2-1992

A Declarative Foundation of λProlog with Equality

Mino Bai

SU-CIS-92-03

A Declarative Foundation of λProlog with Equality

Mino Bai

February, 1992

School of Computer and Information Science
Syracuse University
Suite 4-116, Center for Science and Technology
Syracuse, NY 13244-4100

A Declarative Foundation of λProlog with Equality

Mino Bai*
School of Computer and Information Science
Syracuse University, Syracuse, NY 13244, USA

February 27, 1992

Abstract

We build general model-theoretic semantics for higher-order logic programming languages. Usual semantics for first-order logic is two-level: i.e., at a lower level we define a domain of individuals, and then, we define satisfaction of formulas with respect to this domain. In a higher-order logic which includes the propositional type in its primitive set of types, the definition of satisfaction of formulas is mutually recursive with the process of evaluation of terms. As result of this in higher-order logic it is extremely difficult to define an effective semantics. For example to define $T_{\mathcal{P}}$ operator for logic program $\mathcal{P}$, we need a fixed domain without regard to interpretations. In usual semantics for higher-order logic, domain is dependent on interpretations. We overcome this problem and argue that our semantics provides a more suitable declarative basis for higher-order logic programming than the usual general model semantics. We develop a fix point semantics based on our model. We also show that a quotient of the domain of our model can be the domain of a model for higher-order logic programs with equality.

1 Introduction

Many extended versions of Prolog are developed which incorporate higher-order features in logic programming languages to make programs more versatile and expressive [22, 7, 1]. One important motivation for adding higher-order features to logic programming is the need to allow programs to manipulate such things as formulas, predicates, and programs themselves in the same way as they manipulate *object level terms* (for further motivation for higher-order logic programming see [11, 12, 19, 20]).

In this paper, we build a model-theoretic semantics for a higher-order logic programming language which is suitable for describing declaratively operations of such programming language.

Church [8] introduced a simple theory of types as a system of higher-order logic. This system incorporated λ-notation in its particularly simple syntax which actually be viewed as a version of simply typed λ-calculus. Henkin first gave a semantics for Church's system based on general models. Domain members of a general model are truth values, individuals, and functions. Church's system was proved to

**Address correspondence to* author, School of Computer and Information Science, Center for Science and Technology/Fourth Floor, Syracuse University, Syracuse, New York 13244-4100, USA, *Email address of* author, mbai@top.cis.syr.edu be complete with respect to Henkin's semantics [13]. Andrews studied general models further in [3, 4, 5], and built a non-extensional model which is suitable under settings of resolution theorem proving [2]. The proof theory for this system is shown to have a close resemblance to that of first-order logic: there is, for example, a generalization to Herbrand theorem that holds for a variant of this system [18].

λProlog [22] was the first language to show that higher-order logic could be used as the basis of a practical programming language. λProlog is based on typed λ-calculi which have their ultimate origin in Russel's method of stratifying sets to avoid the set theoretic paradoxes. In higher-order logic on which λProlog is based, compared to first-order case, it is extremely difficult to build an effective model-theoretic semantics. One of these difficulties is that the definition of satisfaction of formulas is mutually recursive with the process of evaluation of terms (see [13, 2, 3, 4, 5]). In first-order case, the model-theory is two level [16, 21, 24]. First we define a domain of individuals, and then define satisfaction wrt this domain. The second reason is that since higher-order logic programming languages are usually formulated in *non-extensional* form, we need a non-extensional model to describe properly such languages.

Henkin's general model semantics is extensional: i.e., if two objects in a model have the same extension, then they must be equal. Extensional models are very difficult to deal with, and unsuitable to describe a higher-order logic programming language like λProlog which contain a propositional type in its primitive set of types. For example, we can define a program $\mathcal{P}_1 = \{p(a), q(a), r(p(a))\}$ in λProlog. Given program $\mathcal{P}_1$, the goal $r(p(a))$ will succeed in λProlog, but the goal $r(q(a))$ will fail, since the unification of $r(q(a))$ and $r(p(a))$ will simply fail. For any extensional model $\mathcal{M}$ for $\mathcal{P}_1$, $\mathcal{M}$ will assign the value $\mathbf{T}$ for $p(a)$ and $q(a)$. So $p(a) = q(a)$ is a logical consequence of $\mathcal{P}_1$. $\mathcal{M}$ will also assign the value $\mathbf{T}$ to $r(p(a))$, so the extension of the predicate which $\mathcal{M}$ will assign to r contains $\mathbf{T}$. Therefore $r(q(a))$ is a logical consequence of the program $\mathcal{P}_1$. Note that for this program the valuation of terms is mutually recursive with the satisfaction of formulas, since a formula can occur as an argument of predicate or functional symbols.

2 λProlog

In this section we describe a higher-order logic programming language for which we build models in the later sections. For the exposition of the logic programming language λProlog we will follow closely those in [22, 20].

The set $\mathcal{T}$ of types contains a collection $\mathcal{T}_0$ of primitive types and is closed under the formation of functional types: i.e., if $\alpha, \beta \in \mathcal{T}$, then $(\alpha \to \beta) \in \mathcal{T}$. The type constructor $\to$ associates to the right.

The type $(\alpha \to \beta)$ is that of a function from objects of type α to objects of type β. We assume that the type constructor $\to$ associates to the right, and we shall often write type expressions such as

$\alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \beta$ in the form $\alpha_1, \cdots, \alpha_n \rightarrow \beta$, or even in the form $\overline{\alpha} \rightarrow \beta$, with β an arbitrary type.

For $\gamma \in \mathcal{T}_0$, the set $\mathcal{T}^\gamma$ of types of *kind* γ [10] is the subset of $\mathcal{T}$ defined inductively by:

(a) $\gamma \in \mathcal{T}^\gamma$.

(b) If $\alpha \in \mathcal{T}$ and $\beta \in \mathcal{T}^\gamma$, then $(\alpha \rightarrow \beta) \in \mathcal{T}^\gamma$.

We introduce a very convenient notation from [23]. For each type symbol α, and each set S containing objects or expressions, we write S_α to denote the *set of things in* S *which are of type* α. We sometimes write $\{S_\alpha\}_\alpha$ to denote S. We can also define a *type assignment mapping* τ on the set S such that $\tau : S \rightarrow \mathcal{T}$ and for all $s \in S$, $\tau(s) = \alpha$ if $s \in S_\alpha$.

Let S and T be sets. The set $S \rightarrow T$ is the *set of all mappings from* S *into* T. Given a mapping $f : S \rightarrow T$, $a \in S$, and $b \in T$, let $f[b/a]$ be that mapping $f' : S \rightarrow T$ such that for $f'a = b$ and $f'c = fc$ for all $c \neq a$. Given a set $S_1 \subseteq S$, we write $f(S_1)$ for $\{f(a) : a \in S_1\}$.

Let S, T_1, T_2 be sets, and $f : S \rightarrow T_1$ and $g : T_1 \rightarrow T_2$. The *composition* of f and g is the function $f \circ g : S \rightarrow T_2$ such that for all $s \in S$, $f \circ g(s) = g(f(s))$.

Let T_1 and T_2 be sets and b be an element in $T_1 \times T_2$, then b^1 and b^2 are the first and second components of b, so $b = \langle b^1, b^2 \rangle$. If f is a mapping whose values are in $T_1 \times T_2$, let f^1 and f^2 be mappings with the same domain as f defined so that for any argument t, $f^i t = (ft)^i$ for $i = 1, 2$. Thus $ft = \langle f^1 t, f^2 t \rangle$.

Let S, T, T_1, T_2 be sets. If $f : S \rightarrow T$ is a mapping, then we say that f is *good* if for all $s \in S$, $\tau(f(s)) = \tau(s)$. If $f : S \rightarrow T_1 \times T_2$, then we say that f is *good* if f^1 and f^2 are good. In this paper we consider only good mappings, so a mapping is always assumed to be good unless said otherwise.

Let $\mathcal{T}_1$ be a subset of $\mathcal{T}$ and S a set. By $S \upharpoonright \mathcal{T}_1$ we mean $\{S_\alpha\}_{\alpha \in \mathcal{T}_1}$ the subset of S consisting of elements in S of types in $\mathcal{T}_1$. When f is a mapping whose domain is S, then $f \upharpoonright \mathcal{T}_1$ is defined to be $f \upharpoonright (S \upharpoonright \mathcal{T}_1)$ the restriction of f to $S \upharpoonright \mathcal{T}_1$.

For each integer $n \in \omega$, we write $[n]$ for the set $\{1, \cdots, n\}$.

We assume that there are denumerably many variables and constants of each type. Let the set of variables and constants be Δ and Σ, respectively. Then for each type symbol α, Δ_α is the set of variables of type α and Σ_α is the set of constants of type α. So, $\Delta = \bigcup_{\alpha \in \mathcal{T}} \Delta_\alpha$ and $\Sigma = \bigcup_{\alpha \in \mathcal{T}} \Sigma_\alpha$.

Simply typed λ*-terms* are built up in the usual fashion from these typed constants and variables via abstraction and application. Our *well formed terms* (wfts) are simply typed λ-terms. We define the set $T(\Sigma)$ of all wfts by giving the definition of the set $T(\Sigma)_\alpha$ of wfts of type α by induction:

(a) $\Delta_\alpha \subseteq T(\Sigma)_\alpha$.

(b) $\Sigma_\alpha \subseteq T(\Sigma)_\alpha$.

(c) If $t_\beta \in T(\Sigma)_\beta$ and $x_\alpha \in \Delta_\alpha$, then $[\lambda x_\alpha t_\beta] \in T(\Sigma)_{\alpha \rightarrow \beta}$.

(d) If $f_{\alpha\to\beta} \in T(\Sigma)_{\alpha\to\beta}$ and $t_\alpha \in T(\Sigma)_\alpha$, then $[f_{\alpha\to\beta}t_\alpha] \in T(\Sigma)_\beta$.

It is assumed that the reader is familiar with most of basic notions and definitions such as bound, free variables, closed terms (c-terms), substitution and λ-conversion for this language; only a few are reviewed here.

Letters $f_\alpha, s_\alpha, t_\alpha, \cdots$, will be used as syntactical variables of wfts of type α. Type subscript symbols may be omitted when context indicates what they should be or irrelevant to discussion. We use the expression $FV(t)$ to denote the set of free variables of a wft t.

By Church-Rosser theorem [6], a λ-normal wfts of a wft is unique upto a renaming of variables. For most part we shall be satisfied with any of these normal forms corresponding to a wft t, and we shall write $\lambda norm(t)$ to denote such a form. In certain situations we shall need to talk about a unique normal form and, in such cases, we shall use $\rho(t)$ to designate what we shall call the *principal normal* or *ρ-normal* form of t; i.e. ρ is a mapping from wfts to λ-normal terms. There are several schemes that may be used to pick a representative of the α-equivalence classes of λ-normal terms and the one implicitly assumed here is that of [2].

We define a notion of substitution as a mapping from and to wfts. A *substitution* is any total good mapping $\sigma : \Delta \to T(\Sigma)$ such that $\sigma(x) \neq x$ for only finitely many $x \in \Delta$. Given a substitution σ, the *domain* of σ is the set of variables $Dom(\sigma) = \{x \in \Delta : \sigma(x) \neq x\}$. A substitution whose domain is empty is called the *identity substitution*, and is denoted by ϵ. Given a substitution σ, if its domain is the set $\{x_1, \cdots, x_n\}$, and if $t_i = \sigma(x_i)$ for $i \in [n]$, then σ is also denoted by listing its binding explicitly: $[t_1/x_1, \cdots, t_n/x_n]$. We say that σ is *λ-normal(ρ-normal, closed) substitution* if each t_i, for $i \in [n]$, is in λ-normal (ρ-normal, closed) form. Given a term t, we define $\sigma(t) = \rho([\lambda x_1 \cdots x_n \cdot t]t_1 \cdots t_n)$. Finally given an arbitrary mapping $\chi : \Delta \to T(\Sigma)$, we can consider χ as a *generalized substitution* such that for each term t, $\chi(t) = (\chi \uparrow FV(t))(t)$.

So far we have introduced λ-term structures and operations on λ-terms. We can introduce logic into λ-term structures by including o, a type for propositions, amongst the set of primitive types $\mathcal{T}_0$, and requiring that the collection Σ of constants contain the following *logical* constants: $\wedge$ and $\vee$ of type $o \to o \to o$; $\top$ of type o; and for every type α, $\exists_\alpha$ of type $(\alpha \to o) \to o$. The constants in Σ other than $\wedge, \vee, \exists$ and $\top$ are called as *non-logical* constants.

A type will be called a *predicate type* if it is a type in the set $\mathcal{T}^o$ of types of kind o, or a *non-predicate type* otherwise. A variable of predicate type will be called *predicate variable.* And a non-logical symbol in Σ of predicate type will be called *predicate constant.* We let $\Pi \subseteq \Sigma$ be the set of predicate constants.

The type subscript on $\exists$ will be omitted except when its value is essential in a discussion and can not be inferred from the context. Expression of the form $\exists(\lambda x G)$ will be abbreviated by $\exists x G$.

Terms of type o are referred to as *goal formula* or just *formula.* The λ-normal form of a formula consists, at the outermost level, of a sequence of applications, and the leftmost symbol in this sequence is called its *top level symbol.* We shall have use for the structure of λ-normal formulas that is described below. A formula is said to be an *atom* (*atomic*) if its leftmost symbol that is not a bracket is either a predicate variable or constant. A λ-normal goal formula G, then, has the following inductive characterization:

(a) it is $\top$.

(b) it is an atom.

(c) it is $G_1 \wedge G_2$ or $G_1 \vee G_2$, where G_1 and G_2 are λ-normal formulas, or

(d) it is $\exists x G$, where G is a λ-normal formula.

Now we identify the formulas that we call higher-order definite clauses, sentences, goal formula, and equations.

Let $\mathcal{G}$ be the collection of all λ-normal formulas. A *goal formula* is a formula G in $\mathcal{G}$. An *atom* is an atomic goal formula A. A *rigid atom* is an atom A_r that has a predicate constant as its head. An atom is thus a formula of the form $pt_1 \cdots t_n$ where $\gamma = \alpha_1, \cdots, \alpha_n \rightarrow o$, p is a predicate constant$_\gamma$ or variable$_\gamma$, and, for each $i \in [n]$, t_i is a λ-normal term$_{\alpha_i}$, it is a rigid atom just in case p is a constant. Sometimes we write $p(t_1, \cdots, t_n)$ or $p(\overline{t})$ for the above atom.

Let G be an arbitrary goal formula and A_r be any rigid atom. Let $\overline{x}$ be an arbitrary listing of all the variables free in either G or A_r. Let a formula C be of the form $A_r \leftarrow G$. Then C is a *(higher-order) definite clause* and the formula $\forall \overline{x} \cdot D$ is a *(higher-order) definite sentence.*

Let $s_\alpha, t_\alpha \in T(\Sigma)$. Then, as usual, an *equation* e is of the form $s_\alpha = t_\alpha$, and an *extensional equation* is of the form $s_\alpha \equiv t_\alpha$.

Let $\mathcal{D}ef$ be the set of all definite clauses. Then given the collection Σ of constants, our *logic programming language* $\mathcal{L} = \mathcal{L}(\Sigma)$ is completely determined as the triple $\langle T(\Sigma), \mathcal{G}, \mathcal{D}ef \rangle$. A formula in a language $\mathcal{L}$ is a goal formula, or a definite clause, or an equation.

We refer a set $\mathcal{P}$ of formulas from $\mathcal{D}ef$ as a *higher-order definite logic program.* As usual, variables in definite clauses are implicitly universally quantified.

We say that a predicate symbol p *occurs extensionally in* formula G if

(a) G is $p(\overline{t})$, or

(b) G is $G_1 \wedge G_2$ or $G_1 \vee G_2$, and p occurs extensionally in G_1 or G_2.

(c) G is $\exists x G_1$, and p occurs extensionally in G_1.

In following sections, we will define semantics for λProlog. We will take advantage of the following situation: Since logic programs compute extensions of predicates, and relations between arguments of predicate symbols constitute extensions of predicates, we don't need extensions of terms until we meet extensional occurrences of predicate symbols in the definition of satisfaction of formulas.

3 Extensional Domains

We shall have extensive use of structures of partially ordered set (poset) in this paper. It is convenient to study structures of this type in a more general setting.

If P is a non-empty set with binary relation $\subseteq_P$ which is a *partial ordering* (i.e., the relation $\subseteq_P$ is reflexive, transitive and anti-symmetric), we say that P is a *partially ordered set*, or simply a *poset.*

There are many examples of posets, but we are interested only in those satisfying special requirement below. In particular, we note that any non-empty set A can be considered a poset under the identity relation where $x \subseteq_A y$ iff $x = y$. We call this type of poset *discrete.*

Let x and y be elements of a poset P. We say that y is an *upper bound* of x in P if $x \subseteq_P y$. If x is an element of P such that every element of P is an upper bound of x, we say that x is a *bottom* element of P. If this element exists it is clearly unique, it is denoted in the form $\perp_P$. An element x in the poset P is *maximal* if whenever $x \subseteq_P y$, then $x = y$, or equivalently when the only upper bound of x is x itself. If P is a discrete poset, then every element of P is maximal.

Let x and y be elements of a poset P. We say that x and y are *consistent* in P if they have a common upper bound, i.e., there is an element z in P such that $x \subseteq_P z$ and $y \subseteq_P z$. A subset M of P is *consistent* if any two elements in M are consistent.

Let M be a subset of the poset P. If x is an element of P which is an upper bound of every element of M we say that x is an *upper bound* of M. We say that M is *directed* if every finite subset of M has an upper bound in M. An element x of P is the *least upper bound* (*lub*) of M in P if it satisfies the following two conditions: (a) x is an upper bound of M; (b) if y is any upper bound of M, then y is an upper bound of x. It is clear that the least upper bound of a set M is unique if it exists. We shall use the notation $\sqcup_P M$ to denote the least upper bound of a subset M of a poset P whenever such element exists.

There is a dual notion that also plays a role in our discussion. We say that an element x of a poset P is a *lower bound* of a subset M if x is a lower bound of every element in M. An element x of P is the *greatest lower bound* (*glb*) of M if it satisfies the following conditions: (a) x is a lower bound of M; (b) if y is any lower bound of M, then y is a lower bound of x. We shall use the notation $\sqcap_P M$ to denote the greatest lower bound of a subset M of a poset P whenever such element exists.

If M is the two element set $\{x, y\}$, then we write $x \sqcup_P y$ for $\sqcup_P M$ and $x \sqcap_P y$ for $\sqcap_P M$.

A *lattice* is a poset in which each two element subset has both an lub and a glb. Thus in a lattice L for any $x, y \in L$ both $x \sqcup_L y$ and $x \sqcap_L y$ exist. A lattice L is a *complete lattice* if $\sqcup_L M$ and $\sqcap_L M$ exist for every subset M of L.

A poset D is a *domain* if every consistent subset of D has a least upper bound. Since the empty set is consistent, it follows that D has a bottom element.

Lemma 3.1 *†domdglb:*[1] *Let D be a domain. If M is a non-empty subset of D, then M has a greatest lower bound.* □

A *transformation* of a poset P is a total mapping T from P to P which satisfies the following monotonicity condition for all x and y in P: if $x \subseteq_P y$, then $Tx \subseteq_P Ty$. If T is a transformation on P, and x is an element of P such that $Tx \subseteq_P x$, we say that x is *closed under* T. If $Tx = x$ we say that x is a fixed point of T.

Theorem 3.2 *†dflfp: Let T be a transformation on the domain D. There is a unique element x in D such that:*
(a) The element x is a fixed point of T.
(b) If y is an element of D which is closed under T, then y is an upper bound of x. □

We call the unique element x in the above theorem the *least fixed point* of the transformation T. This least fixed point has a more convenient formulation by recursion using transfinite ordinals:

Theorem 3.3 *†dtlfpo: Let T be a transformation of a domain D. Define T^ν for all ordinals ν: for $x \in D$,*
(a) $T^0 x = x$
(b) $T^{\nu+1} x = T(T^\nu x)$
(c) $T^\nu x = \sqcup_{\mu<\nu} T^\mu x$ $\quad$ ν is a limit ordinal
Then T has a least fixed point given by $T^\gamma \bot_D$ where γ is the first ordinal such that $T^\gamma \bot_D = T^{\gamma+1} \bot_D$. We call the ordinal γ closure ordinal *for T.* □

Let T be a transformation of a domain D. We say that T is *continuous* if $T(\sqcup_D M) = \sqcup_D T(M)$, for every directed subset M of D.

Theorem 3.4 *†tcontlfp: Let T be a continuous transformation of a domain D. Then the least fixed point of T is $T^\omega \bot_D$.* □

Now a few constructions of posets will be discussed.

Let P_1 and P_2 be disjoint posets. $P_1 \cup P_2$ is a poset $P = P_1 \cup P_2$ such that for all $x, y \in P$, $x \subseteq_P y$ if $x \subseteq_{P_1} y$ or $x \subseteq_{P_2} y$. $P_1 \times P_2$ is a poset $P = P_1 \times P_2$ where for all $x, y \in P$, $x \subseteq_P y$ if $x^1 \subseteq_{P_1} y^1$ and $x^2 \subseteq_{P_2} y^2$.

[1]We use the label starting with †only for editing purpose. In the following, ignore them.

Let S be a set, and P a poset. $S \rightarrow P$ is a poset F such that for all $f, g \in F$, $f \subseteq_F g$ if for all $s \in S$, $f(s) \subseteq_P g(s)$.

Theorem 3.5 *Let S be a set, and D a domain. Then $S \rightarrow D$ is also a domain.*

Proof Let M be a consistent subset of $S \rightarrow D$, and g be a mapping in $S \rightarrow D$ defined as: for all $s \in S$, $g(s) = \sqcup_D\{f(s) : f \in M\}$. Then it is easy to see that $g = \sqcup_{S \rightarrow D} M$. □

In the following, given a poset the poset subscript on $\subseteq$ and $\sqcup$ and $\sqcap$ will be omitted except when its value is essential in a discussion and cannot be inferred from the context.

Let $\mathcal{B}$ be the set of boolean values $\mathbf{T}$ and $\mathbf{F}$ where $\mathbf{F} \subseteq_{\mathcal{B}} \mathbf{T}$. Then $\mathcal{B}$ is a domain. We shall write $\vee$ and $\wedge$ for $\sqcup_{\mathcal{B}}$ and $\sqcap_{\mathcal{B}}$, respectively.

Let A be an arbitrary set. We can consider A a discrete poset. A *predicate P over A of type $\alpha_1, \cdots, \alpha_n \rightarrow o$* is a mapping in $A_{\alpha_1} \times \cdots \times A_{\alpha_n} \rightarrow \mathcal{B}$, or equivalently a subset of $A_{\alpha_1} \times \cdots \times A_{\alpha_n}$. And we consider truth values $\mathbf{T}$ and $\mathbf{F}$ as *null-ary* predicates over A of type $() \rightarrow o$ such that $\mathbf{T}() \equiv \mathbf{T}$ and $\mathbf{F}() \equiv \mathbf{F}$, respectively. More generally, we define predicates $\mathbf{T}^A_{\alpha_1,\cdots,\alpha_n}$ for each list $\alpha_1, \cdots, \alpha_n$ of types where $n \geq 0$ as $A_{\alpha_1} \times \cdots \times A_{\alpha_n}$. We write $\Phi(A)$ for the set of all predicates over A.

Lemma 3.6 *$\Phi(A)_{\overline{\alpha} \rightarrow o}$ is a domain.* □

Given two predicates $P, Q \in \Phi(A)$, it is obvious that $P \subseteq Q$ if P and Q are of same type and P is a subset of Q.

Theorem 3.7 *Let S be a set, and F the set of all good mappings in $S \rightarrow \Phi(A)$. Then F is a domain.* □

Given a list $\alpha_1, \cdots, \alpha_n$ of types and $i \in [n]$, *i-th identity function I^i* is a function in $A_{\alpha_1} \times \cdots \times A_{\alpha_n} \rightarrow A_{\alpha_i}$ such that for all $(a_1, \cdots, a_n) \in A_{\alpha_1} \times \cdots \times A_{\alpha_n}$, $I^i(a_1, \cdots, a_n) = a_i$.

Operations on $\Phi(A)$ can be introduced as follows.

Let $P, Q \in \Phi(A)_{\overline{\alpha} \rightarrow o}$. Then predicates $P \wedge Q$ and $P \vee Q$ defined by

(*Disjunction*) $\quad P \vee Q = P \sqcup Q$ and

(*Conjuction*) $\quad P \wedge Q = P \sqcap Q$

are in $\Phi(A)_{\overline{\alpha} \rightarrow o}$.

Let $P \in \Phi(A)_{\overline{\alpha} \rightarrow o}$. Then a predicate βP defined by

(*Adjoining of a variable$_\beta$*) $\quad \beta P(b, \overline{a}) \equiv P(\overline{a})$

is in $\Phi(A)_{\beta, \overline{\alpha} \rightarrow o}$.

Let $P \in \Phi(A)_{\beta, \overline{\alpha} \rightarrow o}$. Then a predicate ∂P defined by

(*Projection*) $\quad \partial P(\overline{a}) \equiv \exists b \in A_\beta \cdot P(b, \overline{a})$

$\equiv$ There exists a $b \in A_\beta$ such that $P(b, \overline{a})$.

is in $\Phi(A)_{\overline{\alpha}\to o}$.
Given a list of types $\alpha_1, \cdots, \alpha_n$, and a function $\pi : [n] \to [n]$,
and $P \in \Phi(A)_{\alpha_{\pi(1)},\cdots,\alpha_{\pi(n)}\to o}$, the predicate πP defined by
(*Simple substitution*) $\qquad \pi P(\overline{a}) \equiv P(I^{\pi(1)}(\overline{a}), \cdots, I^{\pi(n)}(\overline{a}))$
is in $\Phi(A)_{\overline{\alpha}\to o}$.

Theorem 3.8 *Let $\wp$ be a subset of $\Phi(A)$ where $\wp$ is closed under conjunction and disjunction. Then $\wp_{\overline{\alpha}\to o}$ is a domain.* □

Given a subset $\wp$ of $\Phi(A)$, we can extend $\wp$ to the unique *extensional domain* $\wp' = \{\wp'_\alpha\}_\alpha$ *based on* A:

$$\wp' = \wp \cup A \uparrow (\mathcal{T} - \mathcal{T}^o)$$

In a situation where a set A is given, we shall use $\wp$ to mean $\wp'$.

Let $\wp$ be an extensional domain. We say that $\wp$ is *elementarily closed* if $\wp$ contains predicates $\mathbf{T}$ and $\mathbf{F}$ and is closed under conjunction, disjunction, projection, adjoining of a variable, and simple substitution.

Given any extensional domain $\wp$, we call an extensional domain $\wp^*$ as a *elementary closure* of $\wp$ if $\wp^*$ is a minimal extension of $\wp$ which is elementarily closed. We can also give inductive definition of $\wp^*$.

Definition Given a subset $\wp$ of $\Phi(A)$, for each ordinal ν, $\wp_\nu$ is defined by:
(a) $\wp_0 = \wp$
(b) $P \in \wp_{\nu+1}$ iff $P \in \wp_\nu$, or
P can be obtained by conjunction or disjunction or projection or adjoining of a variable, or simple substitution from predicates in $\wp_\nu$.
(c) ν is a limit ordinal.
$\wp_\nu = \bigcup_{\mu<\nu} \wp_\mu$.
Finally $\wp^*$ defined to be $\bigcup_\nu \wp_\nu$.

Note that $P \in \wp^*$ iff $P \in \wp_\nu$ for some ordinal ν. □

4 General Model Theoretic Semantics

In this Section we build model-theoretic semantics for the language $\mathcal{L}$. As introduced in Section 1 we need a non-extensional model to prove that the resolution system in type theory is complete. The model in [2] is in a sense non-extensional. But it doesn't provide an adequate notion of "general" non-extensional model for our purpose: Domain is defined by indexing extension of the element in it by wfts.

The indexed entity like $\langle t, p \rangle$ is called a V-complexe where V is a truth value evaluation of formulas. So only one kind of domain is used in [2], since the set of all wfts is predetermined given a language $\mathcal{L}$.

In [2], in order to define the domain of interpretation we need a semivaluation function V, as above, which evaluates proposional formulas to **T** or **F**. The definition of domain or the evaluation of terms is mutually recursive with the definition of evaluation of formulas.

Now we generalize Andrews model to a model where we index the extension by an element from an arbitrary set which we call universe. Since our language $\mathcal{L}$ is based on λ-calculus and application is a basic operation of the λ-calculus, any model of $\mathcal{L}$ should be an applicative structure which is a λ-model.

Definition Let A be an arbitrary set and $\cdot$ a binary operation over A such that for all $\alpha, \beta \in \mathcal{T}$, for all $a \in A_{\alpha\to\beta}, b \in A_\alpha$, $a \cdot b$ is an element in A_β. Then $\mathcal{A} = \langle A, \cdot \rangle$ is said to be an *applicative structure.* An *assignment into* a set A is a good mapping $\varphi : \Delta \to A$. A λ-*model* is a triple $\langle A, \cdot, V \rangle$ such that $\langle A, \cdot \rangle$ is an applicative structure and V a binary function such that for each assignment φ into A and term t_α, $V_\varphi t_\alpha \in A_\alpha$. We call the function V a *valuation function* in A. □

Definition A *universe* is a nonempty set D of objects each of which is assigned a type symbol from the set $\mathcal{T}$ in such a way that every object in $D_{\alpha\to\beta}$ is a function from D_α to D_β for all type symbols α and β. A *pre-interpretation* $\mathcal{F}$ of the language $\mathcal{L}$ is a pair $\langle D, J \rangle$ where D is a universe, and J is a function which maps each constant $c_\alpha \in \Sigma$ of type α to an element of D_α. We say that $\mathcal{F}$ is *based on* D. An assignment into a pre-interpretation is an assignment into the universe of the pre-interpretation. □

Note that $D_{\alpha\to\beta}$ is some collection of functions mapping D_α into D_β, i.e. $D_{\alpha\to\beta} \subseteq D_\alpha \to D_\beta$.

A pre-interpretation $\mathcal{F} = \langle D, J \rangle$ is said to be *general* iff there is a binary function $\mathsf{V}^{\mathcal{F}} = \mathsf{V}$ such that for each assignment φ and term t_α, $\mathsf{V}_\varphi t_\alpha \in D_\alpha$, and the following conditions are satisfied for each assignment φ and all terms:

(a) if $x \in \Delta$, then $\mathsf{V}_\varphi x = \varphi x$. (b) if $c \in \Sigma$, then $\mathsf{V}_\varphi c = Jc$.

(c) $\mathsf{V}_\varphi(ft) = (\mathsf{V}_\varphi f)\mathsf{V}_\varphi t$ (the value of the function $\mathsf{V}_\varphi f$ at the argument $\mathsf{V}_\varphi t$)

(d) $\mathsf{V}_\varphi(\lambda x_\alpha t_\beta) = \boldsymbol{\lambda} d \in D_\alpha \cdot \mathsf{V}_{\varphi[d/x]} t_\beta$ i.e. that function from D_α into D_β whose value for each argument $d \in D_\alpha$ is $\mathsf{V}_{\varphi[d/x]} t_\beta$.

Lemma 4.1 †*svfuni: If a pre-interpretation $\mathcal{F}$ is general, the function $\mathsf{V}^{\mathcal{F}}$ is uniquely determined.*

Proof By induction on the definition of terms. □

We call the function $\mathsf{V}^{\mathcal{F}}$ the *connotational valuation function* of terms in the pre-interpretation $\mathcal{F}$. $\mathsf{V}^{\mathcal{F}}_\varphi t$ is called the *value* of t in $\mathcal{F}$ wrt φ. We sometimes write $\mathsf{V}^{\mathcal{F}}_\varphi$ as V_φ, as $\mathsf{V}^{\mathcal{F}}$, or as V, when pre-interpretation or assignment is clear from context, or irrelevant.

A universe D is called a *standard universe* iff for all α and β, $D_{\alpha\to\beta}$ is the set of all functions from D_α into D_β, i.e. $D_{\alpha\to\beta} = D_\alpha \to D_\beta$. We call a pre-interpretation as a *standard pre-interpretation* if its universe is standard. Clearly a standard pre-interpretation is general.

Lemma 4.2 *Let a pre-interpretation $\mathcal{F}$ be general, t a term, and φ and ψ assignments which agree on all free variables of t. Then $\mathsf{V}_\varphi t = \mathsf{V}_\psi t$.*

Proof By induction on the construction of t. □

It is clear that if t is a c-term, then $\mathsf{V}^{\mathcal{F}} t$ may be considered meaningful without regard to any assignment. In this case, $\mathsf{V}^{\mathcal{F}} t$ is called the *connotation* of t in $\mathcal{F}$ and written as t'. Obviously for a general universe D, $\langle D, \cdot, \mathsf{V} \rangle$ where $\cdot$ is interpreted as a functional application is a general λ-model, but in a pre-interpretation logic symbols such as logical operators and predicate constants are not fully interpreted. So we call it a pre-interpretation.

Definition Let D be a universe. A *semivaluation* of D is a function V with domain D_o and range the set $\mathcal{B}$ of truth values such that the following properties hold: for all $c_o, d_o, f_{\alpha\to o} \in D$,

(a) $V(\top') = \mathbf{T}$.

(b) $V(\vee' c_o d_o) = V(c_o) \vee V(d_o)$.

(c) $V(\wedge' c_o d_o) = V(c_o) \wedge V(d_o)$.

(d) $V(\exists'_\alpha f_{\alpha\to o}) = \mathbf{T}$ iff there is some $e \in D_\alpha$ such that $V(f_{\alpha\to o} e) = \mathbf{T}$. □

Lemma 4.3 *Let φ be an assignment into D. Then for all goal formula G, $V(\mathsf{V}_\varphi G) = V(\mathsf{V}_\varphi(\rho(G)))$.*

Proof $\mathsf{V}_\varphi G = \mathsf{V}_\varphi \rho(G)$. □

Definition Given a universe D and a semivaluation V of D, we define the *set $\mathcal{D}$ of V-complexes based on D* as follows: For each type γ we define the set $\mathcal{D}_\gamma$ of V-complexes$_\gamma$ as follows by induction on γ:

(a) $\mathcal{D}_o = \{\langle d, Vd \rangle : d \in D_o\}$.

(b) $\mathcal{D}_\alpha = \{\langle d, d \rangle : d \in D_\alpha\}$ when $\alpha \in \mathcal{T}_0 - \{o\}$.

(c) $\mathcal{D}_{\alpha\to\beta} = \{\langle f, \boldsymbol{p} \rangle : f \in D_{\alpha\to\beta}$ and $\boldsymbol{p} : \mathcal{D}_\alpha \to \mathcal{D}_\beta$ such that for $a \in \mathcal{D}_\alpha$, $\boldsymbol{p}a = \langle f a^1, \boldsymbol{r} \rangle$ for some $\boldsymbol{r}\}$.

We say that $\mathcal{D}$ is the set of V-complexes *based on D*. □

Lemma 4.4 †*dvduni: Given a universe D and a semivaluation V of D, the set $\mathcal{D}$ of V-complexes based on D is unique.*

Proof We prove by induction on γ: For each $d \in D_\gamma$, there is only one $\boldsymbol{p}$ such that $\langle d, \boldsymbol{p} \rangle \in \mathcal{D}_\gamma$.

For $\gamma \in \mathcal{T}_0$ it is obvious. For $\gamma = \alpha \to \beta$, let $f \in D_{\alpha\to\beta}$. Assume $\langle f, \boldsymbol{p} \rangle, \langle f, \boldsymbol{q} \rangle \in \mathcal{D}_{\alpha\to\beta}$ to show $\boldsymbol{p} = \boldsymbol{q}$. Then $\boldsymbol{p}, \boldsymbol{q} : \mathcal{D}_\alpha \to \mathcal{D}_\beta$. For any $a \in \mathcal{D}_\alpha$, $\boldsymbol{p}a = \langle f a^1, \boldsymbol{r}_1 \rangle$ and $\boldsymbol{q}a = \langle f a^1, \boldsymbol{r}_2 \rangle$. By IH $\boldsymbol{r}_1 = \boldsymbol{r}_2$. Therefore $\boldsymbol{p}a = \boldsymbol{q}a$ for all $a \in \mathcal{D}_\alpha$. So $\boldsymbol{p} = \boldsymbol{q}$. □

Definition Let $\mathcal{D}$ be a set of V-complexes. Then we define the *applicative operation* $\star$ of type $(\alpha \to \beta), \alpha \to \beta$: For $a \in \mathcal{D}_{\alpha\to\beta}$ and $b \in \mathcal{D}_\alpha$, $a \star b$ is defined to be $a^2 b$. The operation $\star$ is left associative. □

Let $a \in \mathcal{D}_{\alpha_1,\cdots,\alpha_n\to\beta}$ and $b_i \in \mathcal{D}_{\alpha_i}$ for $i \in [n]$. Then by definition of $\mathcal{D}$ it is easy to see that $a \star b_1 \star \cdots \star b_n \in \mathcal{D}_\beta$.

Lemma 4.5 *$\langle \mathcal{D}, \star \rangle$ is an applicative structure.* □

Lemma 4.6 [2] *†dev: For each $d_\gamma \in D_\gamma$, there exists a $\boldsymbol{v}$ such that $\langle d_\gamma, \boldsymbol{v} \rangle \in \mathcal{D}_\gamma$.*

Proof We choose $\boldsymbol{v}$ as a function of d_γ by induction on γ, and show that $\langle d_\gamma, \boldsymbol{v}(d_\gamma) \rangle \in \mathcal{D}_\gamma$. This is trivial when γ is in $\mathcal{T}_0$. If $\gamma = \alpha \to \beta$, let $\boldsymbol{v}(d_{\alpha\to\beta})a = \langle d_{\alpha\to\beta}a^1, \boldsymbol{v}(d_{\alpha\to\beta}a^1) \rangle$ for each $a \in \mathcal{D}_\alpha$. □

Let $\mathcal{F} = \langle D, J \rangle$ be a pre-interpretation, V a semivaluation of D, and $\mathcal{D}$ a set of V-complexes based on $\mathcal{F}$. Then we can introduce an isomorphic mapping κ between D and $\mathcal{D}$: For all $d \in D$, $\kappa d = \langle d, \boldsymbol{v}(d) \rangle$. Then for any $a \in \mathcal{D}$, $\kappa a^1 = a$, and for any mapping χ whose values are in $\mathcal{D}$, $\chi^1 \circ \kappa = \chi$.

Lemma 4.7 *For the above mapping κ the following holds:*
For all $f \in D_{\alpha\to\beta}, d \in D_\alpha$, $(\kappa f) \star (\kappa d) = \kappa(fd)$.

Proof

$(\kappa f) \star (\kappa d) = (\kappa f)^2(\kappa d) = (\boldsymbol{v}(f))\langle d, \boldsymbol{v}(d) \rangle = \langle fd, \boldsymbol{v}(fd) \rangle = \kappa(fd)$. □

Definition Let $\mathcal{D}$ be a set of V-complexes. We can define a binary mapping $\mathcal{V}$ such that for all assignment φ into $\mathcal{D}$, $\mathcal{V}_\varphi : T(\Sigma) \to \mathcal{D}$, and for all $t \in T(\Sigma)$, $\mathcal{V}^1_\varphi = \mathsf{V}_{\varphi^1} t$. □

Lemma 4.8 *†ksc: Let φ be an assignment into D. Then for all term t, $\kappa \mathsf{V}_\varphi t = \mathcal{V}_{\varphi\circ\kappa} t$.*

Proof $\varphi \circ \kappa$ is an assignment into $\mathcal{D}$ and $(\varphi \circ \kappa)^1 = \varphi$.

$\mathcal{V}_{\varphi\circ\kappa} t = \langle \mathsf{V}_\varphi t, \boldsymbol{v}(\mathsf{V}_\varphi t) \rangle = \kappa \mathsf{V}_\varphi t$. □

Lemma 4.9 *†gdcvuni: Let D be a general universe and $\mathcal{D}$ a set of V-complexes. Then there is the unique $\mathcal{V}$ satisfying that for all $t_\gamma \in T(\Sigma)$ and assignment φ into $\mathcal{D}$, $\mathcal{V}_\varphi t_\gamma \in \mathcal{D}_\gamma$.*

Proof Follows from Lemmas 4.8, 4.1. □

Theorem 4.10 *If D is a general universe and $\mathcal{D}$ a set of V-complexes, then $\langle \mathcal{D}, \star, \mathcal{V} \rangle$ is a λ-model.*

Proof Follows from Lemma 4.9. □

From now on we always assume that our universe D is general, and we are given a fixed set $\mathcal{D}$ of V-complexes based on D for some semivaluation V of D.

Lemma 4.11 $\mathcal{V}^2_\varphi(\lambda x_\alpha t) = \boldsymbol{\lambda} a \in \mathcal{D}_\alpha \cdot \mathcal{V}_{\varphi[a/x_\alpha]} t$.

Proof $\mathcal{V}^2_\varphi(\lambda x_\alpha t) = \boldsymbol{v}(\mathsf{V}_{\varphi^1}(\lambda x_\alpha t))$. For $a \in \mathcal{D}_\alpha$,

$\mathcal{V}^2_\varphi(\lambda x_\alpha t)a = \langle \mathsf{V}_{\varphi^1}(\lambda x_\alpha t)a^1, \boldsymbol{v}(\mathsf{V}_{\varphi^1}(\lambda x_\alpha t)a^1) \rangle = \langle \mathsf{V}_{\varphi^1[a^1/x_\alpha]} t, \boldsymbol{v}(\mathsf{V}_{\varphi^1[a^1/x_\alpha]} t) \rangle = \kappa \mathsf{V}_{\varphi'[a^1/x_\alpha]} t = \mathcal{V}_{\varphi[a/x_\alpha]} t$. □

[2]It is easy to see that given $\mathcal{D}$, the function $\boldsymbol{v}$ in the proof of this lemma is unique. Note also that we use boldface character for extension.

Lemma 4.12 *Let $ft_1 \cdots t_n$ be a wft in $T(\Sigma)$ where $n > 0$.*
Then $\mathcal{V}_\varphi(ft_1 \cdots t_n) = (\mathcal{V}_\varphi f) \star (\mathcal{V}_\varphi t_1) \star \cdots \star (\mathcal{V}_\varphi t_n)$.
Proof By induction on n. □

Lemma 4.13 *†cvapp: Let G_1 and G_2 be goal formulas.*
(a) $\mathcal{V}^2_\varphi(G_1 \vee G_2) = \mathcal{V}^2_\varphi G_1 \vee \mathcal{V}^2_\varphi G_2$.
(b) $\mathcal{V}^2_\varphi(G_1 \wedge G_2) = \mathcal{V}^2_\varphi G_1 \wedge \mathcal{V}^2_\varphi G_2$.
(c) $\mathcal{V}^2_\varphi(\exists x_\alpha G_1) = \mathbf{T}$ iff there exists an $a \in \mathcal{D}_\alpha$ such that $\mathcal{V}^2_{\varphi[a/x_\alpha]} G_1 = \mathbf{T}$.
Proof (a) $\mathcal{V}^2_\varphi(G_1 \vee G_2) = V(\vee'(\mathsf{V}_{\varphi^1} G_1)\mathsf{V}_{\varphi^1} G_2) = V(\mathsf{V}_{\varphi^1} G_1) \vee V(\mathsf{V}_{\varphi^1} G_2) = \mathcal{V}^2_\varphi G_1 \vee \mathcal{V}^2_\varphi G_2$.
(b) Similarly as in (a).
(c) Let $f = \mathsf{V}_{\varphi^1}(\lambda x_\alpha G_1)$. Then $\mathcal{V}^2_\varphi(\exists x_\alpha G_1) = V(\exists'_\alpha f) = \mathbf{T}$ iff there exists a $d \in D_\alpha$ such that $V(fd) = \mathbf{T}$. $fd = \mathsf{V}_{\varphi^1}(\lambda x_\alpha G_1)d = \mathsf{V}_{\varphi^1[d/x_\alpha]} G_1 = \mathbf{T}$. Therefore there is an $a = \kappa d$ such that $\mathcal{V}^2_{\varphi[a/x_\alpha]} G_1 = \mathbf{T}$. □

Definition Given a universe D, we define a *primitive extensional domain* E_α for $\alpha \in \mathcal{T}_0$:
(a) $E_o = \mathcal{B}$.
(b) $E_\alpha = D_\alpha$ for $\alpha \in \mathcal{T}_0 - \{o\}$. □

Definition Given an $a \in \mathcal{D}_{\alpha_1,\cdots,\alpha_n \to \beta}$ where $n \geq 0$ and $\beta \in \mathcal{T}_0$, we define a mapping $a^\odot$ in $D_{\alpha_1} \to \cdots \to D_{\alpha_n} \to E_\beta$ by induction on n:
(a) When $n = 0$, $a^\odot = a^2$.
(b) When $n > 0$, $a^\odot = \boldsymbol{\lambda} d_1 \in D_{\alpha_1} \cdot (a \star \kappa d_1)^\odot$. □

Lemma 4.14 *†eapp: Let $a \in \mathcal{D}_{\alpha_1,\cdots,\alpha_n \to \beta}$ where $n > 0$ and $\beta \in \mathcal{T}_0$. Then for all $d_i \in D_{\alpha_i}$, $i \in [n]$, $a^\odot d_1 \cdots d_n = (a \star \kappa d_1 \star \cdots \kappa d_n)^2$.*
Proof By induction on n. When $n = 1$. $a^\odot d_1 = (a \star \kappa d_1)^\odot = (a \star \kappa d_1)^2$.
When $n > 1$.

$$\begin{aligned} a^\odot d_1 \cdots d_n &= (a \star \kappa d_1)^\odot d_2 \cdots d_n \\ &= (a \star \kappa d_1 \star \kappa d_2 \star \cdots \star \kappa d_n)^2 \qquad \text{by IH.} \end{aligned}$$

□

Lemma 4.15 *†extsam: Let $a \in \mathcal{D}_{\alpha_1,\cdots,\alpha_n \to \beta}$ where $n \geq 0$ and $\beta \in \mathcal{T}_0 - \{o\}$. Then $a^\odot = a^1$.*
Proof By induction on n. When $n = 0$. $a^\odot = a^2 = a^1$ by the definition of $\mathcal{D}_\beta$. When $n > 0$. For all $d_1 \in D_{\alpha_1}$, $(a \star \kappa d_1)^\odot = a^1 d_1$ by IH. Therefore $a^\odot = a^1$. □

Definition Let $\mathcal{F} = \langle D, J \rangle$ be a general pre-interpretation and V a semivaluation of D. An *$\mathcal{L}$-structure* $\mathcal{A}$ is a pair $\langle \mathcal{D}, J \rangle$ such that $\mathcal{D}$ is a set of V-complexes based on D. We say that $\mathcal{A}$ is *based on $\mathcal{F}$* or *on D*. An *assignment into $\mathcal{A}$* is an assignment into $\mathcal{D}$. □

Definition [3] Let $\mathcal{A} = \langle \mathcal{D}, J \rangle$ be an $\mathcal{L}$-structure, φ an assignment into $\mathcal{A}$. When F is a formula in $\mathcal{L}$, we write $\mathcal{A} \models F[\varphi]$ to say that $\mathcal{A}$ *satisfies* F wrt φ.

(a) When $s_\alpha, t_\alpha \in T(\Sigma)$, $\mathcal{A} \models s_\alpha = t_\alpha[\varphi]$ iff $\mathcal{V}_\varphi s_\alpha = \mathcal{V}_\varphi t_\alpha$,
$\mathcal{A} \models s_\alpha \equiv t_\alpha[\varphi]$ iff $(\mathcal{V}_\varphi s_\alpha)^\odot = (\mathcal{V}_\varphi t_\alpha)^\odot$.

(b) When G is a goal formula, $\mathcal{A} \models G[\varphi]$ iff $\mathcal{V}^2_\varphi G = \mathbf{T}$.

(c) When $A \leftarrow G$ is a definite clause, $\mathcal{A} \models A \leftarrow G[\varphi]$ iff $\mathcal{A} \models A[\varphi]$ whenever $\mathcal{A} \models G[\varphi]$.

We write $\mathcal{A} \models F$ to say that a formula F is *valid* in $\mathcal{A}$ if $\mathcal{A} \models F[\varphi]$ for all assignments φ into $\mathcal{A}$. Given a set of definite clause $\mathcal{P}$, we say that $\mathcal{A}$ is a *model* or *D-model* for $\mathcal{P}$, and write $\mathcal{A} \models \mathcal{P}$, if each definite clause in $\mathcal{P}$ is valid in $\mathcal{A}$. Given a closed goal formula G, we say that G is a *logical consequence* of $\mathcal{P}$, and write $\mathcal{P} \models G$ if G is valid in all models of $\mathcal{P}$. □

Lemma 4.16 *†atom: Let $\mathcal{A} = \langle \mathcal{D}, J \rangle$ be an $\mathcal{L}$-structure, φ an assignment into $\mathcal{A}$ and $p(t_1, \cdots, t_n)$ an atom. Then*

(a) $\mathcal{A} \models p(t_1, \cdots, t_n)[\varphi]$ iff $\langle \mathsf{V}_{\varphi^1} t_1, \cdots, \mathsf{V}_{\varphi^1} t_n \rangle \in (\mathcal{V}_\varphi p)^\odot$.

(b) $\mathcal{A} \models p(t_1, \cdots, t_n)[\varphi]$ iff $\langle \mathsf{V}_{\varphi^1} t_1, \cdots, \mathsf{V}_{\varphi^1} t_n \rangle \in (J \circ \kappa(p))^\odot$ if p is a constant or $\langle \mathsf{V}_{\varphi^1} t_1, \cdots, \mathsf{V}_{\varphi^1} t_n \rangle \in (\varphi p)^\odot$ if p is a variable.

Proof

$$\begin{aligned}
\mathcal{V}^2_\varphi(p(t_1, \cdots, t_n)) &= (\mathcal{V}_\varphi p \star \mathcal{V}_\varphi t_1 \star \cdots \star \mathcal{V}_\varphi t_n)^2 \\
&= (\mathcal{V}_\varphi p \star \kappa \mathsf{V}_{\varphi^1} t_1 \star \cdots \star \kappa \mathsf{V}_{\varphi^1} t_n)^2 && \text{by Corollary 4.8.} \\
&= (\mathcal{V}_\varphi p)^\odot (\mathsf{V}_{\varphi^1} t_1, \cdots, \mathsf{V}_{\varphi^1} t_n) && \text{by Lemma 4.14.}
\end{aligned}$$

□

Lemma 4.17 *†lop: Let $\mathcal{A} = \langle \mathcal{D}, J \rangle$ be an $\mathcal{L}$-structure, φ an assignment into $\mathcal{A}$. Then*

(a) $\mathcal{A} \models \top[\varphi]$.

(b) $\mathcal{A} \models G_1 \vee G_2[\varphi]$ iff $\mathcal{A} \models G_1[\varphi]$ or $\mathcal{A} \models G_2[\varphi]$.

(c) $\mathcal{A} \models G_1 \wedge G_2[\varphi]$ iff $\mathcal{A} \models G_1[\varphi]$ and $\mathcal{A} \models G_2[\varphi]$.

(d) $\mathcal{A} \models \exists x_\alpha G[\varphi]$ iff there is an $a \in \mathcal{D}_\alpha$ such that $\mathcal{A} \models G[\varphi[a/x_\alpha]]$.

Proof (a) By (a) in the definition of semivaluation. (b), (c), and (d) follow from Lemma 4.13. □

Definition Let $\langle D, J \rangle$ be a general pre-interpretation. Then we write $\Pi(D)$ for the *D-base* which is defined to be the set $\{p(a_1, \cdots, a_n) : p \in \Pi_{\alpha_1, \cdots, \alpha_n \to o}$ and $a_i \in D_{\alpha_i}$ for all $i \in [n]\}$. □

[3]Note that in this definition the symbol for satisfaction in $\mathcal{A}$ is the small $\models$. The normal size $\models$ is used for another definition of satisfaction which is defined later in this paper.

A subset $\mathcal{K}$ of $\Pi(D)$ induces a unique mapping $I_{\mathcal{K}}$ in $\Pi \to \Phi(D)$ as follows: for all $\overline{d} \in D$, $\langle\overline{d}\rangle \in I_{\mathcal{K}}(p)$ iff $p(\overline{d}) \in \mathcal{K}$. Let $\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \Pi(D)$, then it is easy to see that $I_{\mathcal{K}_1} \subseteq_{\Pi\to\Phi(D)} I_{\mathcal{K}_2}$. Sometimes given $\mathcal{K} \subseteq \Pi(D)$, we write simply $\mathcal{K}$ to mean the mapping $I_{\mathcal{K}}$.

Definition Let D be a general universe and S a subset of D_o.

(1) S is *upward saturated* if

a) $c \in S$ implies $\vee' cd, \vee' dc \in S$ for $d \in D_o$.

b) $c, d \in S$ implies $\wedge' cd \in S$.

c) $f_{\alpha\to o} d_\alpha \in S$ implies $\exists'_\alpha f_{\alpha\to o} \in S$

(2) S is *downward saturated* if

a) If $\vee' cd \in S$, then $c \in S$ or $d \in S$.

b) If $\wedge' cd \in S$, then $c, d \in S$.

c) If $\exists'_\alpha f_{\alpha\to o} \in S$, then there is a $d \in D_\alpha$ such that $f_{\alpha\to o} d \in S$.

(3) S is *saturated* if S is both downward and upward saturated. □

Lemma 4.18 *Let $S \subseteq D_o$. There is a smallest upward saturated set extending S.*

Proof Let $\mathcal{C}$ be the collection of upward saturated set extending S. $\mathcal{C}$ is not empty, since $D_o \in \mathcal{C}$. So $\cap\mathcal{C}$ exists. It is easy to check that it is upward saturated. It fulfills the other considerations, by definition. □

Definition For $S \subseteq D_o$, the smallest upward saturated set extending S is called the *upward saturated closure* of S, and is denoted as S^U. □

Lemma 4.19 *Let $S_1 \subseteq S_2 \subseteq D_o$. Then $S_1^U \subseteq S_2^U$.*

Proof Let $\mathcal{C}_1$ be the collection of upward saturated sets extending S_1, and similarly for $\mathcal{C}_2$. Then $\mathcal{C}_2 \subseteq \mathcal{C}_1$; hence $S_1^U = \cap\mathcal{C}_1 \subseteq \cap\mathcal{C}_2 = S_2^U$. □

Lemma 4.20 *If S is downward saturated, S^U is saturated.*

Proof It is enough to show that S^U is downward saturated. Say $c \notin S^U$; we show $\wedge' cd \notin S^U$. The other cases are similar. If $c \notin S^U$, we cannot have $\wedge' cd \in S$, since S is downward saturated. So we would have $c \in S$, but $S \subseteq S^U$. Hence $S \subseteq S^U - \{\wedge' cd\}$. But $S^U - \{\wedge' cd\}$ is still upward saturated. So $S^U \subseteq S^U - \{\wedge' cd\}$. Then $\wedge' cd \notin S^U$. □

Definition Given $I \subseteq \Pi(D)$, we can introduce set S_I such that $S_I = \{p'\overline{d} : p\overline{d} \in I\} \cup \{\top'\}$. We define a function $V_I : D_o \to \mathcal{B}$ as follows: for each $d \in D_o$, $V_I d = \mathbf{T}$ if $d \in S_I^U$, $\mathbf{F}$ otherwise. □

Lemma 4.21 *S_I is downward saturated and S_I^U saturated. And V_I is a semivaluation of D.* □

Lemma 4.22 *†iivmon: $I_1 \subseteq I_2 \subseteq \Pi(D)$. Then $V_{I_1} \subseteq V_{I_2}$.*

Proof $S^U_{I_1} \subseteq S^U_{I_2}$. $\square$

Definition Let $I \subseteq \Pi(D)$. Then I induces the set $\mathcal{D}^I$ of V_I-complexes based on D and the following functions whose domain is D: the function $\boldsymbol{v}_I$ such that for each $d \in D$, $\langle d, \boldsymbol{v}_I(d)\rangle \in \mathcal{D}^I$, the function $\kappa_I : D \to \mathcal{D}^I$ such that for $d \in D$, $\kappa_I d = \langle d, \boldsymbol{v}_I(d)\rangle$, and the function $\boldsymbol{e}_I$ such that for $d \in D$, $\boldsymbol{e}_I d = (\kappa_I d)^{\odot}$. $\square$

Lemma 4.23 *$I_1 \subseteq I_2 \subseteq \Pi(D)$. Then $\boldsymbol{v}_{I_1} \subseteq \boldsymbol{v}_{I_2}$.*

Proof We prove $\boldsymbol{v}_{I_1} d_\gamma \subseteq \boldsymbol{v}_{I_2} d_\gamma$ for all $d_\gamma \in D$ by induction on γ. When $\gamma = o$. Follows by Lemma 4.22. When $\gamma \in \mathcal{T}_0 - \{o\}$. By definition of V_I-complexes,$\boldsymbol{v}_{I_1} d_\gamma = \boldsymbol{v}_{I_2} d_\gamma$.

$\gamma = \alpha \to \beta$. Let $a \in \mathcal{D}_\alpha$.

$$\begin{aligned}(\boldsymbol{v}_{I_1} d_\gamma)a &= \langle d_\gamma a^1, \boldsymbol{v}_{I_1}(d_\gamma a^1)\rangle \\ &\subseteq \langle d_\gamma a^1, \boldsymbol{v}_{I_2}(d_\gamma a^1)\rangle && \text{by IH} \\ &= (\boldsymbol{v}_{I_2} d_\gamma)a.\end{aligned}$$

$\square$

Lemma 4.24 *†eiapp: Let $I \subseteq \Pi(D)$ and $d \in D_{\alpha\to\beta}$. Then for all $d_1 \in D_\alpha$, $\boldsymbol{e}_I(d)d_1 = \boldsymbol{e}_I(dd_1)$.*

Proof $\boldsymbol{e}_I(d)d_1 = (\kappa_I d)^{\odot} d_1 = (\kappa_I d \star \kappa_I d_1)^{\odot} = (\kappa d d_1)^{\odot} = \boldsymbol{e}_I(dd_1)$. $\square$

Lemma 4.25 *$I_1 \subseteq I_2 \subseteq \Pi(D)$. Then $\boldsymbol{e}_{I_1} \subseteq \boldsymbol{e}_{I_2}$.*

Proof We prove for $d \in D_{\alpha_1,\cdots,\alpha_n\to\beta}, \beta \in \mathcal{T}_0, n \geq 0$, $\boldsymbol{e}_{I_1}(d) \subseteq \boldsymbol{e}_{I_2}(d)$ by induction on n.

$n = 0$. $\boldsymbol{e}_{I_1} d = \langle d, \boldsymbol{v}_{I_1} d\rangle^{\odot} = \boldsymbol{v}_{I_1} d \subseteq \boldsymbol{v}_{I_2} d = \langle d, \boldsymbol{v}_{I_2} d\rangle^{\odot} = \boldsymbol{e}_{I_2} d$.

$n > 0$. For $d_1 \in D_{\alpha_1}$,

$$\begin{aligned}(\boldsymbol{e}_{I_1} d)d_1 &= \boldsymbol{e}_{I_1}(dd_1) && \text{by Lemma 4.24} \\ &\subseteq \boldsymbol{e}_{I_2}(dd_1) && \text{by IH} \\ &= (\boldsymbol{e}_{I_2} d)d_1.\end{aligned}$$

$\square$

Definition Let $\langle D, J\rangle$ be a general pre-interpretation. An *interpretation* $\mathcal{M}$ is a pair $\langle D, I\rangle$ where $I : \Pi \to \Phi(D)$. We call $\mathcal{M}$ a D-interpretation. An *assignment into* $\mathcal{M}$ is a good mapping $\varphi : \mathcal{V} \to D$. $\square$

Definition Let $\mathcal{M} = \langle D, I\rangle$ be an interpretation of $\mathcal{L}$, φ an assignment into $\mathcal{M}$. When F is a formula in $\mathcal{L}$, we write $\mathcal{M} \models F[\varphi]$ to say that $\mathcal{M}$ satisfies F wrt φ. For all goal formulas G, G_1, G_2, for each rigid atom A,

(a) When $s_\alpha, t_\alpha \in T(\Sigma)_\alpha$, $\mathcal{M} \models s_\alpha = t_\alpha[\varphi]$ iff $\mathsf{V}_\varphi s_\alpha = \mathsf{V}_\varphi t_\alpha$,

$\mathcal{M} \models s_\alpha \equiv t_\alpha[\varphi]$ iff $e_I \mathsf{V}_\varphi s_\alpha = e_I \mathsf{V}_\varphi t_\alpha$.

(b) $\mathcal{M} \models \top[\varphi]$

(c) $\mathcal{M} \models p(t_1, \cdots, t_n)[\varphi]$ iff $\langle \mathsf{V}_\varphi t_1, \cdots, \mathsf{V}_\varphi t_n \rangle \in Ip$ if p is a constant

or $\langle \mathsf{V}_\varphi t_1, \cdots, \mathsf{V}_\varphi t_n \rangle \in \varphi \circ e_I(p)$ if p is a variable

(d) $\mathcal{M} \models G_1 \vee G_2[\varphi]$ iff $\mathcal{M} \models G_1[\varphi]$ or $\mathcal{M} \models G_2[\varphi]$

(e) $\mathcal{M} \models G_1 \wedge G_2[\varphi]$ iff $\mathcal{M} \models G_1[\varphi]$ and $\mathcal{M} \models G_2[\varphi]$

(f) $\mathcal{M} \models \exists x_\alpha G$ iff there is a $d \in D_\alpha$ such that $\mathcal{M} \models G[\varphi[d/x_\alpha]]$

(g) $\mathcal{M} \models A \leftarrow G[\varphi]$ iff $\mathcal{M} \models A[\varphi]$ if $\mathcal{M} \models G[\varphi]$.

We write $\mathcal{M} \models F$ to say that a formula F is *valid* in $\mathcal{M}$ if $\mathcal{M} \models F[\varphi]$ for all assignments φ into $\mathcal{M}$. Given a definite program $\mathcal{P}$, we say that $\mathcal{M}$ is a *model* or *D-model* for $\mathcal{P}$, and write $\mathcal{M} \models \mathcal{P}$, if each definite clause in $\mathcal{P}$ is valid in $\mathcal{M}$. Given a closed goal formula G, we say that G is a *logical consequence* of $\mathcal{P}$, and write $\mathcal{P} \models G$ if G is valid in all models of $\mathcal{P}$. □

Assume that G is any formula where the predicate variable p of type $\overline{\alpha} \to o$ does not occur free. We call any formula of the following form

$$\exists p \forall \overline{x}[p(\overline{x}) \leftrightarrow G]$$

as a *relation comprehension formula.*

Theorem 4.26 *Relation comprehension formulas are valid.*

Proof By induction on the goal formula G. □

Definition Let $\mathcal{F} = \langle D, J \rangle$ be a general pre-interpretation, V a semivaluation of D, and $\mathcal{D}$ be the set of V-complexes based on D. Then $\mathcal{D}^\odot$ is defined to be the set $\{a^\odot : a \in \mathcal{D}\}$. Let $\mathcal{A} = \langle \mathcal{D}, J \rangle$ be an $\mathcal{L}$-structure based on $\mathcal{F}$. Then the *D-interpretation* $\mathcal{A}^\odot$ *induced by* $\mathcal{A}$ is defined to be $\langle D, I \rangle$ where $I = J \circ \kappa \circ (\cdot)^\odot \uparrow \Pi$. □

Lemma 4.27 *Let $\mathcal{D}^\odot$ be as above. Then $\wp = \mathcal{D}^\odot$ is an elementarily closed extensional domain based on D.*

Proof Assume $P, Q \in \wp_{\overline{\alpha} \to o}$ to show $P \vee Q \in \wp_{\overline{\alpha} \to o}$. Then there are $a, b \in \mathcal{D}_{\overline{\alpha} \to o}$ such that $P = a^\odot$ and $Q = b^\odot$, and an assignment φ into $\mathcal{D}$ satisfying $\varphi y = a$ and $\varphi z = b$. Let $c = \mathcal{V}_\varphi(\lambda \overline{x} \cdot y\overline{x} \vee z\overline{x}) \in \mathcal{D}_{\overline{\alpha} \to o}$ Then it is easy to see that $c^\odot = P \vee Q$. □

Lemma 4.28 †*ind: Let φ be an assignment into D. Then for all formulas F in $\mathcal{L}$, $\mathcal{A}^\odot \models F[\varphi]$ iff $\mathcal{A} \models F[\varphi \circ \kappa]$.*

Proof F is $s_\alpha = t_\alpha$.

$\mathcal{A} \models F[\varphi \circ \kappa]$ iff $\mathcal{V}_{\varphi\circ\kappa} s_\alpha = \mathcal{V}_{\varphi\circ\kappa} t_\alpha$ iff $\kappa \mathsf{V}_\varphi s_\alpha = \kappa \mathsf{V}_\varphi t_\alpha$ iff $\mathsf{V}_\varphi s_\alpha = \mathsf{V}_\varphi t_\alpha$.

F is $s_\alpha \equiv t_\alpha$.

$\mathcal{A} \models F[\varphi \circ \kappa]$ iff $(\mathcal{V}_{\varphi\circ\kappa} s_\alpha)^\odot = (\mathcal{V}_{\varphi\circ\kappa} t_\alpha)^\odot$ iff $(\kappa \mathsf{V}_\varphi s_\alpha)^\odot = (\kappa \mathsf{V}_\varphi t_\alpha)^\odot$ iff $e_I \mathsf{V}_\varphi s_\alpha = e_I \mathsf{V}_\varphi t_\alpha$.

F is $p(t_1, \cdots, t_n)$.

If p is a constant, $\mathcal{A} \models p(t_1, \cdots, t_n)[\varphi \circ \kappa]$ iff $\langle \mathsf{V}_\varphi t_1, \cdots, \mathsf{V}_\varphi t_n \rangle \in J \circ \kappa \circ (\cdot)^\odot(p)$. If p is a variable, $\mathcal{A} \models p(t_1, \cdots, t_n)[\varphi \circ \kappa]$ iff $\langle \mathsf{V}_\varphi t_1, \cdots, \mathsf{V}_\varphi t_n \rangle \in \varphi \circ \kappa \circ (\cdot)^\odot(p)$.

$\mathcal{A} \models \exists x_\alpha G_1[\varphi \circ \kappa]$. There is an $a \in \mathcal{D}_\alpha$ such that $\mathcal{A} \models G_1[\varphi \circ \kappa[a/x_\alpha]]$. So by IH, $\mathcal{A}^\odot \models G_1[\varphi[a^1/x_\alpha] \circ \kappa]$.

□

Lemma 4.29 †*eimp: For all formula F in $\mathcal{L}$, $\mathcal{A}^\odot \models F$ iff $\mathcal{A} \models F$.*

Proof Assignments into $\mathcal{A}^\odot$ and $\mathcal{A}$ have one-one correspondence between them. So the lemma follows from Lemma 4.28. □

Theorem 4.30 †*e2a: Let $\mathcal{P}$ be a definite program and G a closed goal. Then $\mathcal{P} \models G$ implies $\mathcal{P} \models G$.*

Proof Let $\mathcal{A} \models \mathcal{P}$. Then by Lemma 4.29 $\mathcal{A}^\odot \models \mathcal{P}$, so $\mathcal{A}^\odot \models G$.

Therefore $\mathcal{A} \models G$ by Lemma 4.29. □

Definition Let $\mathcal{F} = \langle D, J \rangle$ be a general pre-interpretation, $\mathcal{M} = \langle D, I \rangle$ a D-interpretation based on $\mathcal{F}$, and $D^\oplus$ be a set of V_I-complexes based on D. Then $\mathcal{M}^\oplus$ is an *$\mathcal{L}$-structure $\langle D^\oplus, J \rangle$ induced by $\mathcal{M}$.* □

Lemma 4.31 *For all formula F in $\mathcal{L}$ and assignment φ into D,*
$\mathcal{M}^\oplus \models F[\varphi \circ \kappa_I]$ iff $\mathcal{M} \models F[\varphi]$. □

Lemma 4.32 †*d2aval: For all formula F in $\mathcal{L}$, $\mathcal{M}^\oplus \models F$ iff $\mathcal{M} \models F$.* □

Theorem 4.33 †*a2e: Let $\mathcal{P}$ be a definite program and G a closed goal. Then*
$\mathcal{P} \models G$ implies $\mathcal{P} \models G$.

Proof Assume $\mathcal{M} \models \mathcal{P}$. Then $\mathcal{M}^\oplus \models \mathcal{P}$. So $\mathcal{M}^\oplus \models G$. By Lemma 4.32, $\mathcal{M} \models G$. □

Theorem 4.34 *For all formula F in $\mathcal{L}$, $\models F$ iff $\models F$.* □

Theorem 4.35 *Let $\mathcal{P}$ be a definite program and G a closed goal. Then $\mathcal{P} \models G$ iff $\mathcal{P} \models G$.*

Proof $\Rightarrow$) By Theorem 4.30. $\Leftarrow$) By Theorem 4.33. □

Theorem 4.36 *The extensionality is not valid.*

Proof Take an extensionality formula $p_o \equiv q_o \to p_o = q_o$. It is obvious that $\mathcal{V}^2_\varphi p_o = \mathcal{V}^2_\varphi q_o$ does not imply that $\mathcal{V}_\varphi p_o = \mathcal{V}_\varphi q_o$.

For the extensionality formula $(\forall x_\alpha \cdot fx \equiv gx) \to f = g$, we take $\alpha \in \mathcal{T}_0$ and $\beta = o$ and D-interpretation I such that $Ip = Iq = \mathbf{F}^D_\alpha$. Then $p \equiv q$ but not always $p = q$. □

Let $\mathcal{M} = \langle D, I\rangle$ be an interpretation based on $\mathcal{F} = \langle D, J\rangle$, we can identify $\mathcal{M}$ with the subset I of $\Pi(D)$. And every subset I of $\Pi(D)$ is a D-interpretation. Obviously the set of all D-interpretation is a complete lattice with the usual set inclusion ordering between D-interpretations.

Theorem 4.37 *†iimon: Let $I_1 \subseteq I_2 \subseteq \Pi(D)$. Then*
If $I_1 \models G[\varphi]$, then $I_2 \models G[\varphi]$.
Proof By induction on G.

When G is $\top$, it is obvious. When G is a rigid atom $p(t_1, \cdots, t_n)$, since $I_1 p \subseteq I_2 p$, $I_2 \models G[\varphi]$. When G is $p(t_1, \cdots, t_n)$ where p is a variable. Since $e_{I_1} \subseteq e_{I_2}$, $I_2 \models p(t_1, \cdots, t_n)[\varphi]$.

When G is $G_1 \wedge G_2$. $I_1 \models G_1[\varphi]$ and $I_1 \models G_2[\varphi]$. By IH $I_2 \models G_1[\varphi]$ and $I_2 \models G_2[\varphi]$. So $I_2 \models G[\varphi]$.

When G is $G_1 \vee G_2$. Assume, wlog, $I_1 \models G_1[\varphi]$. By IH $I_2 \models G_1[\varphi]$.

When G is $\exists x_\alpha G_1$. There exists a $d \in D_\alpha$ such that $I_1 \models G_1[\varphi[d/x_\alpha]]$. By IH $I_2 \models G_1[\varphi[d/x_\alpha]]$. So $I_2 \models G[\varphi]$. □

Let $\mathcal{F} = \langle D, J\rangle$ be a general pre-interpretation. We can define a mapping $\mathsf{T}_{\mathcal{P}}^{\mathcal{F}}$ from the lattice of D-interpretations to itself.

Definition Let $\mathcal{F}$ be a pre-interpretation $\langle D, J\rangle$ of a definite program $\mathcal{P}$ and I a D-interpretation. Then

$$\mathsf{T}_{\mathcal{P}}^{\mathcal{F}}(I) = \{p(d_1, \cdots, d_n) \in \Pi(D) : \text{there exist an assignment } \varphi \text{ into } I \text{ and}$$
$$\text{a clause } p(t_1, \cdots, t_n) \leftarrow G \in \mathcal{P} \text{ such that}$$
$$d_i = \mathsf{V}_\varphi t_i \text{ for each } i \in [n] \text{ and}$$
$$I \models G[\varphi]\}$$

□

Lemma 4.38 *†tmonf: $\mathsf{T}_{\mathcal{P}}^{\mathcal{F}}$ is monotonic, i.e. given $I_1 \subseteq I_2 \subseteq \Pi(D)$, $\mathsf{T}_{\mathcal{P}}^{\mathcal{F}}(I_1) \subseteq \mathsf{T}_{\mathcal{P}}^{\mathcal{F}}(I_2)$.*
Proof Assume $p(d_1, \cdots, d_n) \in \mathsf{T}_{\mathcal{P}}^{\mathcal{F}}(I_1)$ for $p(d_1, \cdots, d_n) \in \Pi(D)$. Then there are an assignment φ into I_1 and a clause $p(t_1, \cdots, t_n) \leftarrow G \in \mathcal{P}$ such that $\mathsf{V}_\varphi t_i = d_i$ for all $i \in [n]$ and $I_1 \models G[\varphi]$. By Theorem 4.37, $I_2 \models G[\varphi]$. □

So $\mathsf{T}_{\mathcal{P}}^{\mathcal{F}}$ is a transformation on the set of all D-interpretations.

Lemma 4.39 *†dmodel: Let $I \subseteq \Pi(D)$. Then $I \models \mathcal{P}$ iff $\mathsf{T}_{\mathcal{P}}^{\mathcal{F}}(I) \subseteq I$.*
Proof $\Rightarrow$) Assume $p(d_1, \cdots, d_n) \in \mathsf{T}_{\mathcal{P}}(I)$ for some $p(d_1, \cdots, d_n) \in \Pi(D)$. Then there are an assignment φ into I and a clause $p(t_1, \cdots, t_n) \leftarrow G \in \mathcal{P}$ such that $\mathsf{V}_\varphi t_i = d_i$ for all $i \in [n]$ and $I \models G[\varphi]$. Then since $I \models \mathcal{P}$, $I \models p(t_1, \cdots, t_n)[\varphi]$. Therefore $p(d_1, \cdots, d_n) \in I$.
$\Leftarrow$) Similarly. □

Lemma 4.40 †*dmints: Let I_1 and I_2 be D-models of $\mathcal{P}$. Then $I_1 \cap I_2$ is also D-model of $\mathcal{P}$.*

Proof Since $\mathsf{T}_{\mathcal{P}}(I_1) \subseteq I_1$ and $\mathsf{T}_{\mathcal{P}}(I_2) \subseteq I_2$, by monotonicity of $\mathsf{T}_{\mathcal{P}}$ operator,
$\mathsf{T}_{\mathcal{P}}(I_1 \cap I_2) \subseteq \mathsf{T}_{\mathcal{P}}(I_1) \subseteq I_1$ and
$\mathsf{T}_{\mathcal{P}}(I_1 \cap I_2) \subseteq \mathsf{T}_{\mathcal{P}}(I_2) \subseteq I_2$.
So $\mathsf{T}_{\mathcal{P}}(I_1 \cap I_2) \subseteq I_1 \cap I_2$. □

But the set of all D-models is not closed under join operation, i.e. $I_1 \cup I_2$ is not necessarily a D-model, whenever I_1 and I_2 are D-models. Take for example the definite program $\mathcal{P}_2 = \{p \leftarrow q, r\}$. Then $\Pi(D) = \{p, q, r\}$. $\{q\}$ and $\{r\}$ are D-models for $\mathcal{P}_2$, but $\{q, r\}$ is not a D-model.

Still we can show that every definite program has the least D-model as follows:

Theorem 4.41 †*4min: Let $\mathsf{M}_{\mathcal{P}} = \bigcap\{I \subseteq \Pi(D) : I \models P\}$, then $\mathsf{M}_{\mathcal{P}}$ is the least D-model of $\mathcal{P}$ and $\mathsf{M}_{\mathcal{P}} = \mathsf{T}^{\gamma}_{\mathcal{P}}(\phi)$ for the closure ordinal γ.*

Proof By Lemmas 4.39,4.40,4.38, and Theorem 3.3. □

5 Herbrand Models

Definition The *Herbrand domain H* is a set such that

(a) H is the set of all ρ-normal c-terms.

(b) Let $f \in H_{\alpha\to\beta}$, then for all $t \in H_\alpha$, $f(t) = \rho(ft)$. □

It is obvious that the Herbrand domain H is countable.

Definition The *Herbrand pre-interpretation $\mathcal{HF}$* is a pre-interpretation $\langle H, J\rangle$ such that H is the Herbrand domain and J satisfies the following:

(a) If c_α is a constant such that α is a primitive type, then $Jc_\alpha = c_\alpha$.

(b) If $d_{\alpha\to\beta}$ is a constant of type $\alpha \to \beta$, then for all $t_\alpha \in H_\alpha$, $(Jd_{\alpha\to\beta})(t_\alpha) = d_{\alpha\to\beta}t_\alpha$. □

Lemma 5.1 *The Herbrand pre-interpretation is general.* □

Definition An *Herbrand interpretation $\mathcal{M}$* is an interpretation $\langle H, I\rangle$ based on the Herbrand pre-interpretation. The *Herbrand base $\mathcal{HB}$* is the set $\Pi(H)$. □

As for D-interpretations, we can identify an Herbrand interpretation $\mathcal{M}$ with a subset I of the Herbrand base $\Pi(H)$.

Let $I \subseteq \Pi(H)$ be an Herbrand interpretation and φ an assignment into I. Then we can consider φ as the generalized substitution σ such that for each term $t \in T(\Sigma)$, $\sigma t = (\varphi \uparrow FV(t))t$. It is easy to see that for every term t, φt is a c-term and $\mathsf{V}_\varphi t = \varphi t$, for each goal formula G, φG a closed goal formula, and for each definite clause C, φC a closed definite clause.

The *Herbrand interpretation* $\mathcal{M}^*$ *induced by* $\mathcal{M}$ is an Herbrand interpretation I^* such that for every $A \in \Pi(H)$, $A \in I^*$ iff $\mathcal{M} \models A$. Let φ and φ' be assignments into I and $\mathcal{M}$, respectively. Then we say that φ' is *induced by* φ if $\varphi' = \varphi \circ \mathsf{V}^{\mathcal{F}}$. The mapping $\mathsf{V}^{\mathcal{F}} : H \to D$ is a *homomorhism from* I^* *into* I, since for $p \in \Pi_{\alpha_1,\cdots,\alpha_n\to o}, h_i \in H_{\alpha_i}, i \in [n]$, if $\langle h_1, \cdots, h_n\rangle \in I^*p$, then $\langle \mathsf{V}^{\mathcal{F}}h_1, \cdots, \mathsf{V}^{\mathcal{F}}h_n\rangle \in Ip$. Moreover,

Lemma 5.2 *†hdhom: Let* $h \in H_{\alpha_1,\cdots,\alpha_n\to o}$. *Then for all* $h_i \in H_{\alpha_i}, i \in [n]$, $\langle h_1, \cdots, h_n\rangle \in e_{I^*}(h)$ *implies* $\langle \mathsf{V}^{\mathcal{F}}h_1, \cdots, \mathsf{V}^{\mathcal{F}}h_n\rangle \in e_I(\mathsf{V}^{F}h)$.

Proof By induction on n. □

Lemma 5.3 *†d2h: Let* $\mathcal{M}, I^*, \varphi', \varphi$ *be as above. Then*

(a) If t *is a term, then* $\mathsf{V}^{\mathcal{F}}_{\varphi'}(\varphi t) = \mathsf{V}^{\mathcal{F}}_{\varphi'}t$,

(b) If A *is a rigid atom then* $I^* \models A[\varphi]$ *iff* $\mathcal{M} \models A[\varphi']$,

(c) If G *is a goal formula such that* $I^* \models G[\varphi]$, *then* $\mathcal{M} \models G[\varphi']$, *and*

(d) If C *is a definite clause such that* $\mathcal{M} \models C[\varphi']$, *then* $I^* \models C[\varphi]$.

Proof (a) Let all free variables of t are in $x_1, \cdots, x_k$.

$$\begin{aligned}
\mathsf{V}^{\mathcal{F}}_{\varphi'}(\varphi t) &= \mathsf{V}^{\mathcal{F}}_{\varphi'}([\lambda x_1 \cdots \lambda x_k t](\varphi x_1)\cdots(\varphi x_k)) && \text{by the definition of substitution}\\
&= (\mathsf{V}^{\mathcal{F}}_{\varphi'}[\lambda x_1 \cdots \lambda x_k t])\mathsf{V}^{\mathcal{F}}_{\varphi'}(\varphi x_1)\cdots \mathsf{V}^{\mathcal{F}}_{\varphi'}(\varphi x_k) && \text{by definition of V}\\
&= (\mathsf{V}^{\mathcal{F}}_{\varphi'}[\lambda x_1 \cdots \lambda x_k t])(\varphi' x_1)\cdots(\varphi' x_k) && \text{by definition of } \varphi'\\
&= \mathsf{V}^{\mathcal{F}}_{\varphi'}t
\end{aligned}$$

(b) Let $A = p(t_1, \cdots, t_n)$ where p is a predicate constant.

$I^* \models A[\varphi]$ iff $\langle \varphi t_1, \cdots, \varphi t_n\rangle \in I^*p$ iff $p(\varphi t_1, \cdots, \varphi t_n) \in I^*$ iff
$\mathcal{M} \models p(\varphi t_1, \cdots, \varphi t_n)[\varphi']$ iff $\quad$ $p(\cdots)$ is a closed goal.
$\langle \mathsf{V}^{\mathcal{F}}_{\varphi'}\varphi t_1, \cdots, \mathsf{V}^{\mathcal{F}}_{\varphi'}\varphi t_n\rangle \in Ip$ iff
$\langle \mathsf{V}^{\mathcal{F}}_{\varphi'}t_1, \cdots, \mathsf{V}^{\mathcal{F}}_{\varphi'}t_n\rangle \in Ip$ iff $\quad$ By (a)
$\mathcal{M} \models A[\varphi']$.

(c) By induction on the construction of G.

G is $\top$, trivial. G is an atomic formula $A = p(t_1, \cdots, t_n)$. When p is a constant, then by (b), $I^* \models A[\varphi]$ iff $\mathcal{M} \models A[\varphi']$. When p is a variable, similarly, by Lemma 5.2 $\langle \varphi t_1, \cdots, \varphi t_n\rangle \in e_{I^*}(\varphi p)$ implies $\langle \mathsf{V}^{\mathcal{F}}_{\varphi'}t_1, \cdots, \mathsf{V}^{\mathcal{F}}_{\varphi'}t_n\rangle \in e_I(\mathsf{V}^{\mathcal{F}}_{\varphi'}\varphi p)$.

G is $\exists x_\alpha G_1$. Then there is an $h \in H_\alpha$ such that $I^* \models G_1[\varphi[h/x_\alpha]]$. So by IH $\mathcal{M} \models G_1[\varphi'[\mathsf{V}^{\mathcal{F}}_{\varphi'}h/x_\alpha]]$. Therefore $\mathcal{M} \models \exists x_\alpha G_1[\varphi']$.

(d) Let $C = A \leftarrow G$. Assume $I^* \models G[\varphi]$ to show $I^* \models A[\varphi]$. $\mathcal{M} \models G[\varphi']$ by (c). $\mathcal{M} \models A[\varphi']$ since $\mathcal{M} \models C[\varphi']$. $I^* \models A[\varphi]$ by (b). □

Theorem 5.4 *Let I be the Herbrand interpretation induced by an interpretation $\mathcal{M}$ and $\mathcal{P}$ a definite program. Then if $\mathcal{M} \models \mathcal{P}$, then $I \models \mathcal{P}$.*

Proof By Lemma 5.3 (d). $\square$

Let $\mathcal{F}$ be a general pre-interpretation. Then $\models_{\mathcal{F}}$ denotes logical implication in the context of fixed domains and functional assignment. Specifically $\models_{\mathcal{HF}}$ denotes logical implication in the context of Herbrand domain and functional assignment.

Let G be a goal formula. We write $\exists(G)$ to denote the existential closure of free variables in G.

Theorem 5.5 *Let $\mathcal{P}$ be a definite program and G a goal formula.*
Then $\mathcal{P} \models \exists(G)$ iff $\mathcal{P} \models_{\mathcal{HF}} \exists(G)$.

Proof $\Longleftarrow$) Let an Herbrand interpretation induced by the given interpretation $\mathcal{M}$ be I. Assume $\mathcal{M} \models \mathcal{P}$. Then $I \models \mathcal{P}$, so $I \models \exists(G)$. Then there is an assignment φ into I such that $I \models G[\varphi]$. Let the assignment φ' into $\mathcal{M}$ be induced by φ. Then $\mathcal{M} \models G[\varphi']$ by Lemma 5.3. So $\mathcal{M} \models \exists(G)$. $\square$

If φ is a substitution, then φ_{-x_α} is that substitution σ such that $\sigma = \varphi \uparrow (\mathcal{V} - \{x_\alpha\})$.

Lemma 5.6 *Let $I \subseteq \Pi(H)$. Then for all closed substitution σ, assignment φ into H, and goal formula G, $I \models \sigma G[\varphi]$ iff $I \models \varphi\sigma G$.*

Proof We prove by induction on G. When G is $\top$ or a rigid atom, it is obvious. When G is $p(t_1, \cdots, t_n)$ where $p \in \mathcal{V}$. $I \models \sigma G[\varphi]$ iff $\langle \varphi\sigma t_1, \cdots, \varphi\sigma t_n\rangle \in e_I(\varphi\sigma p)$ iff $\langle \varphi'\varphi\sigma t_1, \cdots, \varphi'\varphi\sigma t_n\rangle \in e_I(\varphi'[\varphi\sigma p/p]p)$ for all assignment φ' into H iff $I \models \varphi\sigma G[\varphi']$ for all assignment φ' into H iff $I \models \varphi\sigma G$.

When G is $\exists x_\alpha G_1$. $I \models \sigma G[\varphi]$ iff $I \models \exists x_\alpha \sigma_{-x_\alpha} G_1[\varphi]$ iff there is an $h \in H_\alpha$ such that $I \models \sigma_{-x_\alpha} G_1[\varphi[h/x_\alpha]]$ iff there is an $h \in H_\alpha$ such that $I \models \varphi[h/x_\alpha]\sigma_{-x_\alpha} G_1$ by IH iff for all assignment φ' into H, $I \models \varphi'\varphi[h/x_\alpha]\sigma_{-x_\alpha} G_1$, since $\varphi[h/x_\alpha]\sigma_{-x_\alpha} G_1$ is a closed goal. iff $I \models (\varphi'[h/x_\alpha])\varphi_{-x_\alpha}\sigma_{-x_\alpha} G_1$ iff $I \models \varphi_{-x_\alpha}\sigma_{-x_\alpha} G_1[\varphi'[h/x_\alpha]]$ iff $I \models \exists x_\alpha \varphi_{-x_\alpha}\sigma_{-x_\alpha} G_1[\varphi']$ iff $I \models \varphi\sigma G$. $\square$

Corollary 5.7 *For all assignment φ into H, goal formula G, $I \models G[\varphi]$ iff $I \models \varphi G$.* $\square$

Corollary 5.8 *For all assignment φ into H, goal formula G, $I \models \varphi\exists x_\alpha G$ iff there is an $h \in H_\alpha$ such that $I \models \varphi[h/x_\alpha]G$.*

Proof $I \models \varphi\exists x_\alpha G$ iff $I \models \exists x_\alpha G[\varphi]$ iff there is an $h \in H_\alpha$ such that $I \models G[\varphi[h/x_\alpha]]$ iff there is an $h \in H_\alpha$ such that $I \models \varphi[h/x_\alpha]G$. $\square$

Lemma 5.9 $\dagger$*bsisub: Let $\langle I_n\rangle_{n\in\omega}$ be ω-chain of Herbrand interpretations. Then for all assignment φ into H, goal formula G, $\bigsqcup_{n\in\omega} I_n \models \varphi G$ only if there is an $n \in \omega$ such that $I_n \models \varphi G$.*

Proof We prove the lemma by induction on G. (a) When G is $\top$ or a rigid atom, it is obvious.

When G is a flexible atom. If φG is an atom, by (a). If $\varphi G = G_1 \wedge G_2$. Then $\bigsqcup_{n\in\omega} I_n \models G_1, G_2$. So by IH, there are $l, m \in \omega$ such that $I_l \models G_1$ and $I_m \models G_2$. Assume, wlog, $l < m$. Then by Theorem 4.37, $I_m \models G_1$. Therefore $I_m \models G_1 \wedge G_2$. If $\varphi G = \exists x_\alpha G_1$, then there is $h \in H_\alpha$ such that $\bigsqcup_{n\in\omega} I_n \models [h/x_\alpha]G_1$ by Corollary. Therefore by IH, there is $n \in \omega$ such that $I_n \models [h/x_\alpha]G_1$. So $I_n \models \exists x_\alpha G_1$.

When G is $\exists G_1$. There is an $h \in H_\alpha$ such that $\bigsqcup_{n\in\omega} I_n \models \varphi[h/x_\alpha]G_1$. By IH there is an $n \in \omega$ such that $I_n \models \varphi[h/x_\alpha]G_1$. So $I_n \models \varphi\exists x_\alpha G_1$. □

Lemma 5.10 *Let $\langle I_n\rangle_{n\in\omega}$ be ω-chain of Herbrand interpretations. Then for each goal G and assignment φ into H, $\bigsqcup_{n\in\omega} I_n \models G[\varphi]$ only if there is an $n \in \omega$ such that $I_n \models G[\varphi]$.*
Proof By Lemma 5.9. □

Corollary 5.11 *Let $\mathsf{M}_\mathcal{P} = \bigcap\{I \subseteq \Pi(H) : I \models \mathcal{P}\}$. Then $\mathsf{M}_\mathcal{P} \models \mathcal{P}$.*

Proof Follows from Theorem 4.41. □

Let e be an expression with free variables in the list $\overline{x}$. By $|e|$, we mean *ρ-normal instances* of e, $|e| = \{\sigma e : \sigma$ is a closed ρ-normal substitution for $\overline{x}\}$.

If $C = A \leftarrow G$ is a definite clause, then $|C|$ denotes instances of C. This notation can be extended to set of definite clauses.

$$|\mathcal{P}| = \bigcup\{|C| : C \in \mathcal{P}\}$$

We note that, given above definition, $|C|$ is a collection of definite sentences, so also is $|\mathcal{P}|$.

We can now give another equivalent definition of the transformation $\mathsf{T}_\mathcal{P}^{\mathcal{HF}}$ on the set of all H-interpretations. Let $I \subseteq \Pi(H)$. We define $\mathsf{T}_\mathcal{P}^{\mathcal{HF}}(I) = \{A \in \Pi(H) :$ there is a $A \leftarrow G \in |\mathcal{P}|$ such that $I \models G\}$.

Corollary 5.12 $\mathsf{T}_\mathcal{P}^{\mathcal{HF}}$ *is monotone.* □

Lemma 5.13 *†5cont: $\mathsf{T}_\mathcal{P}^{\mathcal{HF}}$ is continuous.*

Proof Let $\langle I_n\rangle_{n\in\omega}$ be a ω-chain of Herbrand interpretations. We need to show:
$\mathsf{T}_\mathcal{P}(\bigsqcup_{n\in\omega} I_n) = \bigsqcup_{n\in\omega} \mathsf{T}_\mathcal{P}(I_n)$.
The monotonicity of $\mathsf{T}_\mathcal{P}$ implies that $\bigsqcup_{n\in\omega} \mathsf{T}_\mathcal{P}(I_n) \subseteq \mathsf{T}_\mathcal{P}(\bigsqcup_{n\in\omega} I_n)$.
Now we need to show that $\mathsf{T}_\mathcal{P}(\bigsqcup_{n\in\omega} I_n) \subseteq \bigsqcup_{n\in\omega} \mathsf{T}_\mathcal{P}(I_n)$.
Let $h_1, \cdots, h_n \in H$, and $p(h_1, \cdots, h_n) \in \Pi(H)$. Assume $p(h_1, \cdots, h_n) \in \mathsf{T}_\mathcal{P}(\bigsqcup_{n\in\omega} I_n)$, to show $p(h_1, \cdots, h_n) \in \bigsqcup_{n\in\omega} \mathsf{T}_\mathcal{P}(I_n)$.
There are $p(t_1, \cdots, t_n) \leftarrow G \in \mathcal{P}$ and an assignment φ into H such that $\mathsf{V}_\varphi t_i = h_i$ for all $i \in [n]$ and $\bigsqcup_{n\in\omega} I_n \models G[\varphi]$.
So there is $n \in \omega$ such that $I_n \models G[\varphi]$. Therefore there is $n \in \omega$ such that $p(h_1, \cdots, h_n) \in \mathsf{T}_\mathcal{P}(I_n)$. □

Theorem 5.14 $(\mathsf{T}_{\mathcal{P}}^{\mathcal{HF}})^{\omega}(\phi)$ *is the least fixed point of* $\mathsf{T}_{\mathcal{P}}^{\mathcal{HF}}$ *and* $\mathsf{M}_{\mathcal{P}} = (\mathsf{T}_{\mathcal{P}}^{\mathcal{HF}})^{\omega}(\phi)$.

Proof Follows from Lemma 5.13 and Theorem 3.4. □

Theorem 5.15 *Let* $A \in \Pi(H)$. *Then*
$\mathcal{P} \models A$ *iff* $\mathsf{M}_{\mathcal{P}} \models A$.

Proof $\mathcal{P} \models A$ iff $\mathcal{P} \models_{\mathcal{HF}} A$ iff
for all H-interpretation I such that $I \models \mathcal{P}$, $A \in I$ iff $A \in \mathsf{M}_{\mathcal{P}}$. □

For the definite program $\mathcal{P}_1$ introduced in section 1, it is easy to see that $\mathsf{T}_{\mathcal{P}_1}^{\omega}(\phi) = \{p(a), q(a), r(p(a))\}$. So $r(p(a))$ is a logical consequence of $\mathcal{P}_1$, while $r(q(a))$ is not.

6 λProlog with Equality and $H/\mathcal{E}$-Interpretations

Much of the research in logic programming concentrates on extensions of Prolog. An important issue is the integration of the essential concepts of functional and logic programming. Another issue is the use of equations to define data types. Works along these lines can be found in [9, 15].

In this section we will develop semantics for λProlog augmented with an equality theory $\mathcal{E}$. We will establish the existence of the least model and least fixed point semantics.

Let D be a universe and R an equivalent relation on D consistent with the intended equality theory $\mathcal{E}$. For each α, we write R_{α} for the restriction of R to D_{α}. Then

$$\mathsf{R} = \bigcup_{\alpha \in \mathcal{T}} \mathsf{R}_{\alpha}$$

Let d be an element of D_{α}. Then $[d]_{\mathsf{R}}$ is the equivalent class containing d. We also say that R is a *congruence relation* on D consistent with the intended equality theory $\mathcal{E}$ if R is an equivalent relation on D consistent with $\mathcal{E}$ and for all $\alpha, \beta \in \mathcal{T}$, for all $d \in D_{\alpha \to \beta}$, for all $c \in D_{\alpha}$, $[d]_{\mathsf{R}}[c]_{\mathsf{R}} = [dc]_{\mathsf{R}}$. We sometimes write $[d]_{\mathsf{R}}$ as $[d]$ when the congruence relation is clear from the context.

Given a universe D and a congruence relation R on D, we define a quotient universe D/R as a universe $\{D_{\alpha}/\mathsf{R}_{\alpha}\}_{\alpha}$.

If $\mathcal{F} = \langle D, J \rangle$ is a pre-interpretation, $\mathcal{F}/\mathsf{R}$ is defined to be a pre-interpretation
$\langle D/\mathsf{R}, J' \rangle$ such that for each constant c, $J'c = [Jc]_{\mathsf{R}}$.

Lemma 6.1 *If a pre-interpretation* $\mathcal{F} = \langle D, J \rangle$ *is general, then* $\mathcal{F}/\mathsf{R}$ *is also general.*
Proof Let V be a valuation function in $\mathcal{F}$, and $\mathcal{F}/\mathsf{R} = \langle D/\mathsf{R}, J' \rangle$ and φ an assignment into $\mathcal{F}/\mathsf{R}$. Then there is an assignment φ' into $\mathcal{F}$ such that $\varphi = \varphi' \circ [\cdot]_{\mathsf{R}}$. Define a binary function V' such that for each term t, $\mathsf{V}'_{\varphi}t = [\mathsf{V}_{\varphi'}t]_{\mathsf{R}}$. We show V' is a valuation function in $\mathcal{F}/\mathsf{R}$ by showing that for each term t_{α},

$\mathsf{V}'_\varphi t_\alpha \in D_\alpha/\mathsf{R}_\alpha$ by induction on t_α.

When t_α is a variable x_α,

$$\mathsf{V}'_\varphi x_\alpha = [\mathsf{V}_{\varphi'} x_\alpha] = [\varphi' x_\alpha] = \varphi x_\alpha$$

When t_α is a constant c_α,

$$\mathsf{V}'_\varphi c_\alpha = [\mathsf{V}_{\varphi'} c_\alpha] = [J c_\alpha] = J' c_\alpha$$

When t_α is $f_{\beta\to\alpha} s_\beta$,

$$\begin{aligned}
\mathsf{V}'_\varphi(f_{\beta\to\alpha} s_\beta) &= [\mathsf{V}_{\varphi'}(f_{\beta\to\alpha} s_\beta)] = [(\mathsf{V}_{\varphi'} f_{\beta\to\alpha})(\mathsf{V}_{\varphi'} s_\beta)] \\
&= [\mathsf{V}_{\varphi'} f_{\beta\to\alpha}][\mathsf{V}_{\varphi'} s_\beta] && \text{by definition of R} \\
&= (\mathsf{V}'_\varphi f_{\beta\to\alpha})(\mathsf{V}'_\varphi s_\beta) && \text{by IH}
\end{aligned}$$

When t_α is $\lambda x_\beta s_\gamma$, let

$$d' = \mathsf{V}'_\varphi(\lambda x_\beta s_\gamma) = [\mathsf{V}_{\varphi'} \lambda x_\beta s_\gamma] = [\boldsymbol{\lambda} b \in D_\beta \cdot \mathsf{V}_{\varphi'[b/x_\beta]} s_\gamma]$$

For $b \in D_\beta$,

$$\begin{aligned}
d'[b] &= [\mathsf{V}_{\varphi'} \lambda x_\beta s_\gamma][b] \\
&= [(\mathsf{V}_{\varphi'} \lambda x_\beta s_\gamma) b] && \text{by definition of R} \\
&= [\mathsf{V}_{\varphi'[b/x_\beta]} s_\gamma] \\
&= \mathsf{V}'_{\varphi[[b]/x_\beta]} s_\gamma && \text{by IH}
\end{aligned}$$

□

Corollary 6.2 *Let $\mathcal{HF}$ be the Herbrand pre-interpretation, then $\mathcal{HF}/\mathsf{R}$ is general.* □

By analogy with the first-order case in [14, 17], we have

Proposition 6.3 *Let A be a closed atom. Then*
$\mathcal{P}, \mathcal{E} \models A$ iff $\mathcal{P}, \mathcal{E} \models_{\mathcal{HF}/\mathsf{R}} A$ for all R. □

We want the existence of a canonical model for the equality theory, i.e. we wish the existence of a congruence R_0 such that

Proposition 6.4 *$\mathcal{E} \models s_\alpha = t_\alpha$ iff $[s_\alpha]_{\mathsf{R}_0} = [t_\alpha]_{\mathsf{R}_0}$ where s_α and t_α are closed terms.* □

But this can be achieved only if the theory $\mathcal{E}$ has a finest congruence relation R_0. This motivates our choice of using *Horn equality clauses* in our framework presented below.

A *definite clause logic program* $\mathcal{P}$ is defined to be a finite set of definite clauses

$$A \leftarrow e_1, \cdots, e_n, G$$

where A is a rigid atom in $\mathcal{G}$, i.e. not an equation, each e_i is an equation and G is a goal formula in $\mathcal{G}$.

A *Horn equality clause* takes one of two forms

$$e \leftarrow e_1, \cdots, e_n$$

or

$$\leftarrow e_1, \cdots, e_n$$

where $n \geq 0$ and all the e_i's therein are equations. As usual, variables in Horn equality clauses are implicitly universally quantified. We define a *Horn clause equality theory* to be a set of equality clauses. A given consistent Horn clause equality theory $\mathcal{E}$ defines a logic programming language whose programs, called *logic programs*, are the pairs $(\mathcal{P}, \mathcal{E})$ where $\mathcal{P}$ is a definite clause logic program.

Lemma 6.5 *There exists a finest congruence over H generated by each definite equality theory $\mathcal{E}$.*

Proof Consider models of $\mathcal{E}$ over the Herbrand domain H, and for our purposes here, a model is a set of pairs in $H \times H$. Suppose now that I is the intersection of a set of models of $\mathcal{E}$. If I is not a model itself, then some closed instance of a clause in $\mathcal{E}$, say $e \leftarrow e_1, \cdots, e_n$, is falsified by I. This means that e is not in I while $e_1, \cdots, e_n$ are in I, contradicting the fact that e is in the models of the set in question. The finest congruence then is given by the intersection of all models of $\mathcal{E}$, with the obvious functional assignment J' such that $(J' f_{\alpha \rightarrow \beta})([t_\alpha]) = [f_{\alpha \rightarrow \beta} t_\alpha]$ for each functional constant $f_{\alpha \rightarrow \beta} \in \Sigma$. □

We thus may now write $H/\mathcal{E}$ to denote this finest congruence. As a consequence

Lemma 6.6 *Let A be a closed atom. Then*
$(\mathcal{P}, \mathcal{E}) \models A$ *iff* $(\mathcal{P}, \mathcal{E}) \models_{H/\mathcal{E}} A$

Proof It suffices to prove that $(\mathcal{P}, \mathcal{E}) \models_{H/\mathcal{E}} A$ iff $(\mathcal{P}, \mathcal{E}) \models_{H/\mathsf{R}} A$ for all R.
$\Longrightarrow$) Let R_0 be the finest congruence relation. For some R, let I be any H/R-interpretation such that $I \models (\mathcal{P}, \mathcal{E})$, but $I \not\models A$. Construct the following $H/\mathcal{E}$-model I' by defining that $I' \models p([t_1]_{\mathsf{R}_0}, \cdots, [t_n]_{\mathsf{R}_0})$ iff $I \models p([t_1]_{\mathsf{R}}, \cdots, [t_n]_{\mathsf{R}})$ for all predicate constant p. This is well defined because R_0 is finer than R. □

Proposition 6.7 *Let A be a closed atom. Then*
$(\mathcal{P}, \mathcal{E}) \models A$ *iff* $\mathcal{P} \models_{H/\mathcal{E}} A$. □

Let $\bar{t}$ denote a sequence of terms $t_1, \cdots, t_n$, $n \geq 0$. We now give definitions wrt a given logic program $(\mathcal{P}, \mathcal{E})$.

The $\mathcal{E}$-base is the set $\Pi(H/\mathcal{E})$. We write $[s]$ to denote the element of $H_\alpha/\mathcal{E}$ assigned to the ground term$_\alpha$ s. Similarly, $[\bar{t}]$ is an element of $H_{\alpha_1}/\mathcal{E} \times \cdots \times H_{\alpha_n}/\mathcal{E}$ and $p([\bar{t}])$ is an element of the $\mathcal{E}$-base. Where S is a set of closed terms, $[S]$ denotes $\{[s] : s \in S\}$.

We are now in a position to give the declarative semantics of higher-order logic program with equality as a natural extension of the declarative semantics of the traditional first-order logic programs.

Theorem 6.8 *There is a least $H/\mathcal{E}$-model $\mathsf{M}_{(\mathcal{P},\mathcal{E})}$ of $(\mathcal{P},\mathcal{E})$, and for all $A \in \Pi(H)$, $(\mathcal{P},\mathcal{E}) \models A$ iff $\mathsf{M}_{(\mathcal{P},\mathcal{E})} \models A$.* □

We now consider the fixpoint formalization of an intuitive semantics of our logic programs. $\mathsf{T}_{(\mathcal{P},\mathcal{E})}$ maps from and into $H/\mathcal{E}$-interpretations and is defined as follows:

$$\begin{aligned}\mathsf{T}_{(\mathcal{P},\mathcal{E})}(I) = \{p(\overline{d}) \in \Pi(H/\mathcal{E}) :\ & \text{there is a closed } H\text{-instance of a clause in } \mathcal{P} \\ & p(\overline{s}) \leftarrow e_1, \cdots, e_n, G \text{ such that} \\ & [\overline{s}] = \overline{d}, \\ & \mathcal{E} \models e_k \text{ for all } k \in [n], \text{ and} \\ & I \models G\}\end{aligned}$$

Lemma 6.9 $\mathsf{T}_{(\mathcal{P},\mathcal{E})}$ *is continuous.* □

Theorem 6.10 $\mathsf{M}_{(\mathcal{P},\mathcal{E})} = \mathsf{T}^{\omega}_{(\mathcal{P},\mathcal{E})}(\phi)$. □

7 Conclusion

We have built a general model theoretic semantics for λProlog and established the least model and least fixed point semantics. We have not given any definition of interpreters for λProlog. But we can prove soundness and completeness of those interpreters developed in [22, 20] by establishing equivalence between the fixed point semantics and the operational semantics of those interpreters based on $\mathcal{P}$-derivations.

References

[1] James H. Andrews. Predicates as parameters in logic programming: A set-theoretic basis. In P. Schroeder-Heister, editor, *Extensions of Logic Programming*, pages 31–47, 1989.

[2] Peter B. Andrews. Resolution in type theory. *The Journal of Symbolic Logic*, 36(3):414–432, 1971.

[3] Peter B. Andrews. General models and extensionality. *The Journal of Symbolic Logic*, 37(2):395–397, 1972.

[4] Peter B. Andrews. General models, descriptions, and choice in type theory. *The Journal of Symbolic Logic*, 37(2):385–394, 1972.

[5] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof.* Academic Press, 1986.

[6] H. P. Barendregt. *The Lambda Calculus.* North-Holland, 1984.

[7] Weidong Chen, Kichael Kifer, and David S. Warren. Hilog: A first-order semantics for higher-order logic programming constructs. In Ewing L. Lusk and Ross A. Overbeek, editors, *Logic Programming Proceedings of North American Conference*, pages 1090–1114, 1989.

[8] Alonzo Church. A formulation of simple theory of types. *The Journal of Symbolic Logic*, 5:56–68, 1940.

[9] D. DeGroot and G. Lindstrom, editors. *Logic Programming: Relations, functions and Equations.* Prentice Hall, 1986.

[10] William Farmer. A partial function version of Church's simple theory of types. *The Journal of Symbolic Logic*, 55:1269–1291, 1990.

[11] Amy P. Felty. *Specifying and Implementing Theorem Provers in a Higher-Order Logic Programming Language.* PhD thesis, University of Pennsylvania, 1989.

[12] John Hannan and Dale Miller. Uses of higher-order unification for implementing program transformers. In Robert A. Kowalski and Kenneth Bowen, editors, *Logic Programming Proceedings of the Fifth International Conference and Symposium*, pages 942–956, 1988.

[13] Leon Henkin. Completeness of the theory of types. *The Journal of Symbolic Logic*, 15:81–91, 1950.

[14] J. Jaffar, J-L. Lassez, and M. J. Maher. A theory of complete logic programs with equality. *J. Logic Programming*, pages 211–223, 1984.

[15] H. Kirchner and W. Wechler, editors. *Algebraic and Logic Programming.* Springer-Verlag, 1990.

[16] John W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1987.

[17] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland, 1978.

[18] Dale Miller. *Proofs in higher-order logic.* PhD thesis, Carnegie-Mellon University, 1983.

[19] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In P. Schroeder-Heister, editor, *Extensions of logic programming*, pages 253–281, 1989.

[20] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. Technical report, University of Pennsylvania, 1989.

[21] J. Donald Monk. *Mathematical Logic.* Springer-Verlag, 1976.

[22] Gopalan Nadathur. *A higher-order logic as the basis for logic programming.* PhD thesis, University of Pennsylvania, 1986.

[23] J. A. Robinson. Mechanizing higher-order logic. *Machine Intelligence*, 4:150–170, 1969.

[24] Joseph R. Shoenfield. *Mathematical Logic.* Addison-Wesley, 1967.