b_1,\ldots,b_n$ a sequence of objects of $\mathbb{C}$. Then $\mathbb{C}[X_1,\ldots,X_n]$, the* category obtained from $\mathbb{C}$ by freely adjoining indeterminate subobjects of $\vec{b}$, *is defined as follows:*

objects: *pairs $\langle A,S\rangle$ where $A \in |\mathbb{C}|$ and S is a sequence $S_1,\ldots,S_n$ of finite sets $S_i \subset Hom_{\mathbb{C}}(A,b_i)$,*

arrows: *triples $\langle A,S\rangle \xrightarrow{f} \langle B,T\rangle$ where $A \xrightarrow{f} B$ is an arrow in $\mathbb{C}$ and $fT \subset S$, that is to say, for every i, $(1 \le i \le n)$ and every $t \in T_i$, $ft_i \in S_i$. The arrow f in $\mathbb{C}$ is called the* **label** *of $\langle A,S\rangle \xrightarrow{f} \langle B,T\rangle$. Composition of arrows is inherited from $\mathbb{C}$. Two arrows $\langle A,S\rangle \xrightarrow{f} \langle B,T\rangle$ and $\langle A',S'\rangle \xrightarrow{f'} \langle B',T'\rangle$ are* **equal** *if they have the same domain and range and if $f = f'$ in $\mathbb{C}$.*

We also call $\mathbb{C}[X_1,\ldots,X_n]$ the category of generic predicates of **sort** *$\vec{b}$.*

Notice that an arrow in $\mathbb{C}[X_1,\ldots,X_n]$ may have an identity arrow in $\mathbb{C}$ as a label, and not even be an isomorphism in $\mathbb{C}[X_1,\ldots,X_n]$. We will be paying special attention to a certain class of such arrows.

Theorem 3.3 *Let $\mathbb{C}$ be a finite product category. The category $\mathbb{C}[X_1,\ldots,X_n]$ has*

- *a terminal object $\langle \mathbf{1},\vec{\emptyset}\rangle$, where $\vec{\emptyset}$ is the sequence $\emptyset,\ldots,\emptyset$ of length n,*
- *products: $\langle A,S\rangle \times \langle B,T\rangle = \langle A\times B, \pi_1 S \cup \pi_2 T\rangle$ where $A \xleftarrow{\pi_1} A\times B \xrightarrow{\pi_2} B$ is a product in $\mathbb{C}$.*

Furthermore, the functor $\mathbb{C} \xrightarrow{\iota} \mathbb{C}[X_1,\ldots,X_n]$ given by mapping objects A to $\langle A,\vec{\emptyset}\rangle$ and arrows $A \xrightarrow{f} B$ to $\langle A,\vec{\emptyset}\rangle \xrightarrow{f} \langle B,\vec{\emptyset}\rangle$, is a limit-preserving, full and faithful embedding.

Functoriality, faithfulness and fullness is obvious from the definition of morphism, composition and equality in $\mathbb{C}[X_1,\ldots,X_n]$. Limit preservation follows from the fact that ι has a left adjoint, namely the forgetful functor U taking objects $\langle A,S\rangle$ to A and arrows to their labels.

Lemma 3.4 *Addition of indeterminate subobjects simultaneously, sequentially, or in permuted order results in isomorphic categories. More precisely:*

1. $\mathbb{C}[X_1,\ldots,X_n] \simeq \mathbb{C}[X_1]\cdots[X_n]$.
2. *Let σ be a permutation of the first n positive integers. Then* $\mathbb{C}[X_1,\ldots,X_n] \simeq \mathbb{C}[X_{\sigma(1)},\ldots,X_{\sigma(n)}]$.

Proof. Straightforward. $\square$

Definition 3.5 *In $\mathbb{C}[X_1,\ldots,X_n]$ define the* indeterminate subobjects $X_1,\ldots,X_n$ *of sorts $b_1,\ldots,b_n$ respectively, to be the subobjects $\langle b_i,J^i\rangle \xrightarrow{id_{b_i}} \langle b,\vec{\emptyset}\rangle$, where the J^i are the* basis vectors

$$(J^i)_k = \begin{cases} \emptyset & \textit{if} \quad i \ne k \\ \{id_{b_i}\} & \textit{o.w.} \end{cases}$$

The following theorem says that the X_i are generic with respect to the class of (images of) arrows from $\mathbb{C}$, which thus comprise a collection of display maps for $\mathbb{C}[X_1,\ldots,X_n]$.

Theorem 3.6 *The indeterminate subobjects X_i of b_i are generic with respect to the maps in the image of $Hom_{\mathbb{C}}(\_, b_i)$ under $\mathbb{C} \xrightarrow{\iota} \mathbb{C}[X_1, \ldots, X_n]$.*

Proof. The following diagram is a pullback for any arrow $\langle A, \vec{\emptyset} \rangle \xrightarrow{t} \langle b_i, \vec{\emptyset} \rangle$:

$$\begin{tikzcd}
\langle A, tJ^i \rangle \arrow[r, "t"] \arrow[d, tail, "id_A"'] \arrow[dr, phantom, "\lrcorner", very near start] & \langle b_i, \{id\} \rangle \arrow[d, tail, "id_{b_i}"] \\
\langle A, \vec{\emptyset} \rangle \arrow[r, "t"'] & \langle b_i, \vec{\emptyset} \rangle
\end{tikzcd}$$

so $X(t) = \langle A, tJ^i \rangle \overset{id_A}{\rightarrowtail} \langle A, \vec{\emptyset} \rangle$ exists for all appropriate t. This arrow cannot be an isomorphism in $\mathbb{C}[X_1, \ldots, X_n]$: its inverse, which would have to be labelled with id_A, would have to satisfy $id_A t \in \emptyset$. □

Definition 3.7 *An object $\langle A, H \rangle$ is* **atomic** *if H is of the form tJ^i for a basis vector J^i (see 3.5), and some arrow $A \xrightarrow[t]{} \sigma_i$. That is to say, H is the formula $X_i(t)$.*

We will discuss the lifting of τ-structure from $\mathbb{C}$ to $\mathbb{C}[X_1, \ldots, X_n]$ below. But first, there is a more elementary notion of canonical subobject worth identifying in $\mathbb{C}[X_1, \ldots, X_n]$.

Definition 3.8 *If A is an object of $\mathbb{C}$, we say that the monic $\langle B, S \rangle \overset{f}{\rightarrowtail} \langle A, \vec{\emptyset} \rangle$ is a canonical (representative of a) subobject of $\langle A, \vec{\emptyset} \rangle$ if B is A and the monic f is id_A.*

3.1.1 Generic Predicate Diagrams

A certain family of formulas will play an important role here: the class of *canonical* or *generic* formula diagrams over a base category $\mathbb{C}$, that is to say, those built up from generic predicates. Spelling it out, we mean the formula diagrams over the signature $(\mathbb{K}, \mathcal{D}, \mathsf{B})$ where $\mathbb{K}$ is $\mathbb{C}[X_1, \ldots, X_n]$, $\mathcal{D}$ is the image of the category $\mathbb{C}$ in $\mathbb{C}[X_1, \ldots, X_n]$ and B is the sequence $X_1, \ldots, X_n$ of generic predicate tokens corresponding to some sort $\vec{b}$. We simplify notation for this class.

Definition 3.9 *Let $\mathbb{C}$ be a finite product category and $\boldsymbol{b} = b_1, \ldots, b_n$ a sequence of sorts from $\mathbb{C}$. By the $\boldsymbol{b}$-signature $\mathbb{C}[X_1, \ldots, X_n]$ or the $\boldsymbol{b}$-signature $\mathbb{C}$ we mean the signature $(\mathbb{C}[X_1, \ldots, X_n], \mathbb{C}, \langle X_1, \ldots, X_n \rangle)$ of canonical formula diagrams, where each X_i is a generic predicate token of sort b_i. The class of canonical formula diagrams over the $\boldsymbol{b}$-signature $\mathbb{C}[X_1, \ldots, X_n]$ is denoted $\mathcal{F}(\mathbb{C}[X_1, \ldots, X_n])$.*

When $\boldsymbol{b}$, the sequence of sorts of generic predicate tokens is clear from context its mention may be omitted.

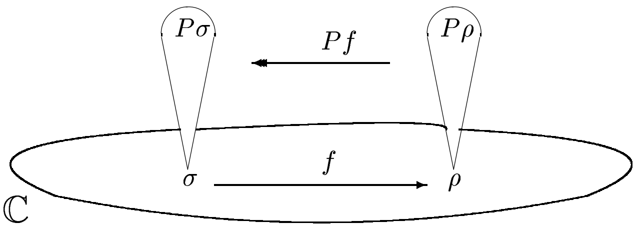
Observe that every object $\langle A, S \rangle$ of $\mathbb{C}[X_1, \ldots, X_n]$ is the source of a canonical subobject of its "sort object" $\langle A, \vec{\emptyset} \rangle$. This allows us to define a natural strict *indexed* structure for $\mathbb{C}[X_1, \ldots, X_n]$ over $\mathbb{C}$, which we then lift to predicate diagrams.

3.2 Predicates via indexed categories

The dependency of predicates on sorts (and later in this paper on underlying programs or states) is nicely captured and generalized via *indexed category structure*. Indexed categories resolve logic and logic programing structure in a clean way into a (vertical) basic logical component (the structure in the fibers) and the (horizontal) predicate logic and substitution component, which, as we shall soon see, is a special case of state change in logic programming.

Indexed categories are a natural categorical formalization of the notion of families of sets indexed by another. Predicates of a given sort σ from a base category $\mathbb{C}$ can be viewed as objects of a category over that sort, called the *fiber* over σ. Arrows between sorts (e.g. substitutions) give rise to induced (co-variant or contravariant) functors between the resulting fibers (such as pullbacks of predicates from one sort to another in the preceding section). Logic programming may be developed in the completely general setting of arbitrary indexed monoidal categories[2, 3]. We will not have need of this generality here, although it is briefly discussed and compared with our approach. It has been treated in [59, 12] for the special case of a base category of sorts. We will consider specific indexed categories over much more general base categories capturing operational and state information, which has not yet been considered in the literature.

We will show that our generic predicate construction is closely related to this general formulation, however, via the so-called *Grothendieck construction*.



Definition 3.10 *A strict indexed $\mathbb{C}$-category (or just a $\mathbb{C}$-category) is a functor*

$$\mathbb{C} \xrightarrow{P} \mathbb{C}AT.$$

An **indexed functor** *from one $\mathbb{C}$-category p to another q is just a natural transformation from p to q. The category $P\sigma$ associated to the object σ of the base category $\mathbb{C}$ is called the* **fiber** *at σ. To each arrow $\sigma \xrightarrow{f} \rho$ between objects in the base category, P associates a functor Pf between the fibers.*

Indexed categories may be covariant or contravariant. The figure illustrates a contravariant indexed category. Indexed functors may also change the base category, but we will not have need of this generality here.

If the word *strict* is dropped, the functoriality condition on p is relaxed: it need only be a pseudo-functor preserving identity and composition up to equivalence rather than on the nose. Pseudo-functoriality arises naturally in the absence of the kind of canonical structure imposed in our work, for example with arbitrary pullbacks.

We denote the action of q on objects σ by $q(\sigma)$, and on arrows $\sigma \xrightarrow{t} \rho$, by q_t.

Definition 3.11 *Let $\mathbb{C}$ be a category and $\boldsymbol{b} = b_1, \ldots, b_n$ a sequence of objects of $\mathbb{C}$. Then*

$$\Pi_{\boldsymbol{b}} : \mathbb{C} \longrightarrow \mathbb{CAT},$$

the indexed cartesian category of generic predicates with sort $\boldsymbol{b}$, is defined as follows. Each fiber $\Pi_{\boldsymbol{b}}(\sigma)$ has **objects** *the members of*

$$FinPow(\mathbb{C}(\sigma, b_1)) \times \cdots FinPow(\mathbb{C}(\sigma, b_n))$$

where $FinPow$ denotes the finite power set, i.e. sequences $S = S_1, \ldots, S_n$ where each S_i is a finite set of arrows from σ to b_i, further endowed with the poset operation of pointwise containment: $S \leq T$ iff for all i $\;S_i \subseteq T_i$. To indicate the fiber in question, we will sometimes write objects as pairs $\langle \sigma, S \rangle$. The action of $\Pi_{\boldsymbol{b}}$ on arrows is given by

$$\Pi_{\boldsymbol{b}}(\sigma \xrightarrow{f} \rho) = f^{\#} : \Pi_{\boldsymbol{b}}(\rho) \longrightarrow \Pi_{\boldsymbol{b}}(\sigma)$$

Notice that the pullback operation referred to by $f^{\#}$ maps $\langle \rho, S \rangle \xrightarrow{id_\rho} \langle \rho, \vec{\emptyset} \rangle$ to $\langle \sigma, fS \rangle \xrightarrow{id_\sigma} \langle \sigma, \vec{\emptyset} \rangle$. Thus $(fg)^{\#}$ is precisely $g^{\#} f^{\#}$ on the nose, so the indexed category is strict.

Finally, observe that each fiber $\Pi_{\boldsymbol{b}}(\sigma)$ is closed under finite unions and intersections, hence $\Pi_{\boldsymbol{b}}$ is finitely complete and co-complete as an indexed category, i.e. in particular, it is cartesian, which is all we will need to make use of in this section.

In the preceding section, we saw how how to define predicates which are generic, i.e. whose meaning is not yet constrained (and will subsequently be defined by a program). We now compare this category with a notion of generalized category of predicates introduced in a related (but somewhat different) form by Hermida [27] and Asperti-Corradini [12].

3.3 From Formula Diagrams to Generalized Formulas

Definition 3.12 *A (generalized) category of predicates over a base category $\mathbb{C}$ is any indexed category q over $\mathbb{C}$. An object in $q(\sigma)$ is called a generalized predicate of sort σ. We say the category of predicates is [FP,cartesian] if each fiber is, respectively, a finite product or a cartesian category. Let $\boldsymbol{b} = b_1, \ldots, b_n$ be a finite sequence of objects of $\mathbb{C}$. Then a q-***signature of sort** $\boldsymbol{b}$ *is a sequence of objects $\boldsymbol{d} = d_1, \ldots, d_n$ with $d_i \in |q(b_i)|$. A signature-preserving morphism of predicate categories is just a natural transformation of indexed categories sending the sequence of distinguished objects in one signature to the other. We call an object of the form $q_t(d)$, where d is a component of $\boldsymbol{d}$ in $|q(b)|$ and t is an arrow in $\mathbb{C}$ with target b, an* **atomic formula** *over the q-signature $\boldsymbol{d}$.* We will always assume here that our categories of predicates are at least FP indexed categories. This assumption may be replaced by a more general monoidal structure (cf. [12]).

The generic predicate category $\Pi_{\boldsymbol{b}}$ comes with a canonical signature over $\boldsymbol{b}$, namely the sequence of *generic objects* $J^1, \cdots, J^n$ (where each J^i is the vector of definition 3.5 above) in $\Pi_{\boldsymbol{b}}(b_i)$.

Lemma 3.13 *Let $\mathbb{C}$ be an FP category, q a cartesian category of predicates, $\boldsymbol{b} = b_1, \ldots, b_n$ a finite sequence of objects of $\mathbb{C}$, and $\boldsymbol{d} = d_1, \ldots, d_n$ a q-signature of sort $\boldsymbol{b}$. Then there is a cartesian, signature-preserving morphism*

$$\Pi_{\boldsymbol{b}} \xrightarrow{\varphi} q$$

unique up to isomorphism.

Thus $\Pi_{\boldsymbol{b}}$ is free in the category of cartesian predicate categories with signatures over $\boldsymbol{b}$.

Proof. Define $\varphi_\sigma(S) = \lim\{q_t(d_i) : t \in S_i, 1 \leq i \leq n\}$. See comments in the proof of theorem 3.18 below. $\square$

Finally, we observe that our original definition of the generic predicate category $\mathbb{C}[X_1, \ldots, X_n]$ can be recovered from the indexed formulation $\Pi_{\boldsymbol{b}}$ just given via a well-known canonical operation: the so-called *Grothendieck* construction of a fibration from an indexed category, presented in e.g. [6], and sketched in the appendix.

Let $\mathsf{G}(\mathbb{C}, \Pi_{\boldsymbol{b}})$ be the source of the Grothendieck fibration induced by the indexed category $\Pi_{\boldsymbol{b}}$, i.e. the category with

objects: pairs $\langle \sigma, S \rangle$ with $\sigma \in |\mathbb{C}|$ and $S \in |\Pi_{\boldsymbol{b}}(\sigma)|$ and

arrows: $\langle \sigma, S \rangle \xrightarrow{\langle \theta, f \rangle} \langle \rho, T \rangle$ for each pair $\sigma \xrightarrow{\theta} \rho$ in $\mathbb{C}$ and $S \xrightarrow{f} \Pi_{\boldsymbol{b}}(\theta)(T)$ an arrow in $\Pi_{\boldsymbol{b}}(\sigma)$. Composition of arrows is given by:

$$\langle \sigma, S \rangle \xrightarrow{\langle \theta, f \rangle} \langle \rho, T \rangle \circ \langle \rho, T \rangle \xrightarrow{\langle \psi, g \rangle} \langle \mu, U \rangle = \langle \sigma, S \rangle \xrightarrow{\langle \theta\psi, \Pi_{\boldsymbol{b}}(\theta)(g) \rangle} \langle \mu, U \rangle$$

Since each $\Pi_{\boldsymbol{b}}(\sigma)$ is a poset, and $\Pi_{\boldsymbol{b}}(\theta)$ is the pullback operation along θ, which is defined by left composition with θ, this reduces to the following definition.

$$\langle \sigma, S \rangle \xrightarrow{\langle \theta, \leq \rangle} \langle \rho, T \rangle$$

is an arrow in t $\mathsf{G}(\mathbb{C}, \Pi_{\boldsymbol{b}})$ whenever $S \leq \theta T$.

This is precisely the category $\mathbb{C}[X_1, \ldots, X_n]$.

We now generalize the notion of formula diagrams, defined in subsection 2.1, along the same lines as definition 3.12. We will mainly be interested in the sequel in goals and programs built from formula diagrams, i.e. from the free instance of this definition, and will not make extensive use of this generality here. The interested reader can supply the appropriate notions of resolution and unification, or read about them in [2, 3].

Definition 3.14 *Let $\mathbb{C}$ be a finite-product category and $\boldsymbol{b} = b_1, \ldots, b_n$ a finite sequence of objects of $\mathbb{C}$. A generalized first-order category of formulas (FOCF) $\mathbb{F}$ over $\mathbb{C}$ with signature $\boldsymbol{d} = d_1, \ldots, d_n$ of sort $\boldsymbol{b}$ is a predicate category over $\mathbb{C}$ with signature $\boldsymbol{d} = d_1, \ldots, d_n$ of sort $\boldsymbol{b}$ (see def. 3.12) with the following additional structure:*

1. *Every fiber $\mathbb{F}(\sigma)$ has an object $\top_\sigma$.*
2. *there are $\mathbb{C}$-indexed covariant bi-functors*

$$\vee, \wedge : \mathbb{F} \times \mathbb{F} \longrightarrow \mathbb{F}$$

and a bi-functor

$$\Rightarrow : \mathbb{F} \times \mathbb{F} \longrightarrow \mathbb{F}$$

contravariant in its first coordinate and covariant in its second.

3. for every $\sigma \xrightarrow{f} \rho \in \mathbb{C}$ there are functors

$$\exists_f, \forall_f : \mathbb{F}(\sigma) \longrightarrow \mathbb{F}(\rho)$$

We have not imposed the requirement that an FOCF be a hyperdoctrine (that existential and universal quantification along f be left and right adjoints to $\mathbb{F}_f$ satisfying Beck and Frobenius conditions, see e.g. [65]) because we want categories of formulas to be a more primitive structural notion, and let the semantics do the extra work. The theory could be developed using both sytactic and semantic hyperdoctrines with more notation but essentially along the same lines. This approach to logic programming is taken up in [2, 3].

We observe that formula diagrams can be viewed as a discrete FOCF with $\mathbb{F}(\sigma)$ the set of formula diagrams of sort σ, $\mathbb{F}(\sigma \xrightarrow{\theta} \rho)$ the generalized pullback operation, and with $\vee, \wedge, \Rightarrow, \exists_f, \forall_f$ the operations that form compound formula diagrams from their components.

Now we define a "canonical intersection" functor for the indexed category $\Pi_{\boldsymbol{b}}$:

Definition 3.15 *Let $\cap : \Pi_{\boldsymbol{b}} \times \Pi_{\boldsymbol{b}} \longrightarrow \Pi_{\boldsymbol{b}}$ be defined by $\langle \sigma, S \rangle \cap_\sigma \langle \sigma, T \rangle = \langle \sigma, S \cup T \rangle$.* It is easy to see that that $\cap$ commutes with pullback, i.e. is a morphism of indexed categories.

The reader can check that indexed canonical intersections of canonical subobjects of the same sort yield representatives of the ordinary intersection on the subobject lattices. That is to say $\langle \sigma, S \rangle \cap_\sigma \langle \sigma, T \rangle$ is a representative of the subobject, in $\mathbb{C}[X_1, \ldots, X_n]$, of $[\langle \sigma, S \rangle] \cap [\langle \sigma, T \rangle]$, where brackets denote the subobject equivalence class, and $\cap$ the intersection obtained by pulling back the following diagram.

$$\begin{tikzcd}
\langle \sigma, S \rangle \arrow[dr, "id_A"'] & & \langle \sigma, T \rangle \arrow[dl, "id_A"] \\
& \langle \sigma, \vec{\emptyset} \rangle &
\end{tikzcd}$$

We now return to the fibration $\mathbb{C}[X_1, \ldots, X_n]$ and establish some key properties we will need when we study semantics.

τ-structure

We have defined a notion of canonical subobject for $\mathbb{C}[X_1, \ldots, X_n]$ that is independent of any τ-structure on $\mathbb{C}$. It is natural to ask if, in the presence of such structure, $\mathbb{C}[X_1, \ldots, X_n]$ inherits it, and if it yields the same notion of canonical subobject and intersection. It does.

Theorem 3.16 $\mathbb{C}[X_1, \ldots, X_n]$ *inherits τ-structure from $\mathbb{C}$ via the forgetful functor $U : \mathbb{C}[X_1, \ldots, X_n] \longrightarrow \mathbb{C}$, adjoint to the full and faithful ι of theorem 3.3.*

Proof. We may define $\tau_{\mathbb{C}[X_1,\ldots,X_n]}$ as the collection of all tables whose image under U is in $\tau_{\mathbb{C}}$. It is straightforward to verify that this definition satisfies the axioms of a τ-category. □

Observe that the image under U of any canonical subobject $\langle\sigma, S\rangle \xrightarrow{id_A} \langle\sigma, \vec{\emptyset}\rangle$ is $A \overset{id}{=\!=} A$, which is in $\tau_{\mathbb{C}}$ by axiom 2 of the definition 2.9 of τ-categories. Thus our canonical subobjects are τ-canonical, and our canonical intersections are τ-canonical pullbacks.

The following theorems make precise the fact that $\mathbb{C}[X_1, \ldots, X_n]$ is called the category obtained by freely adjoining the indeterminate subobjects of the sorts $b_1, \ldots, b_n$.

Lemma 3.17 *Every object $\langle\sigma, S\rangle$ is representable as (i.e. equal on the nose to) the canonical intersection*

$$\bigcap\{t^{\#}(X_i) : t \in S_i, 1 \leq i \leq n\}$$

where the pullbacks are canonical: $t^{\#}(X_i) = \langle\sigma, tJ^i\rangle = \langle\sigma, \emptyset \cdots \emptyset \underbrace{\{t\}}_{i} \emptyset \cdots \emptyset\rangle$.

Proof. Immediate: Since $S = \bigcup\{\{t\} : t \in S\}$, the indicated canonical intersection is precisely $\langle\sigma, S\rangle$. □

Theorem 3.18 (Universal Mapping Property) *Suppose $F : \mathbb{C} \longrightarrow \mathbb{D}$ is a limit preserving functor from the finite-product category $\mathbb{C}$ to the finitely complete category $\mathbb{D}$, and that $F(b_i) = d_i$ for $1 \leq i \leq n$. Furthermore, let $B_1, \ldots, B_n$ be a sequence of subobjects of $d_1, \ldots, d_n$ respectively, in $\mathbb{D}$. Then there is a limit-preserving functor $F_{\vec{B}} : \mathbb{C}[X_1, \ldots, X_n] \longrightarrow \mathbb{D}$, unique up to isomorphism, such that the following diagram commutes.*

$$\begin{tikzcd} & \mathbb{C}[X_1, \ldots, X_n] \arrow[dr, "F_{\vec{B}}"] & \\ \mathbb{C} \arrow[ur, "\iota"] \arrow[rr, "F"'] & & \mathbb{D} \end{tikzcd}$$

$F_{\vec{B}}$ is called the **evaluation** *functor induced by the B_i.*

Proof. Define $F_{\vec{B}}$ on objects by $F_{\vec{B}}(\langle\sigma, S\rangle) = \varprojlim\{F(t)^{\#}(B_i) : t \in S_i, 1 \leq i \leq n\}$. The universal mapping property of limits gives us the action on arrows: if $\langle\sigma, S\rangle \xrightarrow{f} \langle\sigma', S'\rangle$ is an arrow in $\mathbb{C}[X_1, \ldots, X_n]$ then $F_{\vec{B}}(\langle\sigma, S\rangle)$, the limit of the family of monics $\{F(t)^{\#}(B_i) : t \in S_i, 1 \leq i \leq n\}$ targeted at FA, is also, by composing with $F(A \xrightarrow{f} B)$ and using properties of pullbacks and of arrows in $\mathbb{C}[X_1, \ldots, X_n]$, a cone over the family of monics $\{F(t)^{\#}(B_i) : t \in S'_i, 1 \leq i \leq n\}$. There is therefore a unique induced arrow $F\langle\sigma, S\rangle \xrightarrow{\theta} F\langle\sigma', S'\rangle$ which is the value of $F(\langle\sigma, S\rangle \xrightarrow{f} \langle\sigma', S'\rangle)$. The details, and those of the proof of limit preservation, are left to the perseverant reader. □

We are interested in a category $\mathbb{D}$ with richer structure, in which case we are able to sharpen this result a bit.

Corollary 3.19 *Assume the category $\mathbb{D}$ in the preceding theorem is $\mathsf{Set}^{\mathbb{C}^o}$ and that F is the Yoneda embedding. Choose the sequence of subobjects B_i of $Fb_i = Hom_{\mathbb{C}}(\_, b_i)$ to be* canonical, *that is to say, pointwise subsets of Fb_i, and take limits in $\mathsf{Set}^{\mathbb{C}^o}$ to be given pointwise (not just up to isomorphism, but on the nose). Then the evaluation functor $F_{\vec{B}}$ of the preceding theorem is unique.*

3.3.1 Unordered Goals

The most elementary notion of query, or of logical state in a logic program, is that of a conjunction of atoms

$$X_{i_1}(t_1), \ldots, X_{i_n}(t_n)$$

where the X_{i_j} are program predicate symbols. We call these *basic* goals.

We will be considering a more general definition of goals in this paper, given by certain formula diagrams. But it is worth noting that an approximate notion is already present in the category $\mathbb{C}[X_1, \ldots, X_n]$ whose objects are effectively *unordered goals*. By the representation lemma, above, objects $\langle \sigma, S \rangle$ are intersections

$$\bigcap \{ t^{\#}(X_i) : t \in S_i, 1 \leq i \leq n \}.$$

These intersections are free in the sense that one can recover all components $t^{\#}(X_i)$ from them, i.e. by reading off the arrows in the S_i, and in the sense of theorem 3.18. Since the S_i are sets, they cannot capture order or repititions of atoms within goals. Thus, this category does not have enough structure to give an account of a deterministic proof theory, or of an operational semantics sensitive to e.g. goal order or selection rules (such as those studied by [36]), but enough to give a standard interpretation along the lines of the Kowalski-Van Emden T_P-semantics.

Definition 3.20 *Let $\mathbb{C}$ be a finite product category, and $X_1, \ldots, X_n$ a sequence of generic predicates over $\mathbb{C}$ of sorts $b_1, \ldots, b_n$. An* **interpretation** *is a functor extending the Yoneda embedding on $\mathbb{C}$, assigning to each X_i some canonical subobject B_i of $\mathbb{C}(\_\_, b_i)$ as in corollary 3.19.* In other words, an interpretation is a functor

$$[\![\,]\!] : \mathbb{C}[X_1, \ldots, X_n] \longrightarrow \mathsf{Set}^{\mathbb{C}^o}$$

- agreeing with the Yoneda embedding on $\mathbb{C}$, and
- mapping $\langle \sigma, S \rangle$ to $\bigcap \{ ([\![t]\!])^{\#}(B_i) : t \in S_i, 1 \leq i \leq n \}$,

where $[\![t]\!]$ means the Yoneda image of t.

We let $\mathcal{I}^{\mathbb{C}}_{\vec{X}}$ denote the set of interpretations for a given sequence $\vec{X} = X_1, \ldots, X_n$ of generic predicates. When $\mathbb{C}$ or $\vec{X}$ are clear from context, we will drop the sub- and super-scripts.

Note that since interpretations always return canonical subobjects, pointwise set-theoretic containment makes the set of interpretations into a poset $\langle \mathcal{I}_{\vec{X}}, \sqsubseteq \rangle$. It is easily seen that for any two interpretations $[\![\,]\!]$ and $[\![\,]\!]'$, we have $[\![\,]\!] \sqsubseteq [\![\,]\!]'$ if and only if for each generic predicate $[\![X_i]\!] \subseteq [\![X_i]\!]'$

In a more general formulation of semantics based on theorem (3.18), with no stipulations about canonical structure, we would obtain such a poset by using the induced pointwise order on subobjects. The following lemma is immediate.

Lemma 3.21 *The poset of interpretations $\mathcal{I}^{\mathbb{C}}_{\vec{X}}$ is a complete lattice with (general) meet and join operations given pointwise, and top and bottom elements $[\![\,]\!]^{\top}$ and $[\![\,]\!]^{\perp}$ given by*

$$[\![X_i]\!]^{\top} = \mathbb{C}(\_\_, b_i) \overset{id}{=\!=} \mathbb{C}(\_\_, b_i) \tag{1}$$

$$[\![X_i]\!]^{\perp} = \mathbf{0} \overset{\mathsf{i}}{\rightarrowtail} \mathbb{C}(\_\_, b_i) \tag{2}$$

It is straightforward to define (see [20]), given a Horn-clause program P built from generic predicates described above, a continuous operator on this lattice whose least-fixed point is a categorical analogue of the least Herbrand model for P. This is precisely what we will do with weak Hereditarily Harrop formulas below. We briefly sketch here the results for the Horn case from [20].

Definition 3.22 *An interpretation* $[\![\ ]\!]$ *is a model of a Horn clause program* P *if for every clause* $tl_{cl} \Rightarrow hd_{cl}$ *we have* $[\![tl]\!] \subset [\![hd]\!]$. *A goal* G *of sort* α *is said to be* true *in the interpretation if the image* $[\![\beta(G)]\!] \in \mathsf{Set}^{\mathbb{C}^o}$ *of the monic*

$$\beta(G) \xrightarrow{id_\alpha} (\alpha, \emptyset)$$

is an isomorphism.

In the following discussion we refer to the empty goal of sort σ by $\Box_\sigma$.

Lemma 3.23 (Soundness) *Let* $[\![\ ]\!]$ *be a model of program* P. *Suppose* G_1 *and* G_2 *are goals, of sorts* α_1, α_2 *respectively, and there is a resolution proof* $G_1 \rightsquigarrow \overset{\theta,c}{\cdots} \rightsquigarrow G_2$, *where* $\alpha_2 \xrightarrow{\theta} \alpha_1$ *is the computed answer substitution. Then* $[\![G_2]\!] \subset [\![\theta G_1]\!]$. *In particular, if* G_2 *is* $\Box_{\alpha_2}$ *then* θG_1 *is true in the model.*

We now define a categorical analogue of the T_P operator of Kowalski and Van Emden, an operator E_P on the lattice of interpretations. To motivate our definition, we briefly review the classical Kowalsi-van Emden semantics [38] of Horn Clause programs, also known as the bottom-up semantics. Associated with a conventional Horn-clause program P consisting of a finite set of first-order *clause* formulas of the form $cl = Tl \Rightarrow h(t)$, t a vector of terms, there is an operator $T_P : \wp(B_P) \longrightarrow \wp(B_P)$ where B_P is the *Herbrand Base*, the set of all ground instantiations of predicates over the signature of P, and $\wp(\ )$ is the power-set operator. T_P is defined as follows:

$$T_P(X) = \{A\theta : Tl \Rightarrow A \text{ is a clause of } P \text{ and } Tl\theta \in X\}. \tag{3}$$

It is easily shown that T_P is continuous on $\langle \wp(B_P), \subseteq \rangle$ (commutes with unions of chains) and that its least fixed point is the least term model M_P of P, in which a goal is true iff it is derivable in classical logic from P iff it is derivable in minimal intuitionistic logic from P iff it is derivable by SLD-resolution from P. By the Tarski-Knaster theorem this model is a union of a countable chain of finitary approximations M_P^n of the least term model, hence the term bottom-up semantics. Since the definition of the T_P operator resembles resolution in reverse it is easy to show that the least-fixed point of T_P (as a singleton model class) is complete with respect to resolution. The remarkable fact is that it is also sound and complete with respect to all of classical deduction, which gives a semantic proof of the fact that the restricted proof search of systems based on resolution (*operational* proofs in the Miller-Nadathur-Scedrov-Pfenning terminology [46]) does not omit any theorems for the restricted class of Horn Theories and existential goals.

This approach to completeness has had a remarkable success in logic programming: countless extensions of declarative programming to e.g. constraints [30], abstract interpretation [5, 11], Hereditarily Harrop programming [44], higher-order logic [17, 18] build on a similar completeness proof. We take this semantic definition as a foundation for a categorical treatment of uniform logic programming.

A slight change in the presentation of the Kowalski-van Emden semantics is more consistent with the spirit of categorical logic. If we define an interpretation $[\![\,]\!]$ as an operator on formulas (determined by its value on predicate symbols or atoms A) whose value is the set of true instances of its argument, then we may define an operator E_P similar to T_P above (3), as an operator on interpretations, thus:

$$E_P(\llbracket\,\rrbracket)(A) = \bigcup\{\theta : \llbracket Tl\theta \rrbracket = \top\}$$

where the union is taken over all clauses $Tl \Rightarrow B$ in the program whose heads B unify via mgu θ with atomic goal A, and where $\top$ denotes the set of all instances. As was shown for the Horn case in [20] this admits a simple generalization in categorical logic.

Program P is now assumed to be a set of Horn formula diagrams over $\mathbb{C}[X_1, \ldots, X_n]$.

Definition 3.24 *Let* $\llbracket\ \rrbracket$ *be an interpretation and* $X_1, \ldots, X_n$ *the sequence of generic predicates in program* P. *Then*

$$E_P(\llbracket\,\rrbracket)(X_i) = \bigcup_{tl \Rightarrow X_i(t) \in P} \mathsf{Im}_{\llbracket t \rrbracket}(\llbracket tl_{cl} \rrbracket) \tag{4}$$

In [20] it is shown that E_P is a continuous operator on the lattice of interpretations, with a least fixed point $\llbracket\ \rrbracket^*$ (called the Herbrand interpretation for P).

Lemma 3.25 *An interpretation* $\llbracket\ \rrbracket$ *is a model of program* P *if and only if it is a pre-fixed point of* E_P *(that is to say,* $E_P(\llbracket\ \rrbracket) \subseteq \llbracket\ \rrbracket$*) and hence, if and only if* $\llbracket\ \rrbracket^* \subseteq \llbracket\ \rrbracket$.

The following is established in [20]

Theorem 3.26 (Completeness) *If* P *is a program,* $\llbracket\ \rrbracket^*$ *its Herbrand interpretation and* G *a goal,* $\llbracket G \rrbracket^*$ *is an isomorphism if and only if there is an SLD proof* $G \rightsquigarrow \overset{id}{\cdots} \rightsquigarrow \Box$

4 Program Syntax

We now present the definitions of two first-order logic programming languages, Horn and Weak Hereditarily Harrop (WHH) formulas, in the context of formula diagrams over a category. We will restrict attention to generic categorical signatures $\mathbb{C}[X]$, i.e. where $\mathbb{C}$ is an arbitrary τ-category and X a finite set of τ-monics freely adjoined to $\mathbb{C}$. These definitions then give rise to two families of programming languages: Horn($\mathbb{C}$) and WHH($\mathbb{C}$), parametrized by the categorical signature.

We recall that uniform programming languages are given by the following data: a set of program formulas $\mathcal{P}$ and a set of goal formulas $\mathcal{G}$ recursively interdefined, as well as a notion of operational derivation $\vdash_o$ of sequents $P \vdash_o G$ where P is a finite subset of $\mathcal{P}$ and $G \in \mathcal{G}$. The sets $\mathcal{P}$ and $\mathcal{G}$ for Horn($\mathbb{C}$) and WHH($\mathbb{C}$) are defined below. The proof theory will be discussed in section 4.3.

In each of the cases, *formula* will mean *formula diagram* over the categorical signature $\mathbb{C}[X]$. In particular *atomic formulas* will be of the form $A = (u)^{\#}(X_i)$ for some $X_i \in X$. $\top_\alpha$ will mean the identity $\alpha = \alpha$ for any object α.

Definition 4.1

- *Horn*($\mathbb{C}$) *program formulae* D *and goal formulae* G *over a categorical signature are given by:*

$$\begin{array}{rcl} G & ::= & \top \mid A \mid G \wedge G \mid G \vee G \\ D & ::= & A \mid G \Rightarrow A \mid D \wedge D \mid \forall_{x:\alpha} D \end{array}$$

- *WHH* $(\mathbb{C})$ *program formulae and goal formulae over a categorical signature are given by:*

$$\begin{array}{rcl} G & ::= & \top \mid A \mid G \wedge G \mid G \vee G \mid D \Rightarrow G \mid \exists_{x:\alpha} D \\ D & ::= & A \mid G \Rightarrow A \mid D \wedge D \mid \forall_{x:\alpha} D \end{array}$$

Definition 4.2 *A* **program** P *is a finite theory consisting of program formulae (necessarily of the same sort). A* **goal** *is a goal formula* G*.*

4.1 Uniform Programming Categories

To continue the thread of generalized predicates introduced in subsection 3.3, we briefly present the natural indexed category generalization of the preceding definitions. Once again take a base category $\mathbb{C}$ and define a **WHH structure** to be a triple of $\mathbb{C}$-indexed categories

$$\mathsf{Goal}, \mathsf{Atom}, \mathsf{Prog} : \mathbb{C} \longrightarrow \mathbb{CAT}$$

endowed with the following indexed category morphisms:

- $\Rightarrow : \mathsf{Goal} \times \mathsf{Atom} \longrightarrow \mathsf{Prog}$
- $\wedge : \mathsf{Prog} \times \mathsf{Prog} \longrightarrow \mathsf{Prog}$
- $\wedge, \vee : \mathsf{Goal} \times \mathsf{Goal} \longrightarrow \mathsf{Goal}$
- $\longrightarrow : \mathsf{Prog} \times \mathsf{Goal} \longrightarrow \mathsf{Goal}$

satisfying

1. $\mathsf{Atom} \subseteq \mathsf{Goal}$
2. $\mathsf{Atom} \subseteq \mathsf{Prog}$

Furthermore, for each $\sigma \xrightarrow{\theta} \rho$ in $\mathbb{C}$, there are functors

1. $\exists_\theta : \mathsf{Goal}(\sigma) \longrightarrow \mathsf{Goal}(\rho)$
2. $\forall_\theta : \mathsf{Prog}(\sigma) \longrightarrow \mathsf{Prog}(\rho)$

Observe that we can easily define a WHH structure using the canonical formula diagrams $\mathcal{F}(\mathbb{C}[X_1, \ldots, X_n], \mathbb{C}, \langle X_1, \ldots, X_n \rangle)$, by taking the appropriate goal, atom and program formula diagrams over a given sort for the fibers, and taking the connectives $\vee, \wedge, \Rightarrow$ to be the free diagram constructors introduced in the first section, and, e.g. for $\sigma \xrightarrow{\theta} \rho$ in $\mathbb{C}$, defining $\exists_\theta : \mathsf{Goal}(\sigma) \longrightarrow \mathsf{Goal}(\rho)$

by the action

$$\begin{tikzcd} & & P \arrow[d, dash] \\ P \arrow[dd] & & \sigma \arrow[dd, "\theta, \exists"] \\ & \rightsquigarrow & \\ \sigma & & \rho \end{tikzcd}$$

4.2 Programs as Sets of Formal Sequents

In order to recapture the familiar notion of program as a set of *clauses* or formal sequents, with a head and tail, we carry out the translation described below. This translation process yields constituent clauses while cumulatively computing the sort extension that is taking place as quantifiers are removed. The effect of the translation is to replace outermost conjunctions with (finite) sets of formulae, and further translate the formulae by

- removing outer occurences of universal quantification, and
- replacing atoms A by the equivalent clause $\top \Rightarrow A$, where $\top = \top_\sigma$ has the same sort as the atom A.

We obtain clausal formulae of the form

$$tl_{cl} \Rightarrow hd_{cl}(tm_{cl})$$

accompanied by a sort-extending substitution (i.e. a projection).

Definition 4.3 *We inductively define the translation κ by*

- $\kappa(\varphi, A) = \{(\varphi, \top \Rightarrow A)\}$, $\top$ *of the same sort as* A.
- $\kappa(\varphi, G \Rightarrow A) = \{(\varphi, G \Rightarrow A)\}$
- $\kappa(\varphi, P_1 \wedge P_2) = \kappa(\varphi, P_1) \cup \kappa(\varphi, P_2)$
- $\kappa(\varphi, \forall_{f:\alpha\to\beta} P) = \kappa(f\varphi, P)$.

It is sometimes convenient to extend κ to sets of program formulae via $\kappa(\varphi, \{P_i : i \in I\}) = \bigcup \kappa(\varphi, P_i)$.

If P is a program, a **clause**, cl, of P is the second component of a member $(\varphi, cl) \in \kappa(P)$. The pair (φ, cl) will be called an annotated clause. We will write $\kappa(P)$ as abbreviated notation for $\kappa(id, P)$.

Each clause yielded by this translation is of the form $G \Rightarrow A$, where A is an atomic formula diagram $X_i(t)$ with X_i a predicate token and t an arrow in $\mathbb{C}$. Given a clause cl, we define the **head** of the clause, hd_{cl}, to be the token X_i, the **tail** of the clause, tl_{cl}, to be the goal formula G and the **head term** of the clause, tm_{cl}, to be the arrow t.

κ is a (non-sort-preserving) mapping from the set of program formulae to the power set of the class of all formula diagrams. It induces a partial order on programs (and hence on each sort-indexed fiber), namely by:

$$P \subseteq P' \text{ iff } \kappa(P) \subseteq \kappa(P').$$

4.3 Resolution

Program computation is carried out via resolution reduction rules. These rules are given as transitions between state vectors: vectors of pairs $\langle P \mid A \rangle$ where P is a program, A a goal.

Definition 4.4 *Let $\mathbb{C}$ be a τ-category and σ an object in $\mathbb{C}$. A σ-**state** is a pair $\langle P \mid A \rangle_\sigma$ where P is a program over $\mathbb{C}$ of sort σ and A a goal diagram over the same category and sort. When clear from context, mention of the sort σ may be omitted.*

*A **state vector** is a finite sequence $\langle P_1 \mid A_1 \rangle$ & $\cdots$ & $\langle P_i \mid A_i \rangle$ & $\cdots$ & $\langle P_n \mid A_n \rangle$ of Σ-vectors of the same sort and signature.*

State vectors form a natural indexed monoidal category over $\mathbb{C}$. The program and formula of a state can be thought of as a combined diagram over a sort σ (which the reader can think of as labelled by the "connective" $|$). The projections occurring in the goal formula and program indicate which data (variables) are shared, and which are independent (see subsection 2.3.2). Similarly, state vectors are best thought of as diagrams over a common sort with & as a top-level label.

We define a one-step resolution as a labelled binary relation on state vectors (or, equivalently, a ternary relation on state-vectors$\times$labels$\times$state-vectors, where labels are one of $A, \vee_r, \vee_l, \wedge$ or a substitution-clause pair). For the Horn case Corradini and Asperti have shown how propositional resolution with respect to an ambient program P can be formalized categorically via the free indexed monoidal category (whose objects are tensors of propositional atoms) induced by the clauses of P as arrows. Resolutions correspond to reversed arrows in this category. Predicate resolution (with unification) can then recovered via the Grothendieck construction applied to this indexed category. We do not study the generalization of this approach to Hereditarily Harrop programming here, as we will not need it for our main results. This approach is used for an axiomatic presentation of resolution in [2, 3].

For the rest of this section we will restrict our attention to quantification along projections, which corresponds, in categorical logic, to conventional quantification with respect to variables. The more general quantification yielded by categorical logic provides a new logic-based way of programming with certain constraints. We return to this point at the end of the section.

Resolution reduction rules:

backchain $\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid A_i\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle \overset{\theta\pi,(G\Rightarrow A_i')}{\rightsquigarrow}$
$\langle(\pi^{\#}P_1)\theta \mid (\pi^{\#}A_1)\theta\rangle \;\&\; \cdots \;\&\; \langle(\pi^{\#}P_i)\theta \mid G\theta\rangle \;\&\; \cdots \;\&\; \langle(\pi^{\#}P_n)\theta \mid (\pi^{\#}A_n)\theta\rangle$

for atomic formula diagrams A_i, clause diagrams $(G \Rightarrow A_i')$ and substitution arrows $\theta\pi$, where

- $(\pi, G \Rightarrow A_i') \in \kappa(P_i)$
- θ is a unifier of the (sort-extended) atomic goal diagram $\pi^{\#}A_i$ and the head A_i' of the selected clause.

This rule extends conventional backchaining to categorical logic. The only significant novelty is the lack of a canonical choice of unifiers (such as a most general unifier) in the absence of any constraints on the underlying category. All resulting states appear pulled back along π because the clause $G \Rightarrow A_i'$ whose head is unified with the selected goal A_i may result from the peeling off of universal quantifiers in the original program, the effect of which has been to extend the sort of the whole state-vector.

augment $\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid A \Rightarrow B\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle \overset{A}{\rightsquigarrow}$
$\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \wedge A \mid B\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle$

instance $\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid \exists_{x:\alpha}A_i\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle \overset{\pi}{\rightsquigarrow}$
$\langle \pi^{\#}P_1 \mid \pi^{\#}A_1\rangle \;\&\; \cdots \;\&\; \langle \pi^{\#}P_i \mid A_i\rangle \;\&\; \cdots \;\&\; \langle \pi^{\#}P_n \mid \pi^{\#}A_n\rangle$

where π is the projection $\sigma \times \alpha \to \sigma$.

This rule effectively replaces an existentially quantified variable in the selected goal by a free one which must now be instantiated by subsequent proof search, and which therefore behaves like a logic variable.

and $\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid A \wedge B\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle \overset{\wedge}{\rightsquigarrow}$
$\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid A\rangle \;\&\; \langle P_i \mid B\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle$

or-right $\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid A \vee B\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle \overset{\vee_r}{\rightsquigarrow}$
$\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid B\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle$

or-left $\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid A \vee B\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle \overset{\vee_l}{\rightsquigarrow}$
$\langle P_1 \mid A_1\rangle \;\&\; \cdots \;\&\; \langle P_i \mid A\rangle \;\&\; \cdots \;\&\; \langle P_n \mid A_n\rangle$

A **null** resolution vector is one of the form $\langle P_1 \mid \top\rangle \;\&\; \cdots \;\&\; \langle P_n \mid \top\rangle$

Definition 4.5 *Let P be a program diagram and G a goal diagram over a category $\mathbb{C}$, that is to say, a categorical signature $(\mathbb{C}[X_1, \ldots, X_n], \mathbb{C}, \langle \mathsf{X}_1, \ldots, \mathsf{X}_\mathsf{n}\rangle)$. Then an* **SLD derivation** *is a sequence of reductions starting with (singleton) state vector $\langle P \mid G\rangle$.*

An **operational (SLD) proof** *is a (finite) sequence of reductions $\langle P \mid G\rangle \rightsquigarrow \cdots \rightsquigarrow$ NULL where NULL is a null resolution vector.*

Definition 4.6 *A* **computed answer substitution** *θ is the composition of all the substitutions occurring in the backchain and instance steps of an SLD-proof.*

We are now in a position to define *operational inference* $\vdash_o$ based on the notion of resolution. We must be careful, however, to distinguish between the universal role played by open formulas (formulas of non-terminator sort, corresponding to those containing free variables) in a *sequent* and the existential character they have (as formulas with *logic variables*) in a resolution sequence

$$\langle P \mid G \rangle \leadsto^{\theta} \leadsto \cdots \leadsto \mathrm{NULL}.$$

The intended meaning of such a derivation is that θ has successfully instantiated an existential query and that any variables remaining free after application of θ (i.e. if the source of θ is other than $\mathbf{1}$) are universal. Thus, for example, the existence of the derivation above should be equivalent to the assertion $P\theta \vdash_o G\theta$.

We now make these observations precise in the following definition.

Definition 4.7 *We will say that G is operationally derivable from P and write $P \vdash_o G$ iff there is a program $\tilde{P}$ and a formula $\tilde{G}$ such that $\langle \tilde{P} \mid \tilde{G} \rangle \leadsto^{\theta} \leadsto \cdots \leadsto$ NULL, with computed answer substitution θ, $P = \tilde{P}\theta$ and $G = \tilde{G}\theta$.*

In the next three lemmas we establish several important properties of $\vdash_o$ including a characterization (in lemma 4.9) we will often use: $P \vdash_o G$ iff $\langle P \mid G \rangle \leadsto^{id} \leadsto \cdots \leadsto$ NULL.

Lemma 4.8 *If $\langle P(t) \mid G(t) \rangle \leadsto^{\theta} \leadsto \cdots \leadsto$ NULL, then $\langle P \mid G \rangle \leadsto^{(\theta t)} \leadsto \cdots \leadsto$ NULL.*

Proof. See appendix. □

Lemma 4.9 *$P \vdash_o G$ iff there is an SLD-proof $\langle P \mid G \rangle \leadsto^{id} \leadsto \cdots \leadsto$ NULL with computed answer substitution id.*

Proof. See appendix. □

Lemma 4.10 *If $\langle P \mid G \rangle \leadsto^{(\theta t)} \leadsto \cdots \leadsto$ NULL, then $\langle P(t) \mid G(t) \rangle \leadsto^{\theta} \leadsto \cdots \leadsto$ NULL.*

Proof. Suppose $\langle P \mid G \rangle \leadsto^{(\theta t)} \leadsto \cdots \leadsto$ NULL. By lemma 4.9 then,

$$\langle P(\theta t) \mid G(\theta t) \rangle \leadsto^{id} \leadsto \cdots \leadsto NULL$$

which may be rewritten as

$$\langle (P(t))(\theta) \mid (G(t))(\theta) \rangle \leadsto^{id} \leadsto \cdots \leadsto NULL$$

But then, by lemma 4.8,

$$\langle P(t) \mid G(t) \rangle \leadsto^{\theta} \leadsto \cdots \leadsto NULL$$

□

Remark 4.11 (Syntactic Pullback Property) *The combination of Lemmas 4.8 and 4.10 then gives us that*

$$\langle P(t) \mid G(t)\rangle \rightsquigarrow^{\theta} \rightsquigarrow \cdots \rightsquigarrow NULL \quad \text{iff} \quad \langle P \mid G\rangle \rightsquigarrow^{(\theta t)} \rightsquigarrow \cdots \rightsquigarrow NULL$$

Categorical Quantification In the definition of formula diagrams we have allowed the formation of quantification along arrows $\exists_f G$ where G is a formula diagram of sort σ and $\sigma \xrightarrow{f} \rho$ is an arrow in $\mathbb{C}$. In conventional logic the meaning of an entailment

$$\Gamma(x) \vdash \exists_f G$$

is that for every x there is a t of the appropriate sort such that

$$\Gamma(x) \vdash G(t) \wedge f(t) = x.$$

Without introducing equality reasoning directly, we can formalize such a goal using the diagonal predicate $\Delta \overset{\delta}{\rightarrowtail} \rho \times \rho$ which, e.g. in Set, represents the subset $\{(r, r) : r \in \rho\}$. The identity of $f(t)$ and x is equivalent to the assertion that the pullback Δ' along $f(t) \times id_\rho$ of Δ is an isomorphism. It is straightforward to produce a resolution rule for the goal $\exists_f G$ in terms of such a predicate, but subsequent satisfaction of the predicate requires varying the definition of generic predicate to build in this constraint dynamically during execution. The relevant generic predicate definition is taken up in [42] and [37], and will not be further discussed here.

5 Operational Semantics

One of our key aims in proposing a categorical semantics is to give a sufficiently general treatment of the Kowalski-van Emden *fixed point semantics* of logic programming [38, 67], a remarkably robust approach that has been successfully applied to many variants and extensions of Prolog. Our semantics takes this idea as a cornerstone of a more general treatment of declarative programming. The model theory yielded by approach is also known as bottom-up semantics, since it assigns approximate meanings

$$[\![A]\!]^0 \subseteq [\![A]\!]^1 \subseteq \cdots$$

to goal predicates A in stages, starting with the empty interpretation. Because of the continuity of the operator used to build these approximations, the canonical least fixed point interpretation of each predicate is the union of these approximations.

The approximate models produced by the bottom-up semantics do not coincide with conventional topos (Kripke) models, and are best understood as model-fragments. In fact the approximate interpretation $[\![\;]\!]^n$ is a semantic abstraction of the notion of "derivability in n steps". As a result, this semantics closely mirrors the resolution process and gives a nice characterization of SLD resolution in terms of inductive definablity of predicates in the semantics.

Since these models do not give rise, even in the colimit, to a bona fide topos model of full first-order logic, they do not really capture the *a priori* notion of declarative truth that one associates with the text of a program. For this reason we need to distinguish between the bottom-up semantics, which we will henceforth call the *operational* interpretation, and conventional topos semantics, developed in section 6. In that section we will reconcile the conventional and operational interpretations, thus justifying the operational semantics in terms of the declarative meaning of the program.

5.1 The Operational Interpretation

In [46], Miller et al. proposed uniform proof systems, *sequent systems in which connectives can be given a search-theoretic interpretation,* as a fundamental defining characteristic of logic programming systems.

The semantic operator we introduce below captures this uniformity through the interpretation of several of the logical connectives via *shifts* over a category of *states* (in this case *program-signature* pairs). For example, $[\![A \Rightarrow B]\!]_P = [\![B]\!]_{P \wedge A}$ defines the meaning of $A \Rightarrow B$ *over* P as the meaning of B *over* $P \wedge A$. In short, $[\![\,]\!]$ incorporates a search theoretic interpretation of connectives directly into the semantics.

Our operational interpretation extends the Yoneda embedding of the base syntactic category $\mathbb{C}$ to an assignment of predicates in the Yoneda Topos to states (goal-program pairs of a given sort) in a way that preserves the ordering of states and which satisfies a number of additional properties.

Let $\mathcal{S}_\sigma$ be the set of states of sort σ over $\mathbb{C}[X_1, \ldots, X_n]$, that is to say the set of all triples (G, P, σ), where G is a goal formula diagram and P is a program in $\mathcal{K}$, both of sort σ. Endow $\mathcal{S}_\sigma$ with poset structure as follows:

$$(G, P, \sigma) \subseteq (G, P', \sigma) \text{ iff } P \subseteq P'$$

where we recall the order on program formula diagrams is induced by the translation κ of definition 4.3.

Let $\mathcal{A}_\sigma$, $\mathcal{A}$, be the restriction of $\mathcal{S}_\sigma$ and $\mathcal{S}$ to atomic states $(X_i(t), P, \sigma)$, where X_i is a generic subobject in X.

Let $\mathbb{S}$ be the strict indexed category $\mathbb{C} \longrightarrow \mathbb{CAT}$ given by

- $\mathbb{S}(\sigma) = \mathcal{S}_\sigma$
- $\mathbb{S}(\rho \xrightarrow{t} \sigma) = (t)^{\#} : \mathbb{S}(\sigma) \longrightarrow \mathbb{S}(\rho)$

and let $\mathbb{M}$ be the strict indexed category (also over $\mathbb{C}$) given by

- $\mathbb{M}(\sigma) = Sub(Hom_{\mathbb{C}}(\_, \sigma))$
- $\mathbb{M}(\rho \xrightarrow{t} \sigma) = [Hom_{\mathbb{C}}(\_, t)]^{\#} : Sub(Hom_{\mathbb{C}}(\_, \sigma)) \longrightarrow Sub(Hom_{\mathbb{C}}(\_, \rho))$

where $Sub(Hom_{\mathbb{C}}(\_, \sigma))$ is the complete Heyting algebra of subobjects in $\mathsf{Set}^{(\mathbb{C})^o}$ of the Yoneda image $Hom_{\mathbb{C}}(\_, \sigma)$ of σ. Then an *operational interpretation* is a $\mathbb{C}$-indexed functor

$$[\![\,]\!] : \mathbb{S} \longrightarrow \mathbb{M}.$$

whose action on triples (U, P, σ) is denoted $[\![U]\!]_{P,\sigma}$ (or just $[\![U]\!]_P$ when the sort is clear from context) and satisfying:

1. triples $(G, P, \sigma) \in \mathcal{S}_\sigma$ are mapped to monics with target $Hom_{\mathbb{C}}(\_, \sigma)$
2. $[\![\top]\!]_P$ is mapped to the identity arrow on $Hom_{\mathbb{C}}(\_, \sigma)$.
3. $[\![A \wedge B]\!]_P = [\![A]\!]_P \cap [\![B]\!]_P$
4. $[\![A \vee B]\!]_P = [\![A]\!]_P \cup [\![B]\!]_P$

5. $[\![A \Rightarrow B]\!]_P = [\![B]\!]_{P \wedge A}$

6. $[\![\exists_f A]\!]_P = \mathsf{Im}_{[\![f]\!]} [\![A]\!]_{P(f)}$

where $\mathsf{Im}_{[\![f]\!]}$ is the image along f, also denoted $\exists_{[\![f]\!]}$.

Observe that, by naturality of $[\![\,]\!]$, for every $\rho \xrightarrow{t} \sigma$ in $\mathbb{C}$

$$\begin{array}{ccc} \mathbb{S}(\sigma) & \xrightarrow{[\![\,]\!]} & \mathbb{M}(\sigma) \\ {\scriptstyle \mathbb{S}(t)}\downarrow & & \downarrow{\scriptstyle \mathbb{M}(t)} \\ \mathbb{S}(\rho) & \xrightarrow{[\![\,]\!]} & \mathbb{M}(\rho) \end{array}$$

in other words, $[\![U(t)]\!]_{P(t)} = [Hom_{\mathbb{C}}(\_, t)]^{\#} [\![U]\!]_P$.

Letting $\mathbb{V}$ be the indexed category of state vectors, $[\![\,]\!]$ lifts immediately to

$$[\![\,]\!] : \mathbb{V} \longrightarrow \mathbb{M}$$

via

$$[\![V_1 \& V_2]\!] = [\![V_1]\!] \cap [\![V_2]\!]$$

We note that condition 4 above is what distinguishes operational interpretations from conventional Topos-theoretic (or Kripke) semantics, as it does not interpret implication in terms of a semantic implication operator, thus permitting a considerably simpler notion of model. It effectively carries over into the semantics the search-theoretic character of the implication connective. However, it is interesting to note that it can be built into the very definition of indexed category morphism simply by choosing a more robust base category for indexed structure in which programs play a similar role to sorts, as we show in the next subsection. In effect we show that such treatment of the connective is similar to the treatment of substitution itself as a change-of-fiber, that is to say, a semantically safe change of logical state.

5.2 Indexing over state information

Our choice of notation $[\![G]\!]_P$ for the action of $[\![\,]\!]$ on pairs (G, P) suggests indexing over programs (as well as over sorts). We now make this implicit indexed structure precise.

Let $\mathsf{Prog}_0 : \mathbb{C} \longrightarrow \mathbb{CAT}$ be the indexed category given by:

- $\mathsf{Prog}_0(\sigma) = \{P : P \text{ is a program of sort } \sigma\}$ with arrows generated by the identity arrows, and, for each pair of programs P, A, the *right-adjoin* arrows

$$P \xrightarrow{\alpha_A} P \wedge A,$$

  and

- $\mathsf{Prog}_0(\sigma \xrightarrow{\theta} \rho) = \mathsf{Prog}_0(\rho) \xrightarrow{\theta^{\#}} \mathsf{Prog}_0(\sigma)$

Let GP be the category yielded by the Grothendieck construction $\mathsf{G}(\mathbb{C}, \mathsf{Prog}_0)$, namely the category with objects pairs (P, σ) with P a program of sort σ and arrows

$$(P, \sigma) \xrightarrow{(\alpha_A, \theta)} (Q, \rho)$$

where $\sigma \xrightarrow{\theta} \rho$ in $\mathbb{C}$ and $P \xrightarrow{\alpha_A} \theta^{\#}(Q)$, whenever $\theta^{\#}(Q) = P \wedge A$.

We now define categories of goals and models *indexed over* GP:

$$\mathcal{G}\ell, \mathcal{M} : \mathsf{GP} \longrightarrow \mathbb{CAT}$$

as follows:

- $\mathcal{M}(P, \sigma) = Sub(\mathbb{C}(\_\_, \sigma))$
- $\mathcal{M}[(Q, \rho) \xrightarrow{(\alpha_A, \theta)} (P, \sigma)]$ by pullback along θ

and

- $\mathcal{G}\ell(P, \sigma) = \{(G, P, \sigma) : G \text{ is a goal over } \sigma\}$
- $\mathcal{G}\ell(Q, \rho) \xrightarrow{\mathcal{G}\ell(\alpha_A, \theta)} \mathcal{G}\ell(P, \sigma)$ via $(G, Q, \rho) \mapsto (A \Rightarrow \theta^{\#}(G), P, \sigma)$, where $A \Rightarrow \theta^{\#}(G)$ is an implicative formula diagram.

Then an operational interpretation may be defined as a GP-indexed functor

$$[\![\,]\!] : \mathcal{G}\ell \longrightarrow \mathcal{M}.$$

satisfying conditions 1,2,3,5 above, since condition 4 is now guaranteed by naturality of $[\![\,]\!]$ over the base category GP. If we fix the sort σ and vary programs, the commutativity of

$$\begin{tikzcd}
\mathcal{G}\ell(P \wedge A, \sigma) \arrow[r, "{[\![\,]\!]}"] \arrow[d, "\mathcal{G}\ell(\alpha_A)"'] & \mathcal{M}(P \wedge A, \sigma) \arrow[d, "\mathcal{M}(\alpha_A)"] \\
\mathcal{G}\ell(P, \sigma) \arrow[r, "{[\![\,]\!]}"] & \mathcal{M}(P, \sigma)
\end{tikzcd}$$

implies $[\![A \Rightarrow G]\!]_P = [\![G]\!]_{P \wedge A}$.

By suitable choice of base category we have eliminated explicit enforcement of a *search-theoretic semantic condition* 4 corresponding to a uniform proof step. The only remaining conditions are preservation of connectives.

5.3 Soundness

There is a natural partial order on interpretations.

Definition 5.1 $[\![\,]\!] \subseteq [\![\,]\!]'$ *iff for all goal formulae* A *and every* $P \in \mathcal{K}$ *of the same sort,* $[\![A]\!]_P \subseteq [\![A]\!]'_P$.

It suffices to check the order on atoms.

Lemma 5.2 *If* $[\![X_i(t)]\!]_P \subseteq [\![X_i(t)]\!]'_P$ *for all atoms* $X_i(t)$ *and all* $P \in \mathcal{K}$ *of the same sort, then* $[\![\,]\!] \subseteq [\![\,]\!]'$

Proof. By a straightforward induction on the structure of goals. $\square$

Definition 5.3 *We shall say that an interpretation* $[\![\,]\!]$ *is a* **model** *of a program* P *of sort* σ *if for every program* Q *of the same sort which extends* P *and every clause* $(\pi, tl_{cl} \Rightarrow X_i(\mathrm{tm}_{cl})) \in \kappa(Q)$, *we have* $[\![tl_{cl}]\!]_{Q(\pi)} \subseteq [\![X_i(\mathrm{tm}_{cl})]\!]_{Q(\pi)}$, *where we recall* $\sigma \times \alpha \xrightarrow{\pi} \sigma$ *is the projection from the sort of the clause to that of* P.

Proposition 5.4 (Soundness) *If* $[\![\,]\!]$ *is a model of a program* P, *then for any goal* G *and any program* Q *of the same sort which extends* P, *if* $Q \vdash_o G$ *then* $[\![G]\!]_Q$ *is an isomorphism.*

Proof. By induction on length of proof.

Let $[\![\,]\!]$ be a model of P, and let G be a goal and Q a program extending P, both of the same sort, such that $Q \vdash_o G$.

Consider the first resolution rule of the proof of $Q \vdash_o G$, which by lemma 4.9 may be assumed to have computed answer substitution the identity:

Backchain $\langle Q \mid G \rangle \overset{\theta_1\pi,(tl_{cl} \Rightarrow \tilde{G})}{\rightsquigarrow} \langle (\pi^{\#}Q)\theta_1 \mid tl_{cl}\theta_1 \rangle \rightsquigarrow \cdots \rightsquigarrow^{\varphi} \rightsquigarrow \mathrm{NULL}$

By the induction hypothesis then, $[\![(tl_{cl}\theta_1)\varphi]\!]_Q$ is an isomorphism. But, $[\![(tl_{cl}\theta_1)\varphi]\!]_Q \subseteq [\![(\tilde{G}\theta_1)\varphi]\!]_Q = [\![((\pi^{\#}G)\theta_1)\varphi]\!]_Q = [\![G]\!]_Q$ which must then also be an isomorphism.

Conjunction $\langle Q \mid A \wedge B \rangle \overset{\wedge}{\rightsquigarrow} \langle Q \mid A \rangle \;\&\; \langle Q \mid B \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow \mathrm{NULL}.$

By the induction hypothesis then we know that $[\![A]\!]_Q$ and $[\![B]\!]_Q$ are isomorphisms, as then is $[\![A \wedge B]\!]_Q$.

Disjunction $\langle Q \mid A_1 \vee A_2 \rangle \overset{\vee}{\rightsquigarrow} \langle Q \mid A_i \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow \mathrm{NULL}.$

By the induction hypothesis then we know that $[\![A_i]\!]_Q$ is an isomorphism, as then is $[\![A_1 \vee A_2]\!]_Q$.

Augment $\langle Q \mid A \Rightarrow B \rangle \overset{A}{\rightsquigarrow} \langle Q \wedge A \mid B \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow \mathrm{NULL}.$

By the induction hypothesis then, we know that $[\![A \Rightarrow B]\!]_Q = [\![B]\!]_{Q \wedge \{A\}}$ is an isomorphism.

Instance $\langle Q \mid \exists_{x:\alpha} A \rangle \overset{\pi}{\rightsquigarrow} \langle (\pi^{\#})Q \mid A \rangle \rightsquigarrow \cdots \rightsquigarrow^{\langle id, \theta \rangle} \rightsquigarrow \mathrm{NULL}$. The form $\langle id, \theta \rangle$ of the second computed substitution ψ is dictated by the fact that $\pi\psi$ must be id_α. By the induction hypothesis then, we know that $[\![(\langle id, \theta \rangle)^{\#} A]\!]_Q$ is an isomorphism, i.e., $[\![\top_\alpha]\!] \subseteq ([\![\langle id, \theta \rangle]\!])^{\#} [\![A]\!]_{Q\pi}$. Now using the fact that image is left-adjoint to pullback, we have $\mathsf{Im}_{[\![\langle id, \theta \rangle]\!]} [\![\top_\alpha]\!] \subseteq [\![A]\!]_{Q\pi}$. Taking the image along π of both sides, we have $[\![\top_\alpha]\!] \subseteq [\![\exists_{x:\sigma} A]\!]_Q$ as we wanted to show.

□

We now show that the pullback property of operational interpretations extends to nonatomic goals.

Lemma 5.5 (Pullback Lemma) *Suppose* P *is a program and* A *is a goal. Then*

$$[\![A(t)]\!]_{P(t)} = ([\![t]\!])^{\#} ([\![A]\!]_P).$$

Proof. We argue by induction on the structure of goal diagrams A. The atomic case follows from naturality of $[\![\,]\!]$. The cases $[\![(A_1 \wedge A_2)(t)]\!]_{P(t)}$ and $[\![(A_1 \vee A_2)(t)]\!]_{P(t)}$ follow from the fact that pullbacks (which also have left adjoints) commute with intersections and unions. $[\![(A_1 \Rightarrow A_2)(t)]\!]_{P(t)} = [\![A_2(t)]\!]_{P(t)\wedge\{A_1(t)\}}$, which is equal, by induction hypothesis, to $([\![t]\!])^{\#}([\![A_2]\!]_{P\wedge A_1}) = ([\![t]\!])^{\#}([\![A_1 \Rightarrow A_2]\!]_P)$.

For the existential case, observe that $(t)^{\#}(\exists_\pi G)$, or, equivalently, $(\exists_\pi G)(t)$ is $\exists_{\pi'} G(t \times id)$.

$$\begin{tikzcd}
u^{\#}P \arrow[d, dash] & P \arrow[d, dash] \\
\beta \times \sigma \arrow[r, "u"] \arrow[d, "{g, \exists}"] & \alpha \times \sigma \arrow[d, "{f, \exists}"] \\
\beta \arrow[r, "t"'] & \alpha
\end{tikzcd}$$

So $[\![(t)^{\#}(\exists_\pi G)]\!]_{P(t)} = [\![\exists_{\pi'} G(t \times id)]\!]_{P(t)}$, which, by definition is $\mathsf{Im}_{[\![\pi']\!]} [\![G(t \times id)]\!]_{P(\pi' t)} = \mathsf{Im}_{[\![\pi']\!]} [\![G(t \times id)]\!]_{P(t \times id\pi)}$. By the induction hypothesis, this is $\mathsf{Im}_{[\![\pi']\!]}(([\![t \times id]\!])^{\#} [\![G]\!]_{P(\pi)})$. By the Beck-Frobenius condition in $\mathsf{Set}^{\mathbb{C}^o}$, this is $([\![t]\!])^{\#}(\mathsf{Im}_{[\![\pi]\!]} [\![G]\!]_{P(\pi)})$, which is precisely $([\![t]\!])^{\#} [\![\exists_\pi G]\!]_P$ $\square$

5.4 The Semantic Operator E_P: the WHH case

We now give a categorical analogue of the set-theoretic T_P operator for WHH formula diagrams without reference to ground terms and which, depending on the category $\mathcal{K}$ of programs, leads to different generalizations of the Least Herbrand Universe over arbitrary τ-categories. Recall that we are using the notation Im_f for the left-adjoint of the pullback functor $f^{\#} : Sub(\rho) \longrightarrow Sub(\sigma)$ along any arrow $\sigma \xrightarrow{f} \rho$, called the *image* along f, and sometimes denoted $\exists_f$ in the literature. The fact that it also has a right adjoint in any topos will be used later.

Definition 5.6 *Let $[\![\,]\!]$ be an operational semantics, P a program in $\mathcal{K}$, and $X_i(t)$ an atomic goal, both of the same sort. We define*

$$\boxed{E(P, [\![\,]\!], X_i, t) = \bigcup_{\substack{\theta \mathrm{tm}_{cl} = \theta\pi t \\ (\pi, cl) \in \kappa(P), hd_{cl} = X_i}} \mathsf{Im}_{[\![\theta\pi]\!]} [\![\theta t l_{cl}]\!]_{P\theta\pi}}$$

If we restrict attention to *closed* programs (such as Horn clause programs), the values $E(P, \llbracket\,\rrbracket, X_i, t)$ can be expressed in terms of pullbacks of $E(P, \llbracket\,\rrbracket, X_i)$, given, in turn, by the considerably simpler expression (4) of definition 3.24 However, in the general Hereditarily Harrop case, the presence of free variables in programs requires a more delicate treatment, both in the semantics and the proof theory.

Lemma 5.7 *If $\llbracket\,\rrbracket \subseteq \llbracket\,\rrbracket'$, then for any indeterminate monic X_i, arrow t targeted at the sort of X_i, and program $P \in \mathcal{K}$ of the same sort, $E(P, \llbracket\,\rrbracket, X_i, t) \subseteq E(P, \llbracket\,\rrbracket', X_i, t)$*

Proof. By hypothesis we have that $\llbracket tl_{cl} \rrbracket_{P\pi} \subseteq \llbracket tl_{cl} \rrbracket'_{P\pi}$ for each pair $(\pi, cl) \in \kappa(P)$ such that $hd_{cl} = X_i$. Thus for any unifier θ of πt and tm_{cl}, we also have

$$\mathsf{Im}_{\llbracket \theta\pi \rrbracket}(\llbracket \theta \rrbracket)^{\#}(\llbracket tl_{cl} \rrbracket_{P\pi}) \subseteq \mathsf{Im}_{\llbracket \theta\pi \rrbracket}(\llbracket \theta \rrbracket)^{\#}(\llbracket tl_{cl} \rrbracket'_{P\pi})$$

But now the required containment follows directly from elementary properties of unions. □

Proposition 5.8 *Let $C_1 \xrightarrow{r} C_2$ and $C_0 \xrightarrow{t} C_2$ be arrows in $\mathbb{C}$. If $\llbracket t \rrbracket$ factors through $\mathsf{Im}_{\llbracket r \rrbracket}(F)$ for some subobject F of $\llbracket C_2 \rrbracket$, then t is an instance of r in $\mathbb{C}$ 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

$$\begin{tikzcd}
 & & & \llbracket C_0 \rrbracket \arrow[dlll, dotted, "\varrho"'] \arrow[dl, "u"'] \arrow[d, "\llbracket t \rrbracket"] \\
F \arrow[rr, "e"] \arrow[dr, "F"'] & & \mathsf{Im}_{\llbracket r \rrbracket}(F) \arrow[r, tail, "\mathsf{Im}_{\llbracket r \rrbracket}(F)"] & \llbracket C_2 \rrbracket \\
 & \llbracket C_1 \rrbracket \arrow[urr, "\llbracket r \rrbracket"'] & &
\end{tikzcd}$$

where $e\mathsf{Im}_{\llbracket r \rrbracket}(F)$ is an epi-monic image factorization of $F\llbracket r \rrbracket$ and each $\llbracket C_i \rrbracket$ is a representable functor in $\mathsf{Set}^{\mathbb{C}^o}$, 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 $\mathsf{Im}_{\llbracket r \rrbracket}(F)$) 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, $\varphi r = t$. □

5.5 The Approximate Interpretations and their Properties

Definition 5.9 *The sequence of operational interpretations $\llbracket\,\rrbracket^0, \llbracket\,\rrbracket^1, \ldots, \llbracket\,\rrbracket^n, \ldots$ is given by $\llbracket X_i(t) \rrbracket^0_P$ = the unique map $\mathbf{0} \to \llbracket b \rrbracket$ and $\llbracket X_i(t) \rrbracket^{n+1}_P = E(P, \llbracket\,\rrbracket^n, X_i, t)$ for indeterminate monics X_i, where b = sort of $X_i(t)$ and P.*

Lemma 5.10 $\llbracket\,\rrbracket^n \subseteq \llbracket\,\rrbracket^{n+1}$.

Proof. It suffices to show that $\llbracket X_i(t) \rrbracket^n_P \subseteq \llbracket X_i(t) \rrbracket^{n+1}_P$ for each atomic $X_i(t)$ and program P which extends P_o. In other words, that $\llbracket X_i(t) \rrbracket^n_P \subseteq E(P, \llbracket\,\rrbracket^n, X_i, t)$. We proceed by induction on n. If $n = 0$, then $\llbracket X_i(t) \rrbracket^0_P =$ the unique map $\mathbf{0} \to \llbracket b \rrbracket^0$, where $b =$ sort of $X_i(t)$. But then by initiality of 0, there is a map $\mathbf{0} \to \llbracket X_i(t) \rrbracket^1_P$ such that

$$\begin{tikzcd} \mathbf{0} \arrow[rr] \arrow[dr] & & \llbracket X_i(t) \rrbracket^1_P \arrow[dl] \\ & b & \end{tikzcd}$$

Now suppose that $\llbracket X_i(t) \rrbracket^k_P \subseteq \llbracket X_i(t) \rrbracket^{k+1}_P$, and show that $\llbracket X_i(t) \rrbracket^{k+1}_P \subseteq \llbracket X_i(t) \rrbracket^{k+2}_P$. Since by hypothesis, $\llbracket X_i(t) \rrbracket^k_P \subseteq \llbracket X_i(t) \rrbracket^{k+1}_P$, we know by lemma 5.7 that $\llbracket X_i(t) \rrbracket^{k+1}_P = E(P, \llbracket\,\rrbracket^k, X_i, t) \subseteq E(P, \llbracket\,\rrbracket^{k+1}, X_i, t) = \llbracket X_i(t) \rrbracket^{k+2}_P$. $\square$

We now establish some key properties of the interpretations $\llbracket\,\rrbracket^n$ and give a useful characterization of the fixed point operator E in terms of equalizers in the topos $\mathsf{Set}^{(\mathbb{C})^o}$.

Lemma 5.11 $\llbracket X_i(tv) \rrbracket^n_{P(t)}$ *is an isomorphism iff* $(\llbracket t \rrbracket)^{\#}(\llbracket X_i(v) \rrbracket^n_P)$ *is an isomorphism.*

Proof. See appendix. $\square$

Corollary 5.12

$$(\llbracket t \rrbracket)^{\#}(\llbracket X_i(v) \rrbracket^n_P) = \llbracket X_i(tv) \rrbracket^n_{P(t)}$$

Proof. We show that $\llbracket X_i(tv) \rrbracket^n_{P(t)}(\alpha) = (\llbracket t \rrbracket)^{\#}(\llbracket X_i(v) \rrbracket^n_P)(\alpha)$ for every object α in $\mathbb{C}$. Let γ be an arrow in $\mathbb{C}$ whose source is α and whose target is the source of t.

$$\begin{array}{lll} \gamma \in (\llbracket t \rrbracket)^{\#}(\llbracket X_i(v) \rrbracket^n_P)(\alpha) & \text{iff} & (\gamma t) \in \llbracket X_i(v) \rrbracket^n_P(\alpha) \\ & \text{iff} & (\llbracket \gamma t \rrbracket)^{\#}(\llbracket X_i(v) \rrbracket^n_P) \text{ is an isomorphism} \\ & \text{iff} & \llbracket X_i(\gamma t v) \rrbracket^n_{P(\gamma t)} = \llbracket (X_i(tv))(\gamma) \rrbracket^n_{(P(t))(\gamma)} \text{ is an isomorphism} \\ & \text{iff} & (\llbracket \gamma \rrbracket)^{\#}(\llbracket X_i(tv) \rrbracket^n_{P(t)}) \text{ is an isomorphism} \\ & \text{iff} & \gamma \in \llbracket X_i(tv) \rrbracket^n_{P(t)}(\alpha) \end{array}$$

$\square$

The values of the semantic operator $E(P, [\![\,]\!], X_i, t)$ defined above were described in terms of substitutions from the base syntactic category $\mathbb{C}$. We must consider all unifying substitutions, since in arbitrary categorical signatures the existence of mgu's is not guaranteed. But the Yoneda Topos $\mathsf{Set}^{(\mathbb{C})^o}$ has equalizers, in terms of which we can give a characterization of the operator E.

Theorem 5.13

$$E(P, [\![\,]\!], X_i, v) = \bigcup_{\substack{(\pi, cl) \in \kappa(P_i) \\ hd_{cl}=X_i}} \{\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}([\![\pi v]\!],\, [\![tm_{cl}]\!])}(\mathsf{eq}([\![\pi v]\!],\, [\![tm_{cl}]\!]))^{\#}\, [\![tl_{cl}]\!]_{P\pi}\}$$

In the case where most general unifiers exist in $\mathbb{C}$, the equalizers in the formula above are representable.

Proof. See appendix. □

5.6 The Kowalski-van Emden model

We now define a canonical operational interpretation of a logic program, inspired by the Kowalski-van Emden construction [38, 67] of the least Herbrand model. In one form or another, this construction has provided complete operational semantics for a broad class of logic programming languages [29, 64, 48]. One of the aims of this paper is to show that, suitably reformulated as a topos interpretation, it gives a semantics for Weak Hereditarily Harrop programs in the full generality of categorical signatures, with respect to which resolution is complete. After defining the canonical interpretation, we establish some important properties of models and show the interpretation is initial with respect to all models of a program.

Definition 5.14 *The* **Kowalski-van Emden model** *is defined by letting*

$$[\![X_i(t)]\!]^*_P = \varinjlim\{[\![X_i(t)]\!]^n_P\}$$

where $\varinjlim\{[\![X_i(t)]\!]^n_P\}$ *is the filtered colimit in* $\mathsf{Set}^{(\mathbb{C})^o}$ *of the following diagram of inclusions*

$$[\![X_i(t)]\!]^0_P \rightarrowtail [\![X_i(t)]\!]^1_P \rightarrowtail \cdots \rightarrowtail [\![X_i(t)]\!]^k_P \rightarrowtail \cdots$$

Lemma 5.15 $[\![A]\!]^*_P = \varinjlim\{[\![A]\!]^n_P\}$ *for all formulae A and programs $P \in \mathcal{K}$ of the same sort.*

Proof. Follows by a straightforward induction on formulae. □

Definition 5.16 *An operational interpretation $[\![\,]\!]$ is said to be a* **pre-fixed point** *of E at program P if for every atomic goal $X_i(u)$ of the same sort as P, and for every program Q of the same sort as P extending P*

$$E(Q, [\![\,]\!], X_i, u) \subseteq [\![X_i(u)]\!]_Q$$

and a **fixed point** *of E at P if the containment can be replaced by equality.*

Lemma 5.17 *If $[\![\,]\!]$ is a fixed point of E at P it is a model of P. If it is a model of P it is a pre-fixed point of E at P.*

Proof. Suppose $[\![\,]\!]$ is a fixed point of E at program P of sort σ. Let Q extend P and suppose $B \Rightarrow X_i(u)$ is a clause of Q of sort $\sigma \times \alpha$ (necessarily extending the sort of Q since it has had any possible quantification removed). Then

$$[\![X_i(u)]\!]_{Q(\pi)} = \bigcup_{\substack{(\pi', cl)\,\in\,\kappa(Q(\pi))\\ hd_{cl}=X_i}} \{\mathsf{Im}_{[\![\theta\pi']\!]}([\![\theta]\!])^{\#}\,[\![tl_{cl}]\!]_{Q(\pi'\pi)}\}.$$

Consider the expression in the right hand union corresponding to the clause $B(\pi \times id) \Rightarrow X_i((\pi \times id)u)$ of $Q(\pi)$ with θ chosen to be the unifier $\sigma \times \alpha \xrightarrow{id_\sigma \times \delta} \sigma \times \alpha \times \alpha$ where $\alpha \xrightarrow{\delta} \alpha \times \alpha$ is the diagonal map, as shown in the following diagram

$$\begin{tikzcd}
 & B(\pi \times id) \Rightarrow X_i((\pi \times id)u) \arrow[d] & B \Rightarrow X_i(u) \arrow[d] \\
\sigma \times \alpha \arrow[r, "\theta"] & \sigma \times \alpha \times \alpha \arrow[r, "\pi \times id"] \arrow[d, "\pi'{,}\forall"] & \sigma \times \alpha \arrow[d, "\pi{,}\forall"'] \\
 & \sigma \times \alpha \arrow[r, "\pi"] & \sigma
\end{tikzcd}$$

This expression is

$$\mathsf{Im}_{[\![\theta\pi']\!]}([\![\theta]\!])^{\#}\,[\![B(\pi \times id_\alpha)]\!]_{Q(\pi'\pi)}$$

which is equal to

$$([\![\theta]\!])^{\#}\,[\![B(\pi \times id_\alpha)]\!]_{Q((\pi \times id_\alpha)\pi)}$$

since $\theta(\pi \times id_\alpha) = id_{\alpha\times\sigma} = \theta\pi'$. This yields, using the pullback property,

$$([\![\theta(\pi \times id_\alpha)]\!])^{\#}\,[\![B]\!]_{Q(\pi)}$$

which is equal to $[\![B]\!]_{Q(\pi)}$. This shows $[\![B]\!]_{Q(\pi)} \subseteq [\![X_i(u)]\!]_{Q(\pi)}$. Thus, $[\![\,]\!]$ is a model of P.

For the second claim, suppose $[\![\,]\!]$ is a model of P. Then

$$\begin{aligned}
E(Q, [\![\,]\!], X_i, t) &= \bigcup_{\substack{(\pi, cl)\,\in\,\kappa(Q)\\ hd_{cl}=X_i}} \mathsf{Im}_{[\![\theta\pi]\!]}([\![\theta]\!])^{\#}\,[\![tl_{cl}]\!]_{Q(\pi)} \\
&\subseteq \bigcup_{\substack{(\pi, cl)\,\in\,\kappa(Q)\\ hd_{cl}=X_i}} \mathsf{Im}_{[\![\theta\pi]\!]}([\![\theta]\!])^{\#}\,[\![X_i(tm_{cl})]\!]_{Q(\pi)}
\end{aligned}$$

where the θ range over all unifiers of $X_i(\pi t)$ and $X_i(tm_{cl})$ of clauses in Q for which $hd_{cl} = X_i$. But then

$$\begin{aligned}
([\![\theta]\!])^{\#}\,[\![X_i(tm_{cl})]\!]_{Q(\pi)} &= [\![X_i(\theta tm_{cl})]\!]_{Q(\theta\pi)} \\
&= [\![X_i(\theta\pi t)]\!]_{Q(\theta\pi)} \\
&= ([\![\theta\pi]\!])^{\#}\,[\![X_i(t)]\!]_{Q}
\end{aligned}$$

So every expression in the definition of $E(Q, [\![\,]\!], X_i, t)$ is of the form

$$\mathsf{Im}_{[\![\theta\pi]\!]}([\![\theta\pi]\!])^{\#}\,[\![X_i(t)]\!]_Q$$

which, since $\mathsf{Im}_f(f)^{\#}(A) \subseteq A$ for any f and subobject A, is always a subobject of $[\![X_i(t)]\!]_Q$. Thus

$$E(Q, [\![\,]\!], X_i, t) \subseteq [\![X_i(t)]\!]_Q$$

as we wanted to show. □

Lemma 5.18 *The interpretation* $[\![\,]\!]^*$ *is a model of* P.

Proof. We show that, in fact, E is *continuous*, that is to say, E preserves colimits of subobject chains:

$$[\![X_i(t)]\!]^*_Q = \bigcup_{\substack{(\pi, cl) \in \kappa(Q) \\ hd_{cl} = X_i}} \{\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}([\![tm_{cl}]\!],\,[\![\pi t]\!])}(\mathsf{eq}([\![tm_{cl}]\!],\,[\![\pi t]\!]))^{\#}([\![tl_{cl}]\!]^*_{Q\pi})\}$$

for each atomic formula $X_i(t)$ and each program Q of the same sort extending P. This shows that $[\![\,]\!]^*$ is a fixed point of E at P and hence a model of P.

We will let eq_{cl} stand for $\mathsf{eq}([\![tm_{cl}]\!],\,[\![\pi t]\!])$.

First we will show that

$$[\![X_i(t)]\!]^*_Q \subseteq \bigcup_{\substack{(\pi, cl) \in \kappa(Q) \\ hd_{cl} = X_i}} \{\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}_{cl}}(\mathsf{eq}_{cl})^{\#}([\![tl_{cl}]\!]^*_{Q\pi})\}$$

It suffices then to show that

$$[\![X_i(t)]\!]^n_Q \subseteq \bigcup_{\substack{(\pi, cl) \in \kappa(Q) \\ hd_{cl} = X_i}} \{\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}_{cl}}(\mathsf{eq}_{cl})^{\#}([\![tl_{cl}]\!]^*_{Q\pi})\}$$

By lemma 5.15, $[\![tl_{cl}]\!]^{n-1}_{Q\pi} \subseteq [\![tl_{cl}]\!]^*_{Q\pi}$ and hence

$$\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}_{cl}}(\mathsf{eq}_{cl})^{\#}([\![tl_{cl}]\!]^{n-1}_{Q\pi}) \subseteq \mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}_{cl}}(\mathsf{eq}_{cl})^{\#}([\![tl_{cl}]\!]^*_{Q\pi})$$

Thus,

$$\begin{aligned}
[\![X_i(t)]\!]^n_Q &= \bigcup_{\substack{(\pi, cl) \in \kappa(Q) \\ hd_{cl} = X_i}} \{\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}_{cl}}(\mathsf{eq}_{cl})^{\#}([\![tl_{cl}]\!]^{n-1}_{Q\pi})\} \\
&\subseteq \bigcup_{\substack{(\pi, cl) \in \kappa(Q) \\ hd_{cl} = X_i}} \{\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}_{cl}}(\mathsf{eq}_{cl})^{\#}([\![tl_{cl}]\!]^*_{Q\pi})\}
\end{aligned}$$

Now, for the other direction, it suffices to show that for each pair (π, cl) in $\kappa(Q)$ such that $hd_{cl} = X_i$,

$$\mathsf{Im}_{[\![\pi]\!]}\mathsf{Imeq}_{cl}(\mathsf{eq}_{cl})^{\#}([\![tl_{cl}]\!]^*_{Q\pi}) \subseteq [\![X_i(t)]\!]^*_Q$$

We know by the definition of $[\![X_i(t)]\!]^{k+1}_Q$ that $\mathsf{Im}_{[\![\pi]\!]}\mathsf{Imeq}_{cl}(\mathsf{eq}_{cl})^{\#}[\![tl_{cl}]\!]^k_{Q\pi} \subseteq [\![X_i(t)]\!]^{k+1}_Q$. So we know that for each k

$$\begin{aligned}\mathsf{Imeq}_{cl}[\![tl_{cl}]\!]^k_{Q(\pi)} &\subseteq (\mathsf{eq}_{cl})^{\#}([\![\pi]\!])^{\#}[\![X_i(t)]\!]^{k+1}_Q \\ &\subseteq (\mathsf{eq}_{cl})^{\#}([\![\pi]\!])^{\#}[\![X_i(t)]\!]^*_Q\end{aligned}$$

And so, since by lemma 5.15 we know that $[\![tl_{cl}]\!]^*_{Q\pi} = \varinjlim\{[\![tl_{cl}]\!]^k_{Q\pi}\}$, we may conclude that

$$\begin{aligned}(\mathsf{eq}_{cl})^{\#}[\![tl_{cl}]\!]^*_{Q\pi} &= (\mathsf{eq}_{cl})^{\#}\varinjlim\{[\![tl_{cl}]\!]^k_{Q\pi}\} \\ &= \varinjlim\{(\mathsf{eq}_{cl})^{\#}[\![tl_{cl}]\!]^k_{Q\pi}\} \\ &\subseteq (\mathsf{eq}_{cl})^{\#}([\![\pi]\!])^{\#}[\![X_i(t)]\!]^*_Q\end{aligned}$$

From which the result follows by elementary properties of adjunctions and unions.

□

Lemma 5.19 *The interpretation* $[\![\,]\!]^*$ *is an initial model of* P.

Proof. We must show that for any model $[\![\,]\!]'$ of P, $[\![\,]\!]^* \subseteq [\![\,]\!]'$. In particular, it suffices to show by lemma 5.2 that for all atomic formula diagrams $X_i(t)$ and programs Q extending P, $[\![X_i(t)]\!]^*_Q \subseteq [\![X_i(t)]\!]'_Q$.

We first show by induction on k that for all such $X_i(t)$ and Q both of sort σ, and for all k, $[\![X_i(t)]\!]^k_Q \subseteq [\![X_i(t)]\!]'_Q$. Since $[\![X_i(t)]\!]^0_Q$ = the unique map $0 \to [\![\sigma]\!]$, it is clear that $[\![X_i(t)]\!]^0_Q \subseteq [\![X_i(t)]\!]'_Q$. Now suppose that $[\![X_i(t)]\!]^k_Q \subseteq [\![X_i(t)]\!]'_Q$ for all $X_i(t)$ and Q. By Lemma 5.7 and the fact (lemma 5.17) that any model of P is a pre-fixed point of E at P,

$$[\![X_i(t)]\!]^{k+1}_Q = E(Q, [\![\,]\!]^k, X_i, t) \subseteq E(Q, [\![\,]\!]', X_i, t) \subseteq [\![X_i(t)]\!]'_Q$$

It now follows by the definition of $[\![\,]\!]^*$ that $[\![X_i(t)]\!]^*_Q = \varinjlim\{[\![X_i(t)]\!]^n_Q\} \subseteq [\![X_i(t)]\!]'_Q$. □

5.7 Completeness

We now establish completeness of our operational interpretation by showing that every goal true in the Kowalski-van Emden model of a program is derivable from the program by resolution. We first show that goals valid in the interpretation are valid in some finite approximation using elementary properties of the Yoneda embedding. We also show that in the atomic case such goals are *admitted*, i.e. are an instance of the head of some clause of the program. Finally we show that goals valid in some finite approximation are derivable by constructing a resolution proof, the first step of which is follows from admittance.

Lemma 5.20 *Let* G *be a goal formula and* $P \in \mathcal{K}$ *a program, both of sort* σ. *If* $[\![G]\!]^*_P$ *is an isomorphism, then for some* n, $[\![G]\!]^n_P$ *is an isomorphism.*

Proof. We proceed by induction on formulae.

G atomic Suppose $[\![G]\!]^*_P = [\![X_i(t)]\!]^*_P$ is an isomorphism. Then the identity on $[\![\sigma]\!]$ factors through $\varinjlim\{[\![X_i(t)]\!]^n_P\}$. By coprimeness of $[\![\sigma]\!]$ it factors through some $[\![X_i(t)]\!]^n_P$, which is therefore an isomorphism.

IH Now suppose that the result holds for every program P and all simpler formulae.

G = A$_1\wedge$ A$_2$ Suppose $[\![A_1 \wedge A_2]\!]^*_P = [\![A_1]\!]^*_P \cap [\![A_2]\!]^*_P$ is an isomorphism. Then both $[\![A_1]\!]^*_P$ and $[\![A_2]\!]^*_P$ are isomorphisms. By the induction hypothesis, there exist some n and m such that $[\![A_1]\!]^n_P$ and $[\![A_2]\!]^m_P$ are isomorphisms. But then by lemma 5.10, $[\![A_1 \wedge A_2]\!]^{max(n,m)}_P$ is an isomorphism.

G = A$_1\vee$ A$_2$ Suppose $[\![A_1 \vee A_2]\!]^*_P = [\![A_1]\!]^*_P \cup [\![A_2]\!]^*_P$ is an isomorphism. Then again by coprimeness of representables in the Yoneda embedding, either $[\![A_1]\!]^*_P$ or $[\![A_2]\!]^*_P$ is an isomorphism. But the induction hypothesis then, there exist some n such that either $[\![A_1]\!]^n_P$ or $[\![A_2]\!]^n_P$ is an isomorphism. In either case $[\![A_1 \vee A_2]\!]^n_P$ is an isomorphism.

G = C $\Rightarrow$ D Suppose $[\![C \Rightarrow D]\!]^*_P = [\![D]\!]^*_{P\cup\{C\}}$ is an isomorphism. Then by the induction hypothesis, there is an n such that $[\![D]\!]^n_{P\cup\{C\}} = [\![C \Rightarrow D]\!]^n_P$ is an isomorphism.

G = $\exists_{x:\alpha}A$ Suppose that $[\![\exists_{x:\alpha}A]\!]^*_P = \mathsf{Im}_{[\![\pi]\!]}[\![A]\!]_{P(\pi)}$ is an isomorphism.

In particular, there is an isomorphism $Hom(\sigma,\alpha) \xrightarrow{f} [\![\exists_{x:\alpha}A]\!]^*_P$ inverse to $[\![\exists_{x:\alpha}A]\!]^*_P \subseteq Hom(\sigma,\alpha)$. Observe that there is an epimorphism $[\![A]\!]^*_P \xrightarrow{e} [\![\exists_{x:\alpha}A]\!]^*_P$ (there is an epi from any subobject m to its image under $Hom(\_,\sigma\times\alpha) \xrightarrow{[\![\pi]\!]} Hom(\sigma,\alpha)$, which is precisely the first map of the epi-mono factorization of $m[\![\pi]\!]$). Now we proceed by an argument almost identical to that of proposition (5.8). The source of f is representable, hence projective, so it lifts to an arrow $Hom(\sigma,\alpha) \xrightarrow{f'} [\![A]\!]^*_P$. But the composition $f[\![A]\!]^*_P \rightarrowtail Hom(\_,\sigma\times\alpha)$ is an arrow between representables. By Yoneda it is the image of a substitution $\sigma \xrightarrow{\theta} \sigma\times\alpha$. By construction it is left inverse to $[\![\pi]\!]$, i.e. $\theta = \langle id, t\rangle$ for some $\sigma \xrightarrow{t} \alpha$. Thus $[\![\langle id,t\rangle]\!]$ factors through $[\![A]\!]^*_P$, so $[\![A(\langle id,t\rangle)]\!]^*_P$ is an isomorphism.

By induction hypothesis, there is an n such that $[\![A(\langle id,t\rangle)]\!]^n_P$ is an isomorphism, i.e. $[\![\top_\sigma]\!]^n \subseteq ([\![\langle id,t\rangle]\!])^{\#}[\![A]\!]^n_P$. By the argument at the end of the last case of the proof of soundness (lemma 5.4) $[\![\top_\sigma]\!]^n \subseteq [\![\exists_{x:\alpha}A]\!]^n_P$ so $[\![\exists_{x:\alpha}A]\!]^n_P$ is an isomorphism, as we wanted to show.

$\square$

Definition 5.21 *A clause of a program of the form $tl_{cl} \Rightarrow X_k(u)$ is said to* **admit** *a formula $X_j(t)$ if $j = k$ and t is an instance of u.*

Lemma 5.22 *If an atomic formula $A = X_j(t)$ is valid in all models of the program P, (i.e. is interpreted as an isomorphism) then some clause of P admits it.*

Proof. Consider the model $[\![\,]\!]'$ given by

$$[\![X_i(t)]\!]'_Q = \bigcup_{\substack{(\pi, cl) \in \kappa(Q) \\ hd_{cl}=X_i}} \{\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}([\![\pi t]\!],\,[\![tm_{cl}]\!])}(\mathsf{eq}([\![\pi t]\!],\,[\![tm_{cl}]\!]))^{\#}(id_{[\![\sigma\times\alpha]\!]})\}$$

for any program Q extending P, and where by convention, if $X_i \neq hd_{cl}$ for any $cl \in \kappa(Q)$ then $[\![X_i(t)]\!]'_Q$ (the empty union) shall be the unique arrow $\mathbf{0} \to [\![\sigma]\!]$ for $\sigma =$ sort of $X_i(t)$.

First we show that $[\![\,]\!]'$ is a model of P. We must show that for any clause $cl = (tl_{cl} \Rightarrow X_i(v))$ of sort $\sigma \times \beta$ in any extension Q of P, $[\![tl_{cl}]\!]'_{Q(\pi)} \subseteq [\![X_i(v)]\!]'_{Q(\pi)}$, where π is the projection $\sigma\times\beta \longrightarrow \sigma$.

Thus consider any clause $(\pi : \sigma\times\beta \to \sigma, tl_{cl} \Rightarrow X_i(v)) \in \kappa(Q)$, for any extension Q of P of sort σ. To elucidate the following argument, some discussion of sorts will be useful.

Observe that if Q is a program extending P of sort σ, and $B \Rightarrow X_i(u)$ a (possibly different) clause of Q of sort, say, $\sigma\times\alpha$, then the corresponding clause in a sort-adjusted program $Q(\pi)$ is $B(\pi\times id) \Rightarrow X_i((\pi\times id)u)$, as the following diagram illustrates:

$$\begin{tikzcd}
B(\pi\times id) \Rightarrow X_i((\pi\times id)u) \arrow[d] & B \Rightarrow X_i(u) \arrow[d] \\
\sigma\times\beta\times\alpha \arrow[r, "\pi\times id"] \arrow[d, "\pi', \forall"] & \sigma\times\alpha \arrow[d, "\pi'', \forall"'] \\
\sigma\times\beta \arrow[r, "\pi"] & \sigma
\end{tikzcd}$$

Now we observe that by the definition above, for the particular clause under consideration,

$$[\![X_i(v)]\!]'_{Q(\pi)} = \bigcup_{\substack{(\pi', cl) \in \kappa(Q(\pi)) \\ hd_{cl}=X_i}} \{\mathsf{Im}_{[\![\pi']\!]}\mathsf{Im}_{\mathsf{eq}([\![\pi' v]\!],\,[\![(\pi\times id)u]\!])}(\mathsf{eq}([\![\pi' v]\!],\,[\![(\pi\times id)u]\!]))^{\#}(id_{[\![\sigma\times\beta\times\alpha]\!]})\}$$

But one of the expressions in this union is the one corresponding to the clause $cl = (tl_{cl} \Rightarrow X_i(v))$ itself:

$$\mathsf{Im}_{[\![\pi']\!]}\mathsf{Im}_{\mathsf{eq}([\![\pi' v]\!],\,[\![\pi' v]\!])}(\mathsf{eq}([\![\pi' v]\!],\,[\![\pi' v]\!]))^{\#}(id_{[\![\sigma\times\beta\times\beta]\!]})$$

which is just $id_{[\![\sigma\times\beta]\!]}$, implying $[\![X_i(v)]\!]'_{Q(\pi)}$ is an iso. Hence, trivially, $[\![tl_{cl}]\!]'_{Q(\pi)} \subseteq [\![X_i(v)]\!]'_{Q(\pi)}$. And thus $[\![\,]\!]'$ is a model of P.

Now suppose $X_i(t)$ is any goal valid in all models of P, so in particular in $[\![\,]\!]'$. Then $[\![X_i(t)]\!]'_P$ is an isomorphism. By coprimeness of representables, one of the expressions in the union defining this monic is also an iso. Let us suppose it is

$$\mathsf{Im}_{[\![\pi'']\!]}\mathsf{Im}_{\mathsf{eq}([\![u]\!],\,[\![\pi'' t]\!])}(\mathsf{eq}([\![u]\!],\,[\![\pi'' t]\!]))^{\#}(id_{[\![\sigma\times\alpha]\!]})$$

corresponding to some clause $B \Rightarrow X_i(u)$. Since $id_{[\![\sigma]\!]}$ factors through this expression, by proposition 5.8 there is an arrow φ such that $id_{[\![\sigma]\!]} = \varphi\mathsf{eq}([\![u]\!], [\![\pi'' t]\!])[\![\pi'']\!]$. But then

$$\begin{aligned}
[\![t]\!] &= \varphi\mathsf{eq}([\![u]\!], [\![\pi'' t]\!])[\![\pi'']\!][\![t]\!] \\
&= \varphi\mathsf{eq}([\![u]\!], [\![\pi'' t]\!])[\![u]\!]
\end{aligned}$$

But $\varphi\mathsf{eq}(\llbracket u \rrbracket, \llbracket \pi'' t \rrbracket)$ has representable source and target and so, by fullness of the Yoneda embedding, is of the form $\llbracket \theta \rrbracket$ for some θ in $\mathbb{C}$. By faithfulness of the Yoneda embedding $t = \theta u$ which establishes that t is an instance of u. $\square$

Lemma 5.23 *Let G be any goal formula and $P \in \mathcal{K}$ a program, both of the same sort σ. If $\llbracket G \rrbracket^k_P$ is an isomorphism, then $P \vdash_o G$.*

Proof. We proceed by induction on the structure of G and on k.

G atomic If $k = 0$, G must be the identity goal formula $\top_\sigma$, since $\llbracket X_i(t) \rrbracket^0_P = \mathbf{0} \to \llbracket \sigma \rrbracket$ cannot be an isomorphism. But the state $\langle P \mid \top_\sigma \rangle$ is already an SLD deduction of G from P.

Now suppose the result holds for all $k < n$, and all programs $P \in \mathcal{K}$.

Suppose $G = X_i(t)$ for some predicate token X_i and term t. Then

$$\llbracket X_i(t) \rrbracket^n_P = \bigcup_{\substack{\theta tm_{cl} = \theta\pi t \\ (\pi, cl) \in \kappa(P), hd_{cl} = X_i}} \{\mathsf{Im}_{\llbracket \pi \rrbracket} \mathsf{Im}_{\llbracket \theta \rrbracket} \llbracket tl_{cl}\theta \rrbracket^{n-1}_{P\theta\pi}\}$$

which is nonempty since by lemma 5.22 some clause of P admits $X_i(t)$. Thus by hypothesis, the union on the right hand side is an iso. By coprimeness of representables in the Yoneda embedding **one** of the members $\mathsf{Im}_{\llbracket \pi \rrbracket} \mathsf{Im}_{\llbracket \theta \rrbracket} \llbracket tl_{cl}\theta \rrbracket^{n-1}_{P\theta\pi}$ of the union is an iso, corresponding to some clause $cl \in \kappa(P)$ such that $hd_{cl} = X_i$ and $\theta tm_{cl} = \theta\pi t$.

By proposition 5.8 there is an arrow φ such that $id_\sigma = \varphi\theta\pi$ and $\llbracket \varphi \rrbracket$ factors through $\llbracket tl_{cl}\theta \rrbracket^{n-1}_{P\theta\pi}$. Letting $\gamma = \varphi\theta$ we have that $\llbracket tl_{cl}\gamma \rrbracket^{n-1}_{P\gamma\pi}$ is an iso, with $\gamma tm_{cl} = \gamma\pi t$. By the induction hypothesis, there is a reduction sequence

$$\langle P\gamma\pi \mid tl_{cl}\gamma \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL.$$

But then

$$\langle P \mid X_i(t) \rangle \overset{\gamma\pi, (tl_{cl} \Rightarrow X_i(tm_{cl}))}{\rightsquigarrow} \langle P\gamma\pi \mid tl_{cl}\gamma \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

is an SLD proof with computed answer substitution the identity, and so and $P \vdash_o X_i(t)$.

G = $\mathbf{A}_1 \wedge \mathbf{A}_2$ Then $\llbracket A_1 \wedge A_2 \rrbracket^n_P = \llbracket A_1 \rrbracket^n_P \cap \llbracket A_2 \rrbracket^n_P$ is an isomorphism iff both $\llbracket A_1 \rrbracket^n_P$ and $\llbracket A_2 \rrbracket^n_P$ are isomorphisms. But then by the induction hypothesis, $P \vdash_o A_1$ and $P \vdash_o A_2$. Thus there are resolution sequences

$$\langle P \mid A_1 \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

and

$$\langle P \mid A_2 \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

which may be combined to give

$$\langle P \mid A_1 \wedge A_2 \rangle \rightsquigarrow \langle P \mid A_1 \rangle \& \langle P \mid A_2 \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

Hence $P \vdash_o A_1 \wedge A_2$.

G = A$_1 \vee$ **A**$_2$ Then $[\![A_1 \vee A_2]\!]^n_P = [\![A_1]\!]^n_P \cup [\![A_2]\!]^n_P$ is an isomorphism iff either $[\![A_1]\!]^n_P$ or $[\![A_2]\!]^n_P$ are isomorphisms. But then by the induction hypothesis, either $P \vdash_o A_1$ or $P \vdash_o A_2$. Thus there is a resolution sequence

$$\langle P \mid A_i \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

from which we have

$$\langle P \mid A_1 \vee A_2 \rangle \rightsquigarrow \langle P \mid A_i \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

And hence $P \vdash_o A_1 \vee A_2$.

G = C $\Rightarrow$ **D** Then $[\![C \Rightarrow D]\!]^n_P = [\![D]\!]^n_{P \cup \{C\}}$ is an isomorphism. But then by the induction hypothesis, $P \cup \{C\} \vdash_o D$. Thus there is a resolution sequence

$$\langle P \cup \{C\} \mid D \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

from which we have

$$\langle P \mid C \Rightarrow D \rangle \rightsquigarrow \langle P \cup \{C\} \mid D \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

And hence $P \vdash_o C \Rightarrow D$.

G = $\exists_{x:\alpha} A$ Suppose that $[\![\exists_{x:\alpha} A]\!]^n_P = \mathsf{Im}_{[\![\pi]\!]} [\![A]\!]^n_{P(\pi)}$ is entire, i.e. equal to $\mathbb{C}(\_\_, \sigma)$. Then $\mathsf{Im}_{[\![\pi]\!]} [\![A]\!]^n_{P(\pi)}(\sigma) = \mathbb{C}(\sigma, \sigma)$, so $id_\sigma \in (\mathsf{Im}_{[\![\pi]\!]} [\![A]\!]^n_{P(\pi)})(\sigma) = \mathsf{Im}^{set}_{[\![\pi]\!]}([\![A]\!]^n_{P(\pi)}(\sigma))$. Thus there is an arrow t in $\mathbb{C}(\sigma, \alpha)$ with $\langle t, id_\alpha \rangle \in [\![A]\!]^n_{P(\pi)}(\sigma)$. It follows easily that $([\![\langle t, id_\alpha \rangle]\!])^\# [\![A]\!]^n_{P(\pi)}$, or, equivalently (lemma 5.5) $[\![A(\langle t, id_\alpha \rangle)]\!]^n_P$ is an isomorphism. By the induction hypothesis, there is a resolution proof

$$\langle P \mid A(\langle t, id_\alpha \rangle) \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

which means, using lemma 4.10, that the following is also a resolution proof

$$\langle P \mid \exists_{x:\alpha} A \rangle \overset{\pi}{\rightsquigarrow} \langle P\pi \mid A \rangle \rightsquigarrow \cdots \rightsquigarrow^{\langle t, id_\alpha \rangle} \rightsquigarrow NULL.$$

Thus we have $P \vdash_o \exists_{x:\alpha} A$.

□

Theorem 5.24 (Operational Completeness) *Let G be a goal formula and P a program over the signature $\mathbb{C}[X]$, both of sort σ. Then, for any object ρ in $\mathbb{C}$*

$$\rho \xrightarrow{t} \sigma \in [\![G]\!]^*_P(\rho) \qquad \textit{iff} \qquad P(t) \vdash_o G(t).$$

Proof. An immediate consequence of the Yoneda lemma ([33, 23]) is the equivalence of the following two conditions: for G a formula diagram of sort σ and $\rho \xrightarrow{t} \sigma$ in $\mathbb{C}$,

1. $t \in [\![G]\!]^*_P(\rho)$.
2. $Hom_{\mathbb{C}}(\_, \rho) \xrightarrow{[\![t]\!]} Hom_{\mathbb{C}}(\_, \sigma)$ factors through $[\![G]\!]^*_P$,

where, we recall, $[\![t]\!]$ is the natural transformation $Hom_{\mathbb{C}}(\_, t)$ whose action

$$[\![t]\!](\mu) : Hom_{\mathbb{C}}(\mu, \rho) \longrightarrow Hom_{\mathbb{C}}(\mu, \sigma)$$

at each object μ of $\mathbb{C}$ is given by $[\![t]\!](\mu)(\theta) = \theta t$.

Hence $t \in [\![G]\!]^*_P(\rho)$ iff $[\![t]\!]$ factors through $[\![G]\!]^*_P$ iff $[\![G(t)]\!]^*_{P(t)}$ is an isomorphism. By lemmas 5.20,5.23 and soundness (lemmas 5.4,5.18) this holds iff $P(t) \vdash_o G(t)$. □

6 Topos Semantics

Declarative programming is so called because the text of a program has an independent mathematical meaning which coincides, in a suitably defined way, with its operational meaning. The programming languages introduced in this paper consist of formula diagrams over a finite product category. Their operational meaning and resolution proof theory has been the subject of the preceding sections. We now define an analogue of the (first-order, intuitionistic) sequent calculus for such formula diagrams, and prove it sound and complete for conventional topos semantics. We then show that in the case of Weak Hereditarily Harrop programs and goals the notions of operational entailment and topos entailment coincide.

Topoi are categorical generalizations of the category of sets with internal operations that interpret logical connectives in a manner that generalizes their meaning in set-theoretic models. We begin by recalling the definitions of logical connectives in a topos. We then consider a restricted class of topos-theoretic models that generalizes the notion of Herbrand models, or term models, to models over a category, so as to give a Henkin-style proof of completeness. For more background on Topos theory and its logic the reader should consult e.g. [23, 34, 33].

Given some syntactic class of formulas $\mathcal{F}$ and a topos $\mathbb{E}$, an $\mathbb{E}$-interpretation $\{\![\,]\!\} : \mathcal{F} \longrightarrow \mathbb{E}$ maps sorts in $\mathcal{F}$ to objects and atomic formulas in $\mathcal{F}$ to subobjects (monics) of the appropriate object in $\mathbb{E}$.

Definition 6.1 *Let $\mathbb{A}$ be a finite product category, $\mathbb{A}[X]$ a categorical signature and let $\mathbb{E}$ be a topos. A topos interpretation of $\mathbb{A}$-formulas (an $\mathbb{A}$-interpretation) into $\mathbb{E}$ is a cartesian (pullback and terminator preserving) functor*

$$\{\![\,]\!\} : \mathbb{A}[X] \longrightarrow \mathbb{E}$$

whose value $\{\![X_i]\!\}$ on monics $X_i \in X$ of sort σ is a monic whose target is $\{\![\sigma]\!\}$.

A topos interpretation extends to formula diagrams over the signature $\mathbb{A}[X]$ via the conventional topos-theoretic logical operators $\{\cap, \cup, \supset\}$ on the subobject Heyting Algebra as follows:

- $\{\![A(t)]\!\} = (\{\![t]\!\})^{\#}(\{\![A]\!\})$
- $\{\![A_1 \wedge A_2]\!\} = \{\![A_1]\!\} \cap \{\![A_2]\!\}$
- $\{\![A_1 \vee A_2]\!\} = \{\![A_1]\!\} \cup \{\![A_2]\!\}$
- $\{\![A_1 \Rightarrow A_2]\!\} = \{\![A_1]\!\} \supset \{\![A_2]\!\}$
- $\{\![\forall_{x:\alpha} A]\!\} = \forall_\pi(\{\![A]\!\})$
- $\{\![\exists_{x:\alpha} A]\!\} = \exists_\pi(\{\![A]\!\})$

where $\forall_\pi$, $\exists_\pi$ are the right and left adjoints of the pullback functor $(\pi)^{\#} : Sub(\{\![\sigma]\!\}) \longrightarrow Sub(\{\![\sigma]\!\} \times \{\![\alpha]\!\})$ induced by the projection $\sigma \times \alpha \xrightarrow{\pi} \sigma$ in $\mathbb{A}$, and σ is the sort of the formulas $\forall_{x:\alpha} A$ and $\exists_{x:\alpha} A$.

Definition 6.2 *A formula diagram A is* **true** *in the interpretation if the subobject $\{\![A]\!\}$ is an isomorphism.*

We will now introduce a logic for $\mathbb{A}$-interpretations, that is to say a sequent calculus for first-order minimal intuitionistic logic for formula diagrams over a finite-product category.

6.1 First-order Logic over a Finite Product Category

In this section we define a categorical logic over extensions of a finite product τ-category $\mathbb{C}$. By an *indeterminate*, or a *generic constant* of sort σ over $\mathbb{C}$ we mean a new arrow $\mathbf{1} \longrightarrow \sigma$.

We say that a categorical signature $\mathbb{D}$ *extends* $\mathbb{C}$ if it is a category $\mathbb{C}(Y)$ obtained by freely adjoining a possibly infinite set Y of indeterminates to $\mathbb{C}$. This well-known construction is covered in e.g. [33].

6.1.1 Expanding the signature

Given a finite product category $\mathbb{C}$, we consider a category of *polynomial categories*, or *Henkin extensions* of $\mathbb{C}$, created by adding fresh constants drawn from a set $W_{\mathbb{C}}$ of witnesses (countably many of each sort) to $\mathbb{C}$. The collection of Henkin extensions $\mathbb{C}(Y)$ by a finite set $Y \subset W_{\mathbb{C}}$, together with the faithful, finite-product preserving functors $\iota_{1,2} : \mathbb{C}(Y_1) \longrightarrow \mathbb{C}(Y_2)$ obtained from the universal mapping property for polynomial categories, has as a colimit the category $\mathbb{C}^* = \mathbb{C}(W_{\mathbb{C}})$ which is the *full Henkin extension.* An alternative treatment to Henkin completeness using slice categories to add generic arrows and using capitalization is given in [23].

Definition 6.3 *Let $\mathbb{C}$ be a small finite product category, Y a set disjoint from the collection of arrows in $\mathbb{C}$, each member of which is associated with an object of $\mathbb{C}$, (which we will call its sort). Then the category $\mathbb{C}(Y)$ is obtained by* **freely adjoining generic constants** *Y to $\mathbb{C}$.*

One can easily show that $\mathbb{C}(Y)$ has a terminator, finite products, and a τ-structure. The definitions and arguments are similar to those given in e.g. [33], where the following is also shown.

Lemma 6.4 *Let $\mathbb{C}$ and Y be as in the preceding definition. Then there is a faithful finite product functor $\iota_Y : \mathbb{C} \longrightarrow \mathbb{C}(Y)$. Let $\mathbb{C} \xrightarrow{F} \mathbb{B}$ be a finite-product functor and $B = \{\mathbf{1} \xrightarrow{b_y} A_y : y \in Y\}$ a set of arrows in $\mathbb{C}$ in one to one correspondence with Y. Then there is a unique finite-product functor $\Psi_B : \mathbb{C}(Y) \longrightarrow \mathbb{B}$ with $\Psi_B([y]) = b_y$ and $\iota\Psi_B = F$.*

Corollary 6.5 *If $Y_1 \subset Y_2 \subset Y$ there is a unique faithful finite product functor $\mathbb{C}(Y_1) \xrightarrow{\iota_{1,2}} \mathbb{C}(Y_2)$.*

Proof. We let $\mathbb{C} \xrightarrow{F} \mathbb{B}$ be $\mathbb{C} \xrightarrow{\iota_{Y_2}} \mathbb{C}(Y_2)$ in the preceding lemma. □

Definition 6.6 *Let $\mathbb{C}$ be a small finite product category. The Henkin set $W_{\mathbb{C}}$ for $\mathbb{C}$ is the set $\{(\sigma, i) : \sigma \in |\mathbb{C}|$ and $i \in \mathbb{N}\}$. The category of* **finite generic extensions** *$[\mathbb{C}]$ has, for*

objects: the categories $\{\mathbb{C}(Y) : Y$ a finite subset of $W_{\mathbb{C}}\}$, together with the "point at infinity" $\mathbb{C}(W_{\mathbb{C}})$ and

arrows: the collection $\{\mathbb{C}(Y_1) \xrightarrow{\iota_{Y_1,Y_2}} \mathbb{C}(Y_2) : \emptyset \subseteq Y_1 \subseteq Y_2 \subset W_{\mathbb{C}}\}$ and the arrows to the point at infinity $\{\mathbb{C}(Y) \xrightarrow{\iota^*_Y} \; : Y \subseteq W_{\mathbb{C}}\}$

Let $X_1, \ldots, X_n$ be a sequence of generic predicate tokens of sorts $\boldsymbol{b} = b_1, \ldots, b_n$ from $\mathbb{C}$. Recall that the canonical, or generic formula diagrams $\mathcal{F}(\mathbb{C}[X_1, \ldots, X_n])$ of $\boldsymbol{b}$-signature $\mathbb{C}$ (definition 3.9) are the formula diagrams built up using generic predicate tokens of sorts $\boldsymbol{b}$. We can now define generic formula diagrams for the same generic predicate tokens over each category in $[\mathbb{C}]$, since the maps ι between objects $\mathbb{C}_1$ and $\mathbb{C}_2$ in $[\mathbb{C}]$induce maps $\hat{\iota}$ between the sets of formula diagrams over these objects. To see this one first shows that ι induces a faithful FP functor from $\mathbb{C}_1[X_1, \ldots, X_n]$ to $\mathbb{C}_2[X_1, \ldots, X_n]$ which sends objects (σ, S) to $(\sigma, \iota S)$ and arrows between objects (which live in $\mathbb{C}$) to themselves. This functor lifts naturally to a map between the classes of formula diagrams built over each.

We adopt the following coercion conventions for speaking about such formula diagrams. If $A \in \mathcal{F}(\mathbb{C}(Y_1))$ and $Y_1 \subset Y_2$ then we may denote by the same symbol A the formula diagram $\hat{\iota}_{1,2}(A)$ in $\mathcal{F}(\mathbb{C}(Y_2))$ when the context makes clear that we are over the category $\mathbb{C}(Y_2)$. Similarly we will speak of arrows t in $\mathbb{C}(Y_1)$ as arrows of $\mathbb{C}(Y_2)$.

Definition 6.7 (The language of a formula) *Suppose A is a formula diagram over $\mathbb{C}^*$. Then there is a unique minimal $Y \subseteq W_{\mathbb{C}}$ with some $A' \in \mathbb{C}(Y)$ and $A = \iota^*_Y(A')$. We refer to $\mathbb{C}(Y)$ as the* signature or language *of A.*

Let the map $\chi : \mathcal{F}(\mathbb{C}(W_{\mathbb{C}})[X]) \times |\mathbb{C}| \longrightarrow \mathbb{C}^*$ be defined by letting $\chi(A, \sigma)$ be the least (σ, i) in $\mathbb{C}^*$ not in the language of A.

The map χ will be used to provide fresh generic constants for universally quantified formulas.

6.1.2 A Sequent Calculus

We now introduce our sequent calculus for reasoning with formula diagrams over the categories in $[\mathbb{C}]$.

Definition 6.8 *Let $\mathbb{C}$ be a finite product τ-category and X a set of predicate letters with sorts in $\mathbb{C}$. A (categorical) $\mathbb{D}$-sequent is an ordered triple $(\mathbb{D}, \Gamma, A)$, where $\mathbb{D}$ is a language extending $\mathbb{C}$, A is a formula diagram over $\mathbb{D}$, and Γ is a $\mathbb{D}$-theory: a set of formula diagrams* of the same sort *over $\mathbb{D}$. Sequents are written $\mathbb{D}, \Gamma \gg A$.*

Definition 6.9 *A sequent $\mathbb{D}, \Gamma \gg A$ is said to be* **initial** *if $A \in \Gamma$ and A is an atom. A* $\mathbb{D}$**-proof** *is either a $\mathbb{D}$-initial sequent or a tree with $\mathbb{D}$-initial sequents at its leaves, built from smaller proofs by the use of the* **rules of inference** *in the following table.*

$$\frac{\mathbb{D}, B, C, \Gamma \gg A}{\mathbb{D}, B \wedge C, \Gamma \gg A}\wedge\text{-}L \qquad \frac{\mathbb{D}, \Gamma \gg B \qquad \mathbb{D}, \Gamma \gg C}{\mathbb{D}, \Gamma \gg B \wedge C}\wedge\text{-}R$$

$$\frac{\mathbb{D}, \Gamma, B \gg A \qquad \mathbb{D}, \Gamma, C \gg A}{\mathbb{D}, \Gamma, B \vee C \gg A}\vee\text{-}L \qquad \frac{\mathbb{D}, \Gamma \gg B}{\mathbb{D}, \Gamma \gg B \vee C}\vee\text{-}R \quad \frac{\mathbb{D}, \Gamma \gg C}{\mathbb{D}, \Gamma \gg B \vee C}\vee\text{-}R$$

$$\frac{\mathbb{D}, \Gamma \gg B \qquad \mathbb{D}, \Gamma, C \gg A}{\mathbb{D}, \Gamma, B \Rightarrow C \gg A}\Rightarrow\text{-}L \qquad \frac{\mathbb{D}, \Gamma, B \gg C}{\mathbb{D}, \Gamma \gg B \Rightarrow C}\Rightarrow\text{-}R$$

$$\frac{\mathbb{D}, \Gamma, P(\langle id_\sigma, t\rangle) \gg A}{\mathbb{D}, \Gamma, \forall_{x:\alpha}P \gg A}\forall\text{-}L \qquad \frac{\mathbb{D}(a), \Gamma \gg P(\langle id_\sigma, !_\sigma a\rangle)}{\mathbb{D}, \Gamma \gg \forall_{x:\alpha}P}\forall\text{-}R*$$

$$\frac{\mathbb{D}(a), \Gamma, P(\langle id, !_\sigma a\rangle) \gg A}{\mathbb{D}, \Gamma, \exists_{x:\alpha}P \gg A}\exists\text{-}L* \qquad \frac{\mathbb{D}, \Gamma \gg P(\langle id_\sigma, t\rangle)}{\mathbb{D}, \Gamma \gg \exists_{x:\alpha}P}\exists\text{-}R$$

where $(\sigma \xrightarrow{t} \alpha) \in \mathbb{D}$ in $\forall$-L and $\exists$-R, and $!_\sigma a$ means the composition

$$\sigma \xrightarrow{!_\sigma} \mathbf{1} \xrightarrow{a} \sigma.$$

Rules marked with a $*$ have the side condition that 'a' is a Henkin constant which must not be in the language (e.g. $\chi(D,\sigma)$, see definition 6.7) of any formula diagrams in the conclusion of the rule. The starred rules yield quantified formulas from premisses involving generic constants. The predicates are instantiated at $\langle id_\sigma, !_\sigma a\rangle$ because the sorts of the unquantified predicates are product sorts.

We write $\mathbb{D}, \Gamma \vdash_c B$ to mean there is a $\mathbb{D}$-proof whose final sequent is $\mathbb{D}, \Gamma \gg B$.

A closed *formula diagram means a diagram of sort* $\mathbf{1}$. *A closed theory means a set of closed formula diagrams. If Γ is closed, then, in a τ-category$\langle id, !_\sigma a\rangle = a$.*

The following facts, which are easily checked, will prove useful:

Lemma 6.10 *If the language of the theory Γ and the formula diagram A is $\mathbb{D}$, and $\mathbb{D}'$ extends $\mathbb{D}$ in $[\mathbb{C}]$, and*

$$\mathbb{D}', \Gamma \vdash_c A$$

then

$$\mathbb{D}, \Gamma \vdash_c A$$

Lemma 6.11 *Sequents are preserved under pullback: if $\mathbb{D}, \Gamma \vdash_c A$ then $\mathbb{D}, \Gamma(t) \vdash_c A(t)$ for t of the appropriate sort in $\mathbb{D}$, where $\Gamma(t)$ means the set $\{\gamma(t) : \gamma \in \Gamma\}$*

The proof is an easy induction.

Lemma 6.12 (Cut Elimination) *The cut rule is admissible in the $\mathbb{D}$-sequent calculus defined above: if a sequent is derivable using the rule*

$$\frac{\mathbb{D}, \Gamma \gg B \qquad \mathbb{D}, \Gamma', B \gg A}{\mathbb{D}, \Gamma, \Gamma' \gg A} CUT$$

it has a cut-free proof.

The proof is identical to that for cut elimination in the standard sequent calculus, see e.g. [25].

Theorem 6.13 (Soundness) *If $\mathbb{D}, \Gamma \vdash_c A$ then A is true in every $\mathbb{D}$-interpretation of Γ (i.e. in which every formula diagram in Γ is true).*

The proof is identical to the standard proof of soundness of topos semantics for first-order logic.

We will now show these rules are complete for $\mathbb{A}$-interpretations. We will use a canonical interpretation, a topos-theoretic analogue of the Henkin model familiar from logic. To do so, we make use of the following notions and facts from conventional logic which extend easily to categorical logic.

Definition 6.14 *A set Γ of formula diagrams over $\mathbb{D}[X]$ is said to be a* **prime theory** *over $\mathbb{D}$ (or $\mathbb{D}$-saturated theory) if it has the* **disjunction** *and* **existence** *properties:*

$$\mathbb{D}, \Gamma \vdash_c A \vee B \;\Rightarrow\; \mathbb{D}, \Gamma \vdash_c A \;\text{ or }\; \mathbb{D}, \Gamma \vdash_c B$$

$$\mathbb{D}, \Gamma \vdash_c \exists_{x:\alpha} A \;\Rightarrow\; \mathbb{D}, \Gamma \vdash_c A(\langle id_\sigma, u\rangle) \qquad \text{for some } \sigma \xrightarrow{u} \alpha \text{ in } \mathbb{D}.$$

Lemma 6.15 (Lindenbaum) *If $\mathbb{D}, \Gamma \not\vdash_c A$ then, for any $\mathbb{D}'$ extending $\mathbb{D}$, there is a prime theory Δ over $\mathbb{D}'$ such that $\Delta \supseteq \Gamma$ and*

$$\mathbb{D}', \Delta \not\vdash_c A$$

i.e. every consistent theory has a consistent prime extension.

The proof is the same as the standard one for logic: the prime extension is constructed in stages. See e.g. [66].

6.2 Henkin Interpretations

Let $\{V_i : i \in \mathbb{N}\}$ be a countable collection of disjoint sets of *witnesses* (see e.g. section 6.1.1) containing countably many new constants of each sort in $\mathbb{C}$. Let $W_o = \phi$, and for $n \geq 1$, let $W_n = V_1 \cup \cdots \cup V_n$ and $W = \bigcup W_n$. Assume that for each sort σ, the (countable) set W_σ of constants in W of sort σ is well-ordered, so that for any proper subset Y of W_σ we can speak of the least member of W_σ not in Y. Let $\mathcal{B}$ be the collection of polynomial categories $\mathbb{C}(W_n)$ or $\mathbb{C}(W)$ extending $\mathbb{C}$ by freely adjoining the constants in W_n or W, as in section 6.1.1. To simplify notation, we will use the symbols $\mathbb{B}_o, \mathbb{B}, \mathbb{B}', \mathbb{B}'', \ldots$ to range over the collection of categories in $\mathcal{B}$. Note that all members of $\mathcal{B}$ have the same collection of objects and there are canonical faithful finite product preserving functors $\iota_{\mathbb{B}} : \mathbb{B} \to \mathbb{B}'$ whenever $\mathbb{B}'$ is an extension of $\mathbb{B}$.

Definition 6.16 *If Γ is a theory (a set of formula diagrams), we say Γ' extends Γ if $\Gamma \subseteq \Gamma'$.*

Let $\mathbb{B}$ be a category in $\mathcal{B}$, and Γ a **closed** *prime theory over $\mathbb{B}$. $\mathcal{J}(\mathbb{B}, \Gamma)$ is the poset of signature-theory pairs $(\mathbb{B}', \Gamma')$ extending $(\mathbb{B}, \Gamma)$.*

Definition 6.17 *Let $\mathbb{B}$ be a category in $\mathcal{B}$, and Γ a closed prime theory over $\mathbb{B}$. Then $\mathbb{J}(\mathbb{B}, \Gamma)$ is the following category:*

- *<u>objects:</u> triples $(\rho, \mathbb{B}', \Gamma')$ with ρ an object in $\mathbb{C}$, and $(\mathbb{B}', \Gamma')$ in $\mathcal{J}(\mathbb{B}, \Gamma)$.*
- *<u>arrows:</u> $(\rho, \mathbb{B}', \Gamma') \xrightarrow{t} (\rho', \mathbb{B}'', \Gamma'')$ where $(\mathbb{B}', \Gamma') \geq (\mathbb{B}'', \Gamma'')$ and $\rho \xrightarrow{t} \rho'$ is in $\mathbb{B}'(\rho, \rho')$.*

Definition 6.18 *Let $\mathbb{B}_o$ be a category in $\mathcal{B}$ and Γ_o a prime theory over $\mathbb{B}_o$. A* **Henkin** *$\mathbb{B}_o$-interpretation of Γ_o is a topos interpretation into the Yoneda presheaf topos for $\mathbb{J}(\mathbb{B}_o, \Gamma_o)$ given by a functor*

$$\{\![\,]\!\}_{\mathbb{B}_o} : \mathbb{B}_o[X] \longrightarrow \mathsf{Set}^{\mathbb{J}(\mathbb{B}_o, \Gamma_o)^o}$$

mapping monics $X_i(t)$ to monics, and satisfying:

$$\{\![\sigma]\!\}_{\mathbb{B}_o}(\rho, \mathbb{B}, \Gamma) = Hom_{\mathbb{B}}(\rho, \sigma)$$

$$\{\![\sigma]\!\}_{\mathbb{B}_o}((\rho, \mathbb{B}, \Gamma) \xrightarrow{t} (\rho', \mathbb{B}', \Gamma'))(\theta) = t(\iota_{\mathbb{B}'}\theta)$$

and which, on arrows $\sigma \xrightarrow{u} \sigma'$ in $\mathbb{B}_o$, $\{\![u]\!\}_{\mathbb{B}_o} : \{\![\sigma]\!\}_{\mathbb{B}_o} \longrightarrow \{\![\sigma']\!\}_{\mathbb{B}_o}$ is given by

$$\{\![u]\!\}_{\mathbb{B}_o}(\rho, \mathbb{B}, \Gamma)(\theta) = \theta(\iota_{\mathbb{B}_o} u).$$

In other words $\{\![\;]\!\}_{\mathbb{B}_o}$ behaves similarly to a "varying Yoneda embedding" indexed over $\mathcal{B}$. We will not take the trouble to formalize the obvious indexed category structure here.

When the ambient signature of $\mathbb{B}_o$ is clear from context we will also omit reference to it, and write $\{\![B]\!\}$ instead of $\{\![B]\!\}_{\mathbb{B}_o}$. Note also that a Henkin interpretation is completely determined by an assignment of monics in $\mathsf{Set}^{\mathbb{J}(\mathbb{B}_o, \Gamma_o)^o}$ with appropriate targets, to the monics X_i in X.

Remark 6.19 We have defined a **family** of Henkin interpretations, one for each signature-theory pair $(\mathbb{B}_o, \Gamma_o)$. The reader can check that if $(\mathbb{B}', \Gamma')$ extends $(\mathbb{B}, \Gamma)$ then $\mathcal{J}(\mathbb{B}', \Gamma') \subset \mathcal{J}(\mathbb{B}, \Gamma)$ and there is an inclusion functor $\mathbb{J}(\mathbb{B}', \Gamma') \subset \mathbb{J}(\mathbb{B}, \Gamma)$ and an induced functor

$$\mathsf{Set}^{\mathbb{J}(\mathbb{B}, \Gamma)^o} \rightarrowtail \mathsf{Set}^{\mathbb{J}(\mathbb{B}', \Gamma')^o}.$$

Hence, whenever both sides are defined, (i.e. A is a formula diagram over a signature extended by $\mathbb{B}'$and $\mathbb{B}''$, and $\mathbb{B}'''$extends both of these signatures, and Γ''' extends the base theory)

$$\{\![A]\!\}^*_{\mathbb{B}'}(\rho, \mathbb{B}''', \Gamma''') = \{\![A]\!\}^*_{\mathbb{B}''}(\rho, \mathbb{B}''', \Gamma''').$$

6.3 The Canonical Henkin Interpretation

We now give a a completeness theorem for the generalized categorical logic (section 6.1) in the spirit of Aczel's completeness theorem for intuitionistic logic [66]. We define canonical Henkin interpretations $\{\![\,]\!\}^*$, by letting monic assignments be dictated by the proof theory.

Definition 6.20 *Let $\mathbb{C}[X]$be a categorical signature, $\mathbb{B}_o$ a Henkin extension in $\mathcal{B}$, and Γ_o a* closed *prime theory over $\mathbb{B}_o$. The canonical interpretation $\{\![\,]\!\}^*_{\mathbb{B}_o}$ for Γ_o is the Henkin interpretation induced by the following assignment of monics $X_i \in X$ of sort σ in the topos $\mathsf{Set}^{\mathbb{J}(\mathbb{B}_o,\Gamma_o)^o}$*

$$\{\![X_i]\!\}^*_{\mathbb{B}_o}(\rho, \mathbb{B}, \Gamma) := \{t : \rho \xrightarrow{t} \sigma \in \mathbb{B} \;\; \textit{and}\; \mathbb{B}, !_\rho\Gamma \vdash X_i(t)\}$$

Proposition 6.21 (Completeness of $\mathbb{C}$-logic) *Suppose A is a formula diagram of sort σ over $\mathbb{B}_o[X]$, and $\{\![\,]\!\}^*_{\mathbb{B}_o}$ the canonical Henkin interpretation of the closed prime theory Γ_o over $\mathbb{B}_o[X]$. Then, for any object $(\rho, \mathbb{B}, \Gamma)$ in $\mathbb{J}(\mathbb{B}_o, \Gamma_o)$,*

$$\{\![A]\!\}^*_{\mathbb{B}_o}(\rho, \mathbb{B}, \Gamma) = \{t : \rho \xrightarrow{t} \sigma \in \mathbb{B} \;\; \textit{and}\; \mathbb{B}, !_\rho\Gamma \vdash_c A(t)\}$$

Proof. We establish this equation by induction on the structure of the formula diagram A.
The atomic case is true by hypothesis.

<u>$A \equiv A_1 \wedge A_2$</u>: $\{\![A_1 \wedge A_2]\!\}^*(\rho, \mathbb{B}, \Gamma) = \{\![A_1]\!\}^*(\rho, \mathbb{B}, \Gamma) \cap \{\![A_2]\!\}^*(\rho, \mathbb{B}, \Gamma)$ which, by inductive hypothesis and the $\wedge$-R and $\wedge$-L rules, is $\{t : \rho \xrightarrow{t} \sigma \in \mathbb{B}$ and $\mathbb{B}, !_\rho\Gamma \vdash_c (A_1 \wedge A_2)(t)\}$.

<u>$A \equiv A_1 \vee A_2$</u>: $\{\![A_1 \vee A_2]\!\}^*(\rho, \mathbb{B}, \Gamma) = \{\![A_1]\!\}^*(\rho, \mathbb{B}, \Gamma) \cup \{\![A_2]\!\}^*(\rho, \mathbb{B}, \Gamma) =$ $\{\rho \xrightarrow{t} \sigma : \mathbb{B}, !_\rho\Gamma \vdash_c A_1(t)$ or $\mathbb{B}, !_\rho\Gamma \vdash_c A_2(t)\}$. Since Γ satisfies the disjunction property, the latter set is equal to $\{\rho \xrightarrow{t} \sigma : \mathbb{B}, !_\rho\Gamma \vdash_c A(t)\}$.

<u>$A \equiv \exists_{x:\alpha} B$</u>: We begin showing that an easy consequence of the proof rules is the following derived rule of inference:

$$\frac{\mathbb{B}, !_\rho\Gamma \gg B(\langle t, w\rangle)}{\mathbb{B}, !_\rho\Gamma \gg (\exists_{x:\alpha} B)(t)}$$

where the quantification in the conclusion is along the projection $\rho \times \alpha \xrightarrow{\pi'} \rho$.

$$\begin{tikzcd}
 & B(t \times id_\alpha) \arrow[d, tail] & B \arrow[d, tail] \\
\rho \arrow[r, "\langle id_\rho, w\rangle"] & \rho \times \alpha \arrow[r, "t \times id_\alpha"] \arrow[d, "\pi', \exists"] & \sigma \times \alpha \arrow[d, "\pi, \exists"'] \\
 & \rho \arrow[r, "t"'] & \sigma
\end{tikzcd}$$

By definition (2.2) $(\exists_{x:\alpha}B)(t) = \exists_{x:\alpha}(B(t \times id_\alpha))$ and $\langle id_\rho, w\rangle(t \times id_\alpha) = \langle t, w\rangle$. Thus, $\mathbb{B}\,, !_\rho\Gamma \gg B(\langle t, w\rangle)$ iff $\mathbb{B}\,, !_\rho\Gamma \gg B(t \times id_\alpha)(\langle id_\rho, w\rangle)$ from which, by $\exists$-R, we infer $\mathbb{B}\,, !_\rho\Gamma \gg \exists_{x:\alpha}(B(t \times id_\alpha))$, which is precisely $\mathbb{B}\,, !_\rho\Gamma \gg (\exists_{x:\alpha}B)(t)$.

Now suppose $\rho \xrightarrow{u_o} \sigma$ is an arrow in $\mathbb{B}$ and

$$u_o \in \{\![\exists_{x:\alpha}B]\!\}^*(\rho, \mathbb{B}\,, \Gamma).$$

Images are evaluated pointwise, so this yields $u_o \in \exists_\pi(\{\![B]\!\}^*)(\rho, \mathbb{B}\,, \Gamma)$. Hence, for some $\rho \xrightarrow{u_1} \alpha$ we have $\langle u_o, u_1\rangle \in \{\![B]\!\}^*(\rho, \mathbb{B}\,, \Gamma)$. By induction hypothesis $\mathbb{B}\,, !_\rho\Gamma \vdash_c B(\langle u_o, u_1\rangle)$, hence we may infer $\mathbb{B}\,, !_\rho\Gamma \vdash_c (\exists_{x:\alpha}B)(u_o)$. We have shown

$$\{\![\exists_{x:\alpha}B]\!\}^*(\rho, \mathbb{B}\,, \Gamma) \subseteq \{\rho \xrightarrow{t} \sigma : \mathbb{B}\,, !_\rho\Gamma \vdash_c (\exists_{x:\alpha}B)(t)\}.$$

For the other direction, suppose $\mathbb{B}\,, !_\rho\Gamma \vdash_c (\exists_{x:\alpha}B)(t)$, that is to say,

$$\mathbb{B}\,, !_\rho\Gamma \vdash_c \exists_{x:\alpha}(B(t \times id_\alpha)).$$

Since Γ is a prime theory, there is a $\rho \xrightarrow{v} \alpha$ such that

$$\mathbb{B}\,, !_\rho\Gamma \vdash_c (B(t \times id_\alpha))(\langle id_\rho, v\rangle).$$

But $(B(t \times id_\alpha))(\langle id_\rho, v\rangle) = B(\langle id_\rho, v\rangle(t \times id_\alpha)) = B(\langle t, v\rangle)$. So, by induction hypothesis we have $\langle t, v\rangle \in \{\![B]\!\}^*(\rho, \mathbb{B}\,, \Gamma)$ whence $t \in \exists_\pi(\{\![B]\!\}^*)(\rho, \mathbb{B}\,, \Gamma)$, which establishes the other containment.

$\underline{A \equiv A_1 \Rightarrow A_2}$: Implication of predicates in a topos is not evaluated pointwise. $\{\![A_1 \Rightarrow A_2]\!\}^*(\rho, \mathbb{B}\,, \Gamma) = [\{\![A_1]\!\}^* \supset \{\![A_2]\!\}^*](\rho, \mathbb{B}\,, \Gamma)$, which equals

$$\{t : \text{for every } (\nu, \mathbb{B}', \Gamma') \xrightarrow{u} (\rho, \mathbb{B}\,, \Gamma) \text{ if } u(\iota_{\mathbb{B}}\, t) \in \{\![A_1]\!\}^*(\nu, \mathbb{B}', \Gamma') \text{ then } u(\iota_{\mathbb{B}}\, t) \in \{\![A_2]\!\}^*(\nu, \mathbb{B}', \Gamma')\}$$

By the induction hypothesis, this is equal to

$$\{t : \forall u, \text{if } \mathbb{B}', !\Gamma' \vdash_c A_1(ut) \text{then } \mathbb{B}', !\Gamma' \vdash_c A_2(ut)\} \tag{5}$$

We now claim this is, in fact, the set

$$\{t : \mathbb{B}\,, !\Gamma \vdash_c (A_1 \Rightarrow A_2)(t)\} \tag{6}$$

That (6)$\subseteq$(5) follows immediately from the cut rule.

So, it suffices to show that (5)$\subseteq$(6). If not, for some t in (5), $\mathbb{B}\,, !\Gamma \nvdash_c (A_1 \Rightarrow A_2)(t)$. Then, using cut, the $\forall - R$ rule and lemma 6.11, we have that $\mathbb{B}\,(a), \Gamma \nvdash_c (A_1 \Rightarrow A_2)(at)$ and so $\mathbb{B}\,(a), \Gamma, A_1(at) \nvdash_c A_2(at)$. By Lindenbaum's lemma 6.15, some prime extension $\mathbb{B}', \Gamma'$ of the premise $\mathbb{B}\,(a), \Gamma, A_1(at)$ exists for which $\mathbb{B}', \Gamma' \nvdash_c A_2(at)$ despite the fact that $\mathbb{B}', \Gamma' \vdash_c A_1(at)$ contradicting the definition of t.

$\underline{A \equiv \forall_{x:\alpha} B}$: $\{\![\forall_{x:\alpha} B]\!\}^* = \forall_\pi \{\![B]\!\}^*$. $\forall_\pi$, the right adjoint of pullback along $\sigma \times \alpha \xrightarrow{\pi} \sigma$, is not evaluated pointwise:

$$\begin{aligned}\forall_\pi \{\![B]\!\}^*(\rho, \mathbb{B}, \Gamma) &= \\ &\{t : \text{for every}(\nu, \mathbb{B}', \Gamma') \xrightarrow{u} (\rho, \mathbb{B}, \Gamma),\ \{\langle ut, v\rangle : v \in Hom_{\mathbb{B}'}(\nu, \alpha)\} \subseteq \{\![B]\!\}^*(\nu, \mathbb{B}', \Gamma')\}\end{aligned}$$

Suppose $\mathbb{B}, !_\rho\Gamma \vdash_c \forall_{x:\alpha}(B(t \times id_\alpha))$. Then, using cut and lemma 6.11, we have, for every $(\nu, \mathbb{B}', \Gamma') \xrightarrow{u} (\rho, \mathbb{B}, \Gamma)$ that $\mathbb{B}', !_\nu\Gamma' \vdash_c (B(ut \times id_\alpha))\langle id, v\rangle$ for all $v \in Hom_{\mathbb{B}'}(\nu, \alpha)$. Hence, by the induction hypothesis, $\langle ut, v\rangle \in \{\![B]\!\}^*(\nu, \mathbb{B}', \Gamma')$, and so by definition, $t \in \forall_\pi \{\![B]\!\}^*(\rho, \mathbb{B}, \Gamma)$.

Conversely, suppose $t \in \forall_\pi \{\![B]\!\}^*(\rho, \mathbb{B}, \Gamma)$, but $\mathbb{B}, \Gamma \not\vdash_c (\forall_{x:\alpha} B)(t)$. Then for any $\mathbb{B}'$ extending $\mathbb{B}$, and any generic constant c of sort α in $\mathbb{B}'$but not in $\mathbb{B}$, we have $\mathbb{B}', \Gamma \not\vdash_c B(\langle t, !c\rangle)$

By Lindenbaum's lemma, for some $\mathbb{B}'$-prime theory Γ' extending Γ we have $\mathbb{B}', \Gamma' \not\vdash_c B(\langle t, !c\rangle)$. Hence, by the induction hypothesis, $(\langle t, !c\rangle) \notin \{\![B]\!\}^*(\rho, \mathbb{B}', \Gamma')$ so $t \notin \forall_\pi \{\![B]\!\}^*(\rho, \mathbb{B}, \Gamma)$. □

Corollary 6.22 *Suppose A is a formula diagram of sort σ over $\mathbb{B}_o[X]$. Then $\{\![A]\!\}^*$ is an isomorphism (as a subobject of $\{\![\sigma]\!\}^*_{\mathbb{B}_o}$) if and only if $\mathbb{B}_o, \Gamma \vdash_c A$.*

The corollary follows from the fact that $\mathbb{B}_o, \Gamma \vdash_c A$ if and only if the identity morphism is in $\{\![A]\!\}^*(\sigma, \mathbb{B}_o, \Gamma)$ if and only if $\{\![A]\!\}^*$ is an isomorphism.

To show the equivalence of operational and declarative semantics when applied to programs and goals, we use the equivalence of the associated logics of SLD-proofs and $\mathbb{C}$-proofs. The proof of their equivalence is identical to that given in [46] (theorems 4 and the proof of 6), since the relevant properties, permutability of the left $\wedge, \forall, \Rightarrow$ rules with the right $\wedge, \vee \Rightarrow \forall, \exists$ rules are easily seen to hold in $\mathbb{C}$-logic as well. A direct semantic proof of the equivalence of operational and Kripke semantics is given in [17].

Theorem 6.23 *Let P_o be a closed program over $\mathbb{C}$, G a goal of sort σ over $\mathbb{C}$. For each arrow $\rho \xrightarrow{t} \sigma$ in $\mathbb{C}$*

$$t \in \{\![G]\!\}^*(\rho, \mathbb{C}, P_o) \iff t \in [\![G]\!]^*_{!_\sigma P_o}(\rho).$$

Proof. We make use of the respective completeness theorems (5.24,6.21) to reduce the claim to the just cited equivalence of resolution $\vdash_o$ and provability in the $\mathbb{C}$-sequent calculus. Suppose $t \in \{\![G]\!\}^*(\rho, \mathbb{C}, P_o)$. Then $\mathbb{C}, !_\rho P_o \vdash_c G(t)$, whence $!_\rho P_o \vdash_o G(t)$, and thus $\iota_* t \in [\![G]\!]^*_{!_\sigma P_o}(\rho)$.

To prove the converse, suppose $t \in [\![G]\!]^*_{!_\sigma P_o}(\rho)$. Then $!_\rho P_o \vdash_o G(t)$, so $\mathbb{C}, !_\rho P_o \vdash_c G(t)$. By lemma 6.10, this implies $\mathbb{C}, !_\rho P_o \vdash_c G(t)$, whence $t \in \{\![G]\!\}^*(\rho, \mathbb{C}, P_o)$. □

7 Examples and Conclusion

In this paper we have given a categorical treatment of the resolution calculus for Horn and first-order Weak Hereditarily Harrop (WHH) formulas with categorically defined constraints, sound and complete with respect to operational and declarative semantics. Our notion of categorical signature formalizes the structure of many constraint systems. For example, let $\mathcal{L}$ be a first-order language, $\mathcal{E}$ an equational theory over $\mathcal{L}$. We define two finite-product categories: $\mathbb{H}$ the empty *Lawvere algebraic theory* for $\mathcal{L}$, and $\mathbb{H}/\mathcal{E}$ the algebraic theory associated with $\mathcal{E}$, to capture pure logic programming over the Herbrand Universe in the former case, and logic programming over the equational constraints in $\mathcal{E}$ in the latter. We give a brief description here. The reader should consult e.g. [58] for a fuller treatment of algebraic theories.

Define $\mathbb{H}$, 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 $0, 1, 2, \ldots$, and *arrows* the closure under composition of the following:

- For each nonzero pair of objects m, n with $m \leq n$ all the projection maps

  $$\pi^n_m(i_1, \ldots, i_m) : n \to m,$$

  where $i_1, \ldots, i_m$ is a sequence of m elements from $\{1, \ldots, n\}$, and all the diagonal arrows $\delta : m \to n$. These arrows define precisely the opposite category of the full subcategory of Set consisting of the Von-Neumann finite ordinals

  $$\{\} \subset \{0\} \subset \{0,1\} \subset \cdots$$

- In addition, the signature contributes, for each function symbol f of arity n in $\mathcal{L}$ an arrow $\mathbf{f}$: $n \to 1$.

These arrows satisfy the equations for associativity as well as all usual identities for projections and diagonal maps.

This category has products ($n \times m = nm$) and τ-structure, as the reader can check. Furthermore there is a one-to-one correspondence between arrows from 0 to 1 in $\mathbb{H}$ and terms in the Herbrand universe over $\mathcal{L}$. The arrows from 0 to m correspond to m-tuples of terms. If we now give $\mathbb{H}$ the canonical interpretation that associates with each f in $\mathcal{L}$ the arrow $\mathbf{f}$ and let $H[X_1, \ldots, X_n]$ be the category obtained by adjoining indeterminates for each of the predicate letters in $\mathcal{L}$, with X_i of sort n if the corresponding predicate letter has arity n, then $H[X_1, \ldots, X_n]$ captures logic programming in the sense made precise in the theorem below.

Theorem 7.1 *Let P be a Horn (resp. Weak Hereditarily Harrop) program, G a goal in the corresponding language. Let X be a set of generic monics of appropriate sort, corresponding to each of the predicate letters in P and G. Then there is a Horn-SLD (resp. Weak Hereditarily Harrop) proof $P \vdash_o G$ over $\mathbb{H}$ if and only if for every topos $\mathbb{E}$ and every $\mathbb{H}$-interpretation $\{\![\,]\!\} : \mathbb{H}[X] \longrightarrow \mathbb{E}$ in which P is true we have that $\{\![G]\!\}$ is an isomorphism.*

SLD proofs over $\mathbb{H}$ are resolution proofs in our sense. All computed answer substitutions can be taken to be compositions of equalizers (mgu's). Categorical Horn clause programming over $\mathbb{H}$ is essentially the case studied in [4].

Even in the case of Horn clause programs over the Herbrand universe, our operational interpretation gives non-ground bottom-up semantics, which is more general than the usual T_P-operator.

A straightforward adaptation of the preceding construction gives the corresponding free category $\mathbb{H}_{\mathcal{L}}$ for many-sorted logic, by taking sorts as objects, with the obvious projections and diagonals. We can further modify the 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 $\mathbb{H}_{\mathcal{L}}/\mathcal{E}$ of the Herbrand category we obtain the algebraic theory associated with $\mathcal{E}$, containing the same objects as $\mathbb{H}_{\mathcal{L}}$ and congruence classes of arrows induced by $\mathcal{E}$ as morphisms.

SLD-resolution over this category corresponds to finding a computed answer with equational constraints from $\mathcal{E}$. Similarly, the following theorem is an immediate consequence of the results in this paper.

Theorem 7.2 *Let P be a Horn (resp. WHH) program, G a goal, both over the language $\mathcal{L}$. Let X be a set of generic monics of appropriate sort, corresponding to each of the predicate letters in P and G. Then there is a Horn-SLD (resp. WHH) proof $P \vdash_o G$ over $\mathbb{H}_{\mathcal{L}}/\mathcal{E}$ if and only if for every topos $\mathbb{E}$ and every $\mathbb{H}_{\mathcal{L}}/\mathcal{E}$-interpretation $\{\!\![\,]\!\!\} : \mathbb{H}_{\mathcal{L}}/\mathcal{E}[X] \longrightarrow \mathbb{E}$ in which P is true we have that $\{\!\![G]\!\!\}$ is an isomorphism.*

The notion of constraint captured in this paper is considerably more general than these examples: not all cartesian categories are algebraic theories. A precise syntactic characterization of such constraints requires a study of Cartesian Logic [22], which is the subject of a forthcoming paper.

Beyond Generic Predicates The reader may ask how results in this paper are affected by the restriction of categorical signatures to those of the form $(\mathbb{C}[X], X)$ induced by generic predicates. McGrail [42] has shown how to modify the notion of generic predicate to build in functional constraint and monad structure, which is also outlined in [37]. It would be interesting to see how these results could be extended to incorporate higher-order constraint information and e.g. objects and inheritance into logic programming.

7.1 Perspectives on Uniform proofs

In our operational semantics, we characterize the predicates defined by a logic program via a fixpoint construction using a continuous operator E which induces a map from interpretations to interpretations, incrementally defining the meaning of goal formulae. An essential component of the behavior of the interpretations so defined is that they give a search theoretic interpretation of certain critical connectives. For example, the meaning of $A \Rightarrow B$ is given by $[\![A \Rightarrow B]\!]_P = [\![B]\!]_{P,A}$, that is to say, by tearing apart formulae and interpreting them in a shifted semantics, which are only required to give meaning to basic (in our case atomic, conjunctive and disjunctive) formulas. Thus a semantic shift or *change of state* appears to be an essential feature of the uniform or search-theoretic interpretations of connectives. What guarantees correctness with respect to a "standard" interpretation is the agreement of the *limiting* operational bracket with conventional interpretation $\{\!\![\,]\!\!\}$ in a standard model (in our case, a topos). This suggests a quite general blueprint for analyzing abstract connectives, or even defining them *semantically*. One seeks identities of the form

$$[\![A * B]\!]_{\mathbb{B},P} = [\![F(A, B, P, \mathbb{B})]\!]_{G(A,B,P,\mathbb{B})},$$

where the operator G defines the interpretation shift and F yields a formula that is –in some sense– more basic than $A * B$. It would be interesting to study which operations of this kind (over more general semantic categories) corresponding to other logics (e.g. Girard's LU[24], linear logic programming languages [28, 45, 9]), and which give rise to useful operations that still agree with some notion of conventional semantics in the limiting case.

7.2 Directions for future work

This foundation opens the way to a categorical approach to modeling other extensions of logic programming [28, 51, 50, 55, 49], and connecting well-known techniques for modeling functional evaluation, non-local control, side effects (e.g. monads, see [62, 42, 47, 68, 69]), to the declarative and operational semantics of logic programs.

Our operational semantics ignores control issues (such as the selection rule) since it is defined in terms of the declarative text of the program. It would be straightforward to modify the operator E to take such control into account, giving rise to different, more computationally oriented semantics. Also, it would be interesting to capture in a categorical context the structure of other semantics (compositional, open program, unfolding, etc.), studied by Levi et. al. [8, 40, 1, 43].

In addition, an interesting follow-up to the work presented here would be a categorical analysis of the Jaffar-Lassez criteria [29] for constraint domains and of the Lassez-MacAloon [35] constraint calculi. Our categorical study should also be extended to the case of higher-order logic, L-λ, and dependent type theory. It would seem that a categorical formulation of abstract interpretation [5, 14] would yield some interesting insights as well.

Appendix

A Proofs of Theorems

Lemma A.1 (Lemma 4.8) *If* $\langle P(t) \mid G(t)\rangle \rightsquigarrow^{\theta} \rightsquigarrow \cdots \rightsquigarrow NULL$, *then* $\langle P \mid G\rangle \rightsquigarrow^{(\theta t)} \rightsquigarrow \cdots \rightsquigarrow NULL$.

Proof. We proceed by induction on the length of the SLD-proof. Consider the first resolution rule.

backchain Suppose

$$\langle P(t) \mid G(t)\rangle \overset{\varphi\pi,(tl_{cl}\Rightarrow G')}{\rightsquigarrow} \langle P(t)(\varphi\pi) \mid tl_{cl}\varphi\rangle \rightsquigarrow \cdots \rightsquigarrow^{\psi} \rightsquigarrow NULL$$

where $\psi\varphi\pi = \theta$ and $(\pi, tl_{cl} \Rightarrow G') \in \kappa(P(t))$. Consider the diagram

$$\begin{tikzcd}
\pi^{\#}_{\rho}(P(t)) \arrow[d] & P(t) \arrow[d] & P \arrow[d] \\
\rho \times \sigma \arrow[r, "\pi_\rho"] \arrow[dr, "t \times id_\sigma"'] & \rho \arrow[r, "t"] & \alpha \\
& \alpha \times \sigma \arrow[ur, "\pi_\alpha"'] &
\end{tikzcd}$$

By the inductive definition of the pullback $P(t)$, any $(tl_{cl} \Rightarrow G') \in \kappa(P(t))$ must be of the form $H(t \times id_\sigma) \Rightarrow B'(t \times id_\sigma)$ where $(H \Rightarrow B') \in \kappa(P)$. So the resolution sequence above is equivalent to

$$\langle P(t) \mid G(t)\rangle \overset{\varphi\pi_\rho,(H(t\times id_\sigma)\Rightarrow B'(t\times id_\sigma))}{\rightsquigarrow} \langle(\pi^{\#}_{\alpha}P)(t \times id_\sigma)\varphi \mid H(t \times id_\sigma)\varphi\rangle \rightsquigarrow \cdots \rightsquigarrow^{\psi} \rightsquigarrow NULL$$

But then

$$\langle P \mid G\rangle \overset{\varphi(t\times id_\sigma)\pi_\alpha,(H(t\times id_\sigma)\Rightarrow B'(t\times id_\sigma))}{\rightsquigarrow} \langle(\pi_\alpha^{\#}P)(t\times id_\sigma)\varphi \mid H(t\times id_\sigma)\varphi\rangle\rightsquigarrow\cdots\rightsquigarrow^{\psi}\rightsquigarrow NULL$$

with computed answer substitution $\psi\varphi(t\times id_\sigma)\pi_\alpha = \psi\varphi\pi_\rho t = \theta t$.

augment Suppose $\langle P(t) \mid (A_1 \Rightarrow A_2)(t)\rangle\rightsquigarrow\cdots\rightsquigarrow^{\theta}\rightsquigarrow NULL$. By the inductive definition of pullback then, this is

$$\langle P(t) \mid A_1(t) \Rightarrow A_2(t)\rangle\rightsquigarrow\langle P(t)\cup A_1(t) \mid A_2(t)\rangle\rightsquigarrow\cdots\rightsquigarrow^{\theta}\rightsquigarrow NULL$$

By the induction hypothesis, $\langle P\cup A_1 \mid A_2\rangle\rightsquigarrow\cdots\rightsquigarrow^{\theta t}\rightsquigarrow NULL$, so

$$\langle P \mid A_1 \Rightarrow A_2\rangle\rightsquigarrow\langle P\cup A_1 \mid A_2\rangle\rightsquigarrow\cdots\rightsquigarrow^{\theta t}\rightsquigarrow NULL$$

instance Suppose $\langle P(t) \mid (\exists_{x:\sigma}A)(t)\rangle\rightsquigarrow\cdots\rightsquigarrow^{\theta}\rightsquigarrow NULL$. Then since

$$\begin{tikzcd}
\exists_{x:\sigma}(A(t\times id_\sigma)) = (\exists_{x:\sigma}A)(t) \arrow[d] & A \arrow[d] \\
\rho\times\sigma \arrow[r, "t\times id_\sigma"] \arrow[d, "{\pi_\alpha,\exists}"'] & \alpha\times\sigma \arrow[d, "{\pi_\alpha,\exists}"] \\
\rho \arrow[r, "t"'] & \alpha
\end{tikzcd}$$

and

$$\begin{tikzcd}
\pi_\rho^{\#}(P(t)) \arrow[d] & P(t) \arrow[d] & P \arrow[d] \\
\rho\times\sigma \arrow[r, "\pi_\rho"] \arrow[dr, "t\times id_\sigma"'] & \rho \arrow[r, "t"] & \alpha \\
& \alpha\times\sigma \arrow[ur, "\pi_\alpha"'] &
\end{tikzcd}$$

the resolution above must be of the form

$$\langle P(t) \mid (\exists_{x:\sigma}A)(t)\rangle \overset{\pi_\rho}{\rightsquigarrow} \langle\pi_\rho^{\#}(P(t)) = (\pi_\alpha^{\#}P)(t\times id_\sigma) \mid A(t\times id_\sigma)\rangle\rightsquigarrow\cdots\rightsquigarrow^{\langle\theta,\beta\rangle}\rightsquigarrow NULL$$

for some β. By the induction hypothesis then since

$$\langle\pi_\rho^{\#}(P(t)) = (\pi_\alpha^{\#}P)(t\times id_\sigma) \mid A(t\times id_\sigma)\rangle\rightsquigarrow\cdots\rightsquigarrow^{\langle\theta,\beta\rangle}\rightsquigarrow NULL$$

we have

$$\langle \pi_\alpha^{\#} P \mid A\rangle \leadsto \cdots \leadsto^{\langle\theta,\beta\rangle(t\times id_\sigma)} \leadsto NULL$$

and so

$$\langle P \mid \exists_{x:\sigma} A\rangle \overset{\pi_\alpha}{\leadsto} \langle \pi_\alpha^{\#} P \mid A\rangle \leadsto \cdots \leadsto^{\langle\theta,\beta\rangle(t\times id_\sigma)} \leadsto NULL$$

with computed answer substitution $\langle\theta,\beta\rangle(t \times id_\sigma)(\pi_\alpha) = \theta t$.

and Suppose

$$\langle P(t) \mid (A_1(t) \wedge A_2(t))\rangle \leadsto \langle P(t) \mid A_1(t)\rangle \ \& \ \langle P(t) \mid A_2(t)\rangle \leadsto \cdots \leadsto^{\theta} \leadsto NULL$$

Then by the induction hypothesis,

$$\langle P \mid A_1\rangle \leadsto \cdots \leadsto^{\theta t} \leadsto NULL$$

and

$$\langle P \mid A_2\rangle \leadsto \cdots \leadsto^{\theta t} \leadsto NULL$$

Thus,

$$\langle P \mid (A_1 \wedge A_2)\rangle \leadsto \langle P \mid A_1\rangle \ \& \ \langle P \mid A_2\rangle \leadsto \cdots \leadsto^{\theta t} \leadsto NULL$$

or-right Suppose

$$\langle P(t) \mid (A_1(t) \vee A_2(t))\rangle \leadsto \langle P(t) \mid A_2(t)\rangle \leadsto \cdots \leadsto^{\theta} \leadsto NULL$$

Then by the induction hypothesis,

$$\langle P \mid A_2\rangle \leadsto \cdots \leadsto^{\theta t} \leadsto NULL$$

Thus

$$\langle P \mid (A_1 \vee A_2)\rangle \leadsto \langle P \mid A_2\rangle \leadsto \cdots \leadsto^{\theta t} \leadsto NULL$$

or-left Similarly.

□

Lemma A.2 (Lemma 4.9) *$P \vdash_o G$ iff there is an SLD-proof $\langle P \mid G\rangle \leadsto^{id} \leadsto \cdots \leadsto NULL$ with computed answer substitution id.*

Proof. That an SLD-proof of the form $\langle P \mid G\rangle \leadsto^{id} \leadsto \cdots \leadsto$ NULL implies that $P \vdash_o G$ is immediate by definition.

Suppose $P \vdash_o G$. Then there is a program Q and a formula H such that $\langle Q \mid H\rangle \leadsto^{\theta} \leadsto \cdots \leadsto$ NULL, $P = Q\theta$ and $G = H\theta$. We proceed by induction on the length of this SLD-proof. Consider the first resolution rule.

backchain Suppose

$$\langle Q \mid H\rangle \overset{\varphi\pi,(tl_{cl} \Rightarrow H')}{\leadsto} \langle Q(\varphi\pi) \mid tl_{cl}\varphi\rangle \leadsto \cdots \leadsto^{\psi} \leadsto NULL$$

where

$$\begin{tikzcd}
\rho \arrow[d, "\psi"] \arrow[dr, "\theta"] & \\
\beta \arrow[d, "\varphi"] & \alpha \\
\alpha \times \sigma \arrow[ur, "\pi_\alpha"'] &
\end{tikzcd}$$

which implies in particular that $\psi\varphi = \langle \theta, f \rangle$ for some $f : \rho \to \sigma$, and where $(tl_{cl} \Rightarrow H') \in \kappa(Q)$. By the induction hypothesis, we have that

$$\langle (Q(\varphi\pi))\psi \mid (tl_{cl}\varphi)\psi \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

And by considering the following diagram

$$\begin{tikzcd}
 & & \rho \arrow[dr, "\theta"] & \\
\rho \arrow[r, "\langle id, f\rangle"] \arrow[urr, "id"] \arrow[drr, "\langle \theta, f\rangle"'] & \rho \times \sigma \arrow[ur, "\pi_\rho"] \arrow[dr, "\theta \times id"] & & \alpha \\
 & & \alpha \times \sigma \arrow[ur, "\pi_\alpha"'] &
\end{tikzcd}$$

we have also that $(Q(\varphi\pi_\alpha))\psi = Q(\psi\varphi\pi_\alpha) = Q\theta = P = P(\langle id, f\rangle\pi_\rho) = (\pi_\rho^\# P)\langle id, f\rangle$, as well as $(tl_{cl}\varphi)\psi = tl_{cl}(\psi\varphi) = tl_{cl}\langle \theta, f\rangle = (tl_{cl}(\theta \times id))\langle id, f\rangle$. So

$$\langle (\pi_\rho^\# P)\langle id, f\rangle \mid (tl_{cl}(\theta \times id))\langle id, f\rangle \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

and since $(tl_{cl} \Rightarrow H') \in \kappa(Q)$ also implies by the inductive definition of pullback that $(tl_{cl}(\theta \times id) \Rightarrow H'(\theta \times id)) \in \kappa(Q\theta = P)$ we have

$$\langle P \mid G \rangle \overset{\langle id, f\rangle\pi_\rho = id,\, tl_{cl}(\theta \times id) \Rightarrow H'(\theta \times id)}{\rightsquigarrow} \langle (\pi_\rho^\# P)\langle id, f\rangle \mid (tl_{cl}(\theta \times id))\langle id, f\rangle \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

with computed answer substitution id.

augment Suppose

$$\langle Q \mid H_1 \Rightarrow H_2 \rangle \rightsquigarrow \langle Q \cup H_1 \mid H_2 \rangle \rightsquigarrow \cdots \rightsquigarrow^{\theta} \rightsquigarrow NULL$$

Then by the induction hypothesis,

$$\langle P \cup G_1 \mid G_2 \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

and so

$$\langle P \mid G_1 \Rightarrow G_2 \rangle \rightsquigarrow \langle P \cup G_1 \mid G_2 \rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

instance Suppose

$$\langle Q \mid \exists_{x:\sigma} H\rangle \stackrel{\pi_\alpha}{\rightsquigarrow} \langle(\pi^{\#}Q) \mid H\rangle \rightsquigarrow \cdots \rightsquigarrow^{\psi} \rightsquigarrow NULL$$

where the computed answer substitution is $\psi\pi_\alpha = \theta$ (and so $\psi = \langle\theta, \delta\rangle$ for some arrow $\delta : \rho \to \sigma$) and

$$\begin{tikzcd}
G \arrow[d] & H \arrow[d] \\
\rho \times \sigma \arrow[r, "\theta \times id_\sigma"] \arrow[d, "{\pi_\rho, \exists}"] & \alpha \times \sigma \arrow[d, "{\pi_\alpha, \exists}"'] \\
\rho \arrow[r, "\theta"'] & \alpha
\end{tikzcd}$$

By the induction hypothesis,

$$\langle P \mid H\psi\rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

But since $P = (\pi^{\#}P)\langle id_\rho, \delta\rangle$ and $H\psi = H\langle\theta, \delta\rangle = (H(\theta \times id_\sigma))(\langle id_\rho, \delta\rangle) = G\langle id_\rho, \delta\rangle$ this is equivalent to

$$\langle(\pi^{\#}P)\langle id_\rho, \delta\rangle \mid G\langle id_\rho, \delta\rangle\rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

Then, by lemma 4.8 we have

$$\langle P \mid \exists_{x:\sigma} G\rangle \stackrel{\pi_\rho}{\rightsquigarrow} \langle \pi_\rho^{\#} P \mid G\rangle \rightsquigarrow \cdots \rightsquigarrow^{\langle id_\rho, \delta\rangle} \rightsquigarrow NULL$$

with computed answer substitution id.

and Suppose

$$\langle Q \mid H_1 \wedge H_2\rangle \rightsquigarrow \langle Q \mid H_1\rangle \;\&\; \langle Q \mid H_2\rangle \rightsquigarrow \cdots \rightsquigarrow^{\theta} \rightsquigarrow NULL$$

Then by the induction hypothesis,

$$\langle P \mid G_1\rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

and

$$\langle P \mid G_2\rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

Thus

$$\langle P \mid G_1 \wedge G_2\rangle \rightsquigarrow \langle P \mid G_1\rangle \;\&\; \langle P \mid G_2\rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

or-right Suppose

$$\langle Q \mid H_1 \vee H_2\rangle \rightsquigarrow \langle Q \mid H_2\rangle \rightsquigarrow \cdots \rightsquigarrow^{\theta} \rightsquigarrow NULL$$

Then by the induction hypothesis,

$$\langle P \mid G_2\rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

and so

$$\langle P \mid G_1 \vee G_2\rangle \rightsquigarrow \langle P \mid G_2\rangle \rightsquigarrow \cdots \rightsquigarrow^{id} \rightsquigarrow NULL$$

or-left Similarly.

□

Lemma A.3 (Lemma 5.11) $[\![X_i(tv)]\!]^n_{P(t)}$ *is an isomorphism iff* $([\![t]\!])^{\#}([\![X_i(v)]\!]^n_P)$ *is an isomorphism.*

Proof. We proceed by induction on n. The case n=0 is trivial.

First, suppose $([\![t]\!])^{\#}([\![X_i(v)]\!]^n_P)$ is an isomorphism, where P is of sort ρ, X_i of sort σ, v is an arrow from ρ to σ and t is necessarily an arrow whose target is ρ and whose source is, say, η (see diagram below).

This means that for some pair $(\pi_1, B \Rightarrow X_i(u)) \in \kappa(P)$ and some substitution θ such that $\theta\pi_1 v = \theta u$, $([\![t]\!])^{\#}(\mathsf{Im}_{[\![\theta]\!]\pi_1}([\![\theta]\!])^{\#}[\![B]\!]^{n-1}_{P\pi_1})$ is an iso. By proposition 5.8 there is a substitution φ such that $t = \varphi\theta\pi_1$ and such that $[\![\varphi]\!]$ factors through $([\![\theta]\!])^{\#}[\![B]\!]^{n-1}_{P\pi_1}$. Thus $([\![\varphi\theta]\!])^{\#}[\![B]\!]^{n-1}_{P\pi_1}$ is an iso, and by the induction hypothesis, so is $[\![B(\varphi\theta)]\!]^{n-1}_{P\varphi\theta\pi_1}$.

The reader should note that if P is of sort ρ and the clause $B \Rightarrow X_i(u)$ of sort $\rho \times \mu$ is in $\kappa(P)$ then

- $P(t)$ is of sort η.
- The corresponding clause in $\kappa(P(t))$ is $(\ell, B(\langle \ell t, r\rangle) \Rightarrow X_i(\langle \ell t, r\rangle u))$ of sort $\eta \times \mu$, as it is the result of stripping quantifiers off of some formula diagram in $P(t)$ along a projection ℓ.
- A unifier γ of the head of this clause and the appropriately extended goal formula $X_i(\ell t v)$ of sort $\eta \times \mu$ must be targeted at $\eta \times \mu$ and satisfy $\gamma\langle \ell t, r\rangle u = \gamma\langle \ell t, r\rangle \pi_1 v$

as shown in the diagram below (where two triangles fail to commute: $\langle \ell t, r\rangle \neq \ell\varphi\theta$ and $u \neq \pi_1 v$)

$$\begin{tikzcd}
& X_i(\langle \ell t, r\rangle u) \arrow[d] & & X_i(u) \arrow[d] & B \arrow[dl, leftrightarrow, "\Leftarrow"'] \\
{} \arrow[r, "\gamma"] & \eta \times \mu \arrow[rr, "\langle \ell t, r\rangle"] \arrow[dd, "\ell"] & & \rho \times \mu \arrow[r, "u"] \arrow[dd, "\pi_1"'] & \sigma \\
& & \cdot \arrow[ur, "\theta"] & & \\
& \eta \arrow[uul, "\psi"] \arrow[ur, "\varphi"] \arrow[rr, "t"'] & & \rho \arrow[uur, "v"'] & \\
& & & & \langle P \mid X_i(v)\rangle \arrow[ul, leftrightarrow]
\end{tikzcd}$$

Now we consider $[\![X_i(tv)]\!]^n_{P(t)}$ which, by definition, is

$$\bigcup_{\substack{(\ell, B(\langle \ell t, r\rangle) \Rightarrow X_i(\langle \ell t, r\rangle u)) \in \kappa(P(t)) \\ \gamma\langle \ell t, r\rangle u = \gamma\langle \ell t, r\rangle \pi_1 v}} \{\mathsf{Im}_{[\![\ell]\!]}\mathsf{Im}_{[\![\gamma]\!]}(\,[\![\gamma]\!])^{\#}(\,[\![B(\langle \ell t, r\rangle)]\!]^{n-1}_{P(\ell t)})\} \tag{7}$$

One of the members of this union is that for which γ is the arrow $\eta \xrightarrow{\langle id_\eta, \varphi\theta\pi_0\rangle} \eta \times \mu$ where $\eta \xrightarrow{\varphi\theta} \rho \times \mu \xrightarrow{\pi_0} \mu$ since

$$\begin{aligned} \gamma\langle \ell t, r\rangle &= \langle t, \varphi\theta\pi_0\rangle \\ &= \langle \varphi\theta\pi_1, \varphi\theta\pi_0\rangle \\ &= \varphi\theta \end{aligned}$$

and we have shown that $\theta\pi_1 v = \theta u$. Also observe that $\gamma\ell$ is the identity id_η.

Now recall that we had concluded above (from consideration of the fact that $(\,[\![t]\!])^{\#}(\,[\![X(v)]\!]^n_P)$ is an isomorphism) that $[\![B(\varphi\theta)]\!]^{n-1}_{P(\varphi\theta\pi_1)}$ is an iso. Since $\gamma\langle \ell t, r\rangle = \varphi\theta$, we have that $[\![B\gamma\langle \ell t, r\rangle]\!]^{n-1}_{P(\gamma\langle \ell t, r\rangle\pi_1)}$ is an iso. Using the induction hypothesis again, we may conclude that $(\,[\![\gamma]\!])^{\#}\,[\![B\langle \ell t, r\rangle]\!]^{n-1}_{P(\ell t)}$ is an iso.

But then, since $\gamma\ell$ is the identity id_η, we have that

$$\mathsf{Im}_{[\![\gamma\ell]\!]}(\,[\![\gamma]\!])^{\#}(\,[\![B\langle \ell t, r\rangle]\!]^{n-1}_{P(\ell t)})$$

is also an iso. This is a member of the union in (7), so the union is an isomorphism, and thus so is $[\![X(tv)]\!]^n_{P(t)}$.

For the other direction, we refer to the same diagram above. Suppose $[\![X(tv)]\!]^n_{P(t)}$ is an isomorphism. This means $id_{[\![\eta]\!]}$ factors through it, and so through

$$\mathsf{Im}_{[\![\gamma\ell]\!]}(\,[\![\gamma]\!])^{\#}(\,[\![B\langle \ell t, r\rangle]\!]^{n-1}_{P(\ell t)})$$

for some substitution-clause pair $(\ell, B(\langle \ell t, r\rangle) \Rightarrow X_i(\langle \ell t, r\rangle u))$ in $\kappa(P(t))$ where $\gamma\langle \ell t, r\rangle u = \gamma\langle \ell t, r\rangle\pi_1 v$. Thus by proposition 5.8, there is a substitution ψ from η to the source of γ such that

- $id_\eta = \psi\gamma\ell$, and
- $(\,[\![\psi\gamma]\!])^{\#}(\,[\![B\langle \ell t, r\rangle]\!]^{n-1}_{P(\ell t)})$ is an iso.

Using the induction hypothesis this yields

$$[\![B\psi\gamma\langle \ell t, r\rangle]\!]^{n-1}_{P(\psi\gamma\ell t)} \qquad \text{is an iso .} \tag{8}$$

Now we wish to show that

$$(\,[\![t]\!])^{\#}(\,[\![X_i(v)]\!]^n_P) = \bigcup_{\substack{\theta\pi_1 v = \theta u \\ (\pi_1, B \Rightarrow X_i(u)) \in \kappa(P)}} \{(\,[\![t]\!])^{\#}(\mathsf{Im}_{[\![\pi_1]\!]}\mathsf{Im}_{[\![\theta]\!]}(\,[\![\theta]\!])^{\#}(\,[\![B]\!]^{n-1}_{P\pi_1}))\} \tag{9}$$

is an isomorphism. Consider the member of the union corresponding to the clause $B \Rightarrow X_i(u)$ and with the substitution θ chosen to be $\psi\gamma\langle \ell t, r\rangle$ (taking φ to be the identity on η in the diagram). We have $\theta\pi_1 = \psi\gamma\ell t$, so from (8) we infer that $[\![B\theta]\!]^{n-1}_{P\theta\pi_1}$ is an iso. By the induction hypothesis again we have $([\![\theta]\!])^{\#}([\![B]\!]^{n-1}_{P\pi_1})$ is an iso. Furthermore, since $id_\eta = \psi\gamma\ell$, we have

$$\begin{aligned} \mathsf{Im}_{[\![\pi_1]\!]}\mathsf{Im}_{[\![\theta]\!]} &= \mathsf{Im}_{[\![\theta\pi_1]\!]} \\ &= \mathsf{Im}_{[\![\psi\gamma\ell t]\!]} \\ &= \mathsf{Im}_{[\![t]\!]} \end{aligned}$$

Thus the member of the union in (9) corresponding to this clause and substitution, ie:

$$([\![t]\!])^{\#}\mathsf{Im}_{[\![\pi_1]\!]}\mathsf{Im}_{[\![\theta]\!]}([\![\theta]\!])^{\#}([\![B]\!]^{n-1}_{P\pi_1})$$

reduces to $([\![t]\!])^{\#}\mathsf{Im}_{[\![t]\!]}([\![\theta]\!])^{\#}([\![B]\!]^{n-1}_{P\pi_1})$. Now, since for any subobject F and (suitably targeted) arrow f, $F \subset (f)^{\#}\mathsf{Im}_f(F)$, this member of the union is an isomorphism, and thus so is $([\![t]\!])^{\#}([\![X_i(v)]\!]^n_P)$. □

Theorem A.4 (Theorem 5.13)

$$E(P, [\![\,]\!], X_i, v) = \bigcup_{\substack{(\pi, cl) \in \kappa(P) \\ hd_{cl} = X_i}} \{\mathsf{Im}_{[\![\pi]\!]}\mathsf{Im}_{\mathsf{eq}([\![\pi v]\!],\, [\![tm_{cl}]\!])}(\mathsf{eq}([\![\pi v]\!],\, [\![tm_{cl}]\!]))^{\#}\, [\![tl_{cl}]\!]^{n-1}_{P\pi}\}$$

Proof. We establish that for any arrows u, t with same source (say σ) and target, and any subobject F of $[\![\sigma]\!]$

$$\mathsf{Im}_{\mathsf{eq}([\![u]\!],\, [\![t]\!])}(\mathsf{eq}([\![u]\!],\, [\![t]\!]))^{\#}(F) = \bigcup_{\theta u = \theta t} \mathsf{Im}_{[\![\theta]\!]}([\![\theta]\!])^{\#}(F). \tag{10}$$

from which the theorem follows easily.

We will need to be careful with notation here: we let $\overline{\mathsf{eq}([\![u]\!],\, [\![t]\!])}$ denote the equalizer *object*, and e the monic $\mathsf{eq}([\![u]\!],\, [\![t]\!])$ from the equalizer to the source σ of the equalized arrows.

We begin by observing that, as with any object in $\mathsf{Set}^{\mathbb{C}^o}$, $\overline{\mathsf{eq}([\![u]\!],\, [\![t]\!])}$ is the colimit of a family of representable objects $[\![\varepsilon_\alpha]\!]$, each equipped with a map f_α into $\overline{\mathsf{eq}([\![u]\!],\, [\![t]\!])}$.

$$\begin{tikzcd} {[\![\varepsilon_\alpha]\!]} \arrow[r, "f_\alpha"] & \overline{\mathsf{eq}([\![u]\!],\, [\![t]\!])} \arrow[r, tail, "e"] & \sigma \arrow[r, shift left, "{[\![u]\!]}"] \arrow[r, shift right, "{[\![t]\!]}"'] & {} \end{tikzcd}$$

For each α, the composition $f_\alpha e$ has representable source and target. Since the Yoneda embedding is full, each one is the image $[\![\theta]\!]$ of some arrow θ in $\mathbb{C}$ which must be a unifier of u and t as the reader can check. We now omit all reference to the objects $[\![\varepsilon_\alpha]\!]$ and use the label ι_θ for the arrows f_α from the source of each $[\![\theta]\!]$ into $\overline{\mathsf{eq}([\![u]\!],\, [\![t]\!])}$. This information is displayed in the diagram below, together with the epi-mono image factorization of $[\![\theta]\!]$

$$\begin{tikzcd} \overline{\mathsf{eq}([\![u]\!],\, [\![t]\!])} \arrow[rr, tail, "e"] & & \sigma \arrow[r, shift left, "{[\![u]\!]}"] \arrow[r, shift right, "{[\![t]\!]}"'] & {} \\ & \cdot \arrow[ul, "\iota_\theta"] \arrow[ur, "{[\![\theta]\!]}"] \arrow[r, "u_\theta"'] & \mathsf{Im}_{[\![\theta]\!]}(id) \arrow[u, tail, "j_\theta"'] & \end{tikzcd}$$

It is straightforward to show, using the universal properties of equalizers, colimits and unions, that e, whose source is by hypothesis the colimit of the objects that are the sources of the $[\![\theta]\!]$, is equal, as a subobject of σ, to the union $\bigcup j_\theta$.

Making use of this fact we have, for any subobject F of σ,

$$\mathsf{Im}_e(e)^{\#}(F) = \mathsf{Im}_{\bigcup j_\theta}(\bigcup j_\theta)^{\#}(F)$$

Note that if a and b are any subobjects of an object σ

$$a \cap b = \mathsf{Im}_a(a)^{\#}(b) \tag{11}$$

so $\mathsf{Im}_{\bigcup j_\theta}(\bigcup j_\theta)^{\#}(F)$ is precisely the intersection

$$(\bigcup j_\theta) \cap F$$

which distributes in the co-complete topos $\mathsf{Set}^{\mathbb{C}^o}$ to yield

$$\bigcup (j_\theta \cap F).$$

Using fact (11) again, we obtain

$$\bigcup \mathsf{Im}_{j_\theta}(j_\theta)^{\#}(F)$$

It follows that

$$\mathsf{Im}_e(e)^{\#}(F) = \bigcup \mathsf{Im}_{j_\theta}(j_\theta)^{\#}(F).$$

Observe, however, that

$$\begin{aligned} &\mathsf{Im}_{[\![\theta]\!]}([\![\theta]\!])^{\#}(F) \\ &\qquad = \mathsf{Im}_{u_\theta j_\theta}(u_\theta j_\theta)^{\#}(F) \\ &\qquad = \mathsf{Im}_{j_\theta}\mathsf{Im}_{u_\theta}(u_\theta)^{\#}(j_\theta)^{\#}(F) \\ &\qquad = \mathsf{Im}_{j_\theta}(j_\theta)^{\#}(F) \end{aligned}$$

the last equation following from the fact that the composition $\mathsf{Im}_{u_\theta}(u_\theta)^{\#}$ yields identity since u_θ is epic. This estabishes the identity (10) above.

Now letting F be $[\![tl_{cl}]\!]_{P\pi}$ and taking the arrow t to be πv we conclude that

$$\mathsf{Im}_{\mathsf{eq}([\![\pi v]\!],\, [\![u]\!])}(\mathsf{eq}([\![\pi v]\!],\, [\![u]\!]))^{\#}([\![tl_{cl}]\!]_{P\pi}) = \bigcup_{\theta u = \theta\pi v} \{\mathsf{Im}_{[\![\theta]\!]}([\![\theta]\!])^{\#}([\![tl_{cl}]\!]_P)\}.$$

Applying the image functor $\mathsf{Im}_{[\![\pi]\!]}$ to both sides (and observing that it commutes with unions), the theorem follows readily. $\square$

B A Sketch of the Grothendieck Construction

An indexed category $\mathcal{I}$ over a base category $\mathbb{C}$ is a $\mathbb{C}$-indexed family of categories (the fibers), together with a functor $\mathcal{I}f$ between fibers $\mathcal{I}\sigma$ and $\mathcal{I}\rho$ associated with every arrow $\sigma \xrightarrow{f} \rho$ in the base category. By such a device we resolve logic and logic programing structure into a (vertical) propositional logical component (the structure in the fibers) and the (horizontal) predicate logic and substitution component, which, as it turns out, is a special case of state change in logic programming. We have seen that the base category may also capture change of ambient program in the Weak Hereditarily Harrop case.

But logic program resolutions act on state-goal pairs, or vectors of such. So actual resolution steps do not correspond to any arrows in the indexed category framework: they are paths consisting of arrows in the fibers followed by shifts along reindexing functors. This problem can be remedied by a simple, elegant construction due to Grothendieck that builds a single category out of an indexed category, indeed, a category together with a functor *into* the base category, called a fibration (or op-fibration if the indexed category is covariant). This construction allows us to visualize quite intricate logic programming resolutions in conventional or extended logic programming as automatically generated by a simple core of data: program clauses in the fibers and state change definitions in the base of an indexed category.

In this paper we have also shown how a category $\mathbb{C}[X]$ of freely generated (generic) predicates over a base category arises as a result of the Grothendieck construction over basic (propositional) data presented in an indexed category $\Pi_{\boldsymbol{b}}$.

Definition B.1 *Given an indexed category $\mathcal{I} : \mathbb{C} \longrightarrow \mathbb{CAT}$, the category $\mathsf{G}(\mathbb{C}, \mathcal{I})$ is defined as follows*

objects: *pairs (σ, G) where $\sigma \in |\mathbb{C}|$ and $G \in |\mathcal{I}\sigma|$.*

arrows: *pairs $(\theta, u) : (\sigma, G) \longrightarrow (\rho, K)$ where $\sigma \xrightarrow{\theta} \rho$ is an arrow in the base category $\mathbb{C}$ and $\mathcal{I}\theta(K) \xrightarrow{u} G$ an arrow in the fiber $\mathcal{I}(\sigma)$.*

Composition

$$(\sigma, G) \xrightarrow{(\theta,u)} (\rho, K \xrightarrow{(\varphi,v)})(\alpha, V)$$

is given by the pair $(\theta\varphi, \mathcal{I}\theta(v)u)$, the second component of which is the composition

$$\mathcal{I}\theta(\mathcal{I}\varphi(V)) \xrightarrow{\mathcal{I}\theta(v)} \mathcal{I}\theta(U) \xrightarrow{u} G.$$

We refer the reader to the exposition in [6] for a proof of the fact that $\mathsf{G}(\mathbb{C}, \mathcal{I})$ is category which, together with projection functor $\mathsf{G}(\mathbb{C}, \mathcal{I}) \longrightarrow \mathbb{C}$ yields a split fibration, and that this correspondence induces an equivalence of the categories of indexed categories and split fibrations.

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] Gianluca Amato. *Sequent Calculus and Indexed Categories as Foundations for Logic Programming.* PhD thesis, University of Pisa, 2000.

[3] Gianluca Amato and James Lipton. Indexed categories and bottom-up semantics of logic programing. In *Proceedings of LPAR '01*, volume 2250 of *Lecture Notes in Artificial Intelligence*, 2002.

[4] A. Asperti and S. Martini. Projections instead of variables, a category theoretic interpretation of logic programs. In *Proc. 6^{th} ICLP*, pages 337–352. MIT Press, 1989.

[5] R. Barbuti, R. Giacobazzi, and G. Levi. A general framework for semantics based, bottom-up abstract interpretation of logic programs. *ACM Transactions on Programming Languages and Systems*, 15(1):133–181, 1993.

[6] Michael Barr and Charles Wells. *Category Theory for Computing Science.* Prentice Hall, 1995.

[7] N. Benton, G. Bierman, V. de Paiva, and J.M.E. Hyland. Term assignment for intuitionistic linear logic. In *Proc. CSL '92*, 1993.

[8] 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.

[9] Jawahar Chirimar. *Proof Theoretic Approach to Specification Languages.* PhD thesis, University of Pennsylvania, 1995.

[10] Jacques Cohen. Constraint logic programming languages. *Communications of the ACM*, 33(7):52–68, July 1990.

[11] M. Comini, C. Meo, and G. Levi. A theory of observables in logic programming. *Information and Computation*, 169(1):23–80, 2000.

[12] 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.

[13] Andrea Corradini and Ugo Montanari. An algebraic semantics for structured transition systems and its application to logic programs. *Theoretical Computer Science*, 103:51–106, 1992.

[14] Patrick Cousot and Radhia Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Programming*, 13(2-3):103–179, July 1992.

[15] Roy Crole. *Categories for Types.* Cambridge University Press, 1993.

[16] Oege de Moor. *Categories, Relations, and Dynamic Programming.* PhD thesis, St. John's College, Oxford, 1991.

[17] M. DeMarco. *Higher Order Logic Programming in the Theory of Types.* PhD thesis, Wesleyan University, 1999.

[18] M. DeMarco and J. Lipton. The semantics of higher order hereditary harrop logic programming. To appear.

[19] R. Diaconescu. *Category Semantics for Equational and Constraint Logic Programming.* PhD thesis, Oxford University, 1994.

[20] Stacy E. Finkelstein, Peter Freyd, and James Lipton. Logic programming in tau categories. In *Computer Science Logic '94, LNCS 933.* Springer, 1995.

[21] P. Freyd, P. Mulry, G. Rosolini, and D. Scott. Extensional PERs. *Information and Computation*, 98(2)::211–227, 1992.

[22] Peter Freyd. The logic of cartesian categries. To appear.

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

[24] J.Y. Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59:201–217, 1993.

[25] J.Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* Cambridge University Press, 1998.

[26] J. Harland and D. Pym. A uniform proof-theoretic investigation of linear logic programming. *Journal of Logic and Computation*, 4(2), 1994.

[27] Claudio Hermida. *Fibrations,logical predicates and indeterminates.* PhD thesis, University of Edinburgh, 1993.

[28] Joshua Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, May 1994.

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

[30] Joxan Jaffar and Michael Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[31] Neil D. Jones. Abstract interpretation and partial evaluation in functional and logic programming. In Maurice Bruynooghe, editor, *Logic Programming. Proceedings of the 1994 International Symposium*, pages 17–22. The MIT Press, 1994.

[32] T. K. Lakshman and Uday S. Reddy. Typed Prolog: A semantic reconstruction of the Mycroft-O'Keefe type system. In *Proceedings of the 1991* IEEE Symposium on Logic Programming, pages 202–220, 1991.

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

[34] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic*. Springer-Verlag, 1992.

[35] J. L. Lassez and K. MacAloon. A constraint sequent calculus. In *Proceedings of tthe 5th Symp. on LICS*, 1990.

[36] G. Levi. Models, unfolding rules, and fixpoint semantics. In R.A. Kowalski and K. A. Bowen, editors, *Proc. Fifth International Symposium of Logic Programming*. The MIT Press, 1988.

[37] J. Lipton and R. McGrail. Encapsulating data in logic programming via categorical constraints. In C. Palamidessi, H. Glaser, and K. Meinke, editors, *Principles of Declarative Programming*, volume 1490 of *LNCS*. Springer Verlag, 1998.

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

[39] M. Martelli M. Falaschi, G. Levi and C. Palamidessi. Declarative modeling of the operational behavior of logic languages. *TCS*, 69(3):289–318, 1989.

[40] M. Martelli M. Falaschi, G. Levi and C. Palamidessi. A model-theoretic reconstruction of the operational semantics of logic programs. *Information and Computation*, 102(1):86–113, 1993.

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

[42] R. McGrail. *Monads and Control in Logic Programming*. PhD thesis, Wesleyan University, 1997.

[43] Maria Chiara Meo. *A Framework for Reasoning about the Semantics of Logic Programs*. PhD thesis, University of Pisa, 1996.

[44] Dale Miller. A logical analysis of modules in logic programming. *Journal of Logic Programming*, pages 79–108, 1989.

[45] Dale Miller. A multiple-conclusion meta-logic. In *Proceedings of the 1994 Symposium on Logics in Computer Science*, pages 272–281, 1994.

[46] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51(1-2):125–157, 1991.

[47] E. Moggi. Notions of computation and monads. *Information and Computation*, 93, 1991.

[48] J. J. Moreno-Navarro and M. Rodríguez Artalejo. Logic programming with functiona and predicates: The language babel. *The Journal of Logic Programming*, 12(3/4):191–223, 1992.

[49] A. Mycroft and R. A. O'Keefe. A Polymorphic Type System for Prolog. *Artificial Intelligence*, 23(3):295–307, 1984.

[50] Gopalan Nadathur and Dale Miller. Higher-order horn clauses. *Journal of the ACM*, 37(4):777–814, 1990.

[51] Gopalan Nadathur and Dale Miller. Higher-order logic programming. In *Handbook of Logics for Artificial Intelligence and Logic Programming*, volume 5. Oxford University Press, 1996.

[52] P. W. O'Hearn and R. D. Tennent. Parametricity and local variables. Technical report, Syracuse University, 1993.

[53] L. Ong. Non-determinism in functional programming. In *Proceedings of the 8th IEEE Symposium on Logic and Computer Science*, 1993.

[54] 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.

[55] Frank Pfenning. *Types in Logic Programming*, chapter Dependent Types in Logic Programming. MIT, 1992.

[56] David Pitt, Samson Abramsky, Axel Poigne, and David Rydeheard, editors. *Category Theory and Computer Programming*. Springer-Verlag, New York, 1985. Lecture Notes in Computer Science 240.

[57] G. Plotkin and J. Reynolds. *Logical Foundations of Functional Programming*, chapter On functors expressible in the polymorphic lambda calculus. Addison-Wesley, 1990.

[58] A. Poigné. Algebra categorically. In *Category Theory and Computer Programming.* Springer, 1986.

[59] J. Power and Y. Kinoshita. A new foundation for logic programming. In *Extensions of Logic Programming '96.* Springer Verlag, 1996.

[60] D. Pym. Functorial kripke models of the $\lambda\pi$-calculus. Lecture at Newton Institute Semantics Programme, Workshop on Category Theory and Logic Programming, Cambridge, September 1995, 1995.

[61] D.E. Rydeheard and R.M. Burstall. A categorical unification algorithm. In *Category Theory and Computer Programming*, volume 240 of *LNCS*, pages 493–505, 1985.

[62] M. Spivey. A functional theory of exceptions. *Science of Computer Programming*, 14(1):25–42, 1990.

[63] T. Streicher. *Semantics of Type Theory.* Birkhäuser, 1992.

[64] P. J. Stuckey. Negation and constraint logic programming. *Information and Computation*, 118, 1995.

[65] Paul Taylor. *Practical Foundations of Mathematics.* Cambridge University Press, 1999.

[66] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics: An Introduction*, volume 2. Elsevier Science Publishers, 1988.

[67] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23, 1976.

[68] P. Wadler. Comprehending monads. *Mathematical Structures in Computer Science*, 2, 1992.

[69] P. Wadler. How to declare an imperative. In *Proc.International Logic Programming Symposium.* MIT Press, 1995.