In: J. Heering, K. Meinke, B. Möller, T. Nipkow (eds.): Higher-Order algebra, logic and term rewriting. Lecture Notes in Computer Science **816**. Berlin: Springer 1994, 223–255

Ordered and Continuous Models of Higher-Order Specifications

Bernhard Möller

Institut für Mathematik, Universität Augsburg, D-86135 Augsburg, Germany,
e-mail: moeller@uni-augsburg.de

Abstract. We investigate the existence of continuous and fixpoint models of higher-order specifications. Particular attention is paid to the question of extensionality. We use ordered specifications, a particular case of Horn specifications. The main tool for obtaining continuous models is the ideal completion. Unfortunately, it may destroy extensionality. This problem is inherent: we show that there is no completion method which is guaranteed to preserve extensionality. To restore it, generally a quotient has to be taken. It is shown that under certain conditions this preserves the existence of least fixpoints. Examples of the specification method include the essential concepts of Backus's FP and Hoare's CSP.

1 Introduction

During the last years a number of papers have dealt with the extension of first-order algebraic specifications to higher-order ones (cf. [21, 36, 13, 38, 39, 8, 29, 30, 31, 32, 45, 24, 25]). The basic questions about existence of models are answered by now. In this paper we want to treat one particular approach to the question about existence of models in which recursion equations are solvable. This is especially of interest in connection with the algebraic specification of infinite objects (cf. [28]), for which, in turn, non-strict operations are a crucial concept. Based on the ideas of [40], it has turned out that ordered and continuous algebras (cf. [34, 10, 15]) are a convenient framework for modelling these notions. Therefore we consider higher-order specifications that admit continuous models. This idea is also pursued in [13], but not in [38, 39]. The present treatment is based on [29, 30] where (extensional) higher-order algebras with partially ordered carrier sets are considered. The paper is a reworking of some chapters of [30] based on the first-order reduction in [32] and the model theory presented in [25]. For reasons of space we only include proofs of the most essential theorems; the remaining proofs can be found in [28, 30] and the other references.

2 First-Order Horn Specifications and Their Models

2.1 First-Order Languages, Structures and Terms

In the presentation we follow closely the papers [32] and [25] and use the reduction of higher-order notions to the first-order case. This allows a direct re-use of

the results of [28] on ordered and continuous algebras. To accommodate the order more conveniently we use the framework of Horn specifications (cf. [22, 20, 35]).

A **(first-order) language** $L = (S, F, P, \phi, \psi)$ is a quintuple consisting of a set S of **sorts**, a set F of **operators**, a set P of **predicate symbols** and **arity functions** $\phi : F \to S^+$ and $\psi : P \to S^*$ such that for every $s \in S$ there is an equality predicate symbol $=_s \in P$ with $\psi(=_s) = ss$. An operator $f \in F$ with arity $\phi(f) = s \in S$ is called a **constant**. L is called **algebraic** if P contains no other symbols than the $=_s$ $(s \in S)$. An algebraic language corresponds to a signature.

In the sequel we shall frequently use families of sets, functions, relations etc. Rather than saying that E is a family $(E_i)_{i \in I}$ of entities E_i, we call E an **I-indexed** entity. For instance, and I-indexed set X is a family $(X_i)_{i \in I}$. All set-theoretic notions are extended componentwise to I-indexed entities; for instance, requiring X to be nonempty means requiring $X_i \neq \emptyset$ for all $i \in I$.

An **L-structure** (see e.g. [14]) A consists of the following items:

1. an S-indexed non-empty set, also denoted by A, called the **carrier** of A;
2. for each operator $f \in F$ of arity $\phi(f) = ws$ with $w \in S^*$ and $s \in S$ an **operation** $f^A : A_w \to A_s$, where for $w = s_1 \cdots s_n$ we set $A_w \stackrel{\text{def}}{=} A_{s_1} \times \cdots \times A_{s_n}$;
3. for each predicate symbol $p \in P$ a relation $p^A \subseteq A_{\psi(p)}$.

We call A **normal** if $=^A$ is the S-indexed identity relation on A. The interpretation of a constant f is a function $f^A : \{()\} \to A_s$ from the set consisting only of the empty tuple $()$; as usual it will be identified with its result value on $()$. The class of all normal L-structures is denoted by $\mathrm{STR}(L)$.

For every language L one can construct a **trivial** L-structure $\mathbb{1}_L$ by choosing as the carrier an S-indexed singleton set and assigning to all predicate symbols except $=$ the singleton relation.

Given a language $L = (S, F, P, \phi, \psi)$ and an S-indexed set X of **variables** disjoint from L, the S-indexed set $WL(X)$ of L-terms is the smallest S-indexed set satisfying the following inductive properties:

1. $X \subseteq WL(X)$.
2. For $f \in F$ with $\phi(f) = s_1 \cdots s_n s$ and $t_i \in WL(X)_{s_i}$ we have $f(t_1, \ldots, t_n) \in WL_s(X)$.

$WL(X)$ is made into a normal L-structure by defining the operations in the standard way, interpreting the equality predicate symbols by the respective identity relations and setting $p^{WL(X)} \stackrel{\text{def}}{=} \emptyset$ for all other $p \in P$.

For $X = \emptyset$ we call $WL \stackrel{\text{def}}{=} WL(\emptyset)$ the **Herbrand structure** over L. A sort s is **populated** in L if $WL_s \neq \emptyset$. For an algebraic language the Herbrand structure corresponds to the ground term algebra.

A **valuation** of X in an L-structure A is an S-indexed function $v : X \to A$ that assigns to every variable an element of the respective carrier of A. We denote the set of all valuations of X in A by A^X. A valuation of X in WL is called a **ground term valuation**.

Every valuation $v \in A^X$ is extended into an S-indexed function $\overline{v} : WL(X) \to A$ by

1. $\overline{v}_s(x) \stackrel{\text{def}}{=} v_s(x)$ for $x \in X_s$.
2. $\overline{v}_s(f(t_1, \ldots, t_n)) \stackrel{\text{def}}{=} f^A(\overline{v}_{s_1}(t_1), \ldots, \overline{v}_{s_n}(t_n))$ for $\phi(f) = s_1 \cdots s_n s$ and $t_i \in WL(X)_{s_i}$.

For a ground term t the value $\overline{v}(t)$ is independent of $v \in A^X$; we denote it by t^A.

2.2 Homomorphisms and Term-Generated Structures

An **L-homomorphism** $h : A \to B$ between two L-structures A and B consists of an S-indexed function $h : A \to B$ that satisfies $h_s(f^A(x_1, \ldots, x_n)) = f^B(h_{s_1}(x_1), \ldots, h_{s_n}(x_n))$ for all $f \in F$ with $\phi(f) = s_1 \cdots s_n s$ and $x_i \in A_{s_i}$ and $h(p^A) \subseteq p^B$ for all predicate symbols $p \in P$. An **L-epimorphism** is a surjective L-homomorphism, whereas an **L-isomorphism** is a bijective L-homomorphism. In working with L-homomorphisms we omit L when it is clear from the context.

We write $A \preceq B$ if there is a homomorphism from A to B. $\preceq$ is a pre-order on $\mathrm{STR}(L)$. Likewise, we write $A \cong B$ if there is an isomorphism between A and B.

The interaction between homomorphisms and valuations is given by the freeness of $WL(X)$ over X (see e.g. [14]), viz. by

Lemma 1. *Consider $A, B \in \mathrm{STR}(L)$ and a valuation $v \in A^X$.*

1. *$\overline{v} : WL(X) \to A$ is an L-homomorphism.*
2. *If $h : A \to B$ is an L-homomorphism then $h \circ \overline{v} = \overline{h \circ v}$.*

Let $K \subseteq \mathrm{STR}(L)$ be a class of L-structures. A structure $I \in K$ is **initial** in K if for all $A \in K$ there is exactly one homomorphism from I to A. $Z \in K$ is **terminal** in K if for all $A \in K$ there is exactly one homomorphism from A to Z. For every $A \in \mathrm{STR}(L)$ the functions $\iota_s^A : u \mapsto u^A$ define a unique L-homomorphism $\iota^A : WL \to A$, so that WL is initial in $\mathrm{STR}(L)$.

If ι^A is surjective then A is called **term-generated** (cf. e.g. [3, 4]), **reachable** (cf. e.g. [32]) or **minimal** (cf. e.g. [25]). This means that every element of the carrier of A can be denoted by an L-ground-term or, in other words, that it can be obtained by applying finitely many operations of the structure. By $\mathrm{GEN}(L)$ we denote the class of all term-generated L-structures. Since WL itself is term-generated, it is also initial in $\mathrm{GEN}(L)$.

2.3 Horn Specifications and Their Models

Consider a language $L = (S, F, P, \phi, \psi)$ and an S-indexed set X of variables. First-order formulas over L and X and the validity of a formula Φ in an L-structure A, denoted by $A \models \Phi$, are defined as usual. For binary relation symbols infix notation will be used.

A **(positive) literal** over L is a formula of the form $p(t_1, \ldots, t_m)$ where $p \in P$ with $\psi(p) = s_1 \cdots s_m$ and $t_i \in WL(X)_{s_i}$ $(1 \leq i \leq n)$. If $t_i \in WL_{s_i}$ for all i then $p(t_1, \ldots, t_m)$ is called a **ground** literal. A **Horn formula** has the form

$$\bigwedge_{i \in I} Q_i \Rightarrow Q \ ,$$

where I is a finite or infinite index set and the Q_i and Q are literals over L.

The **equivalence axioms** over L are the Horn formulas

$$\begin{gathered} x_s =_s x_s \ , \\ x_s =_s y_s \wedge y_s =_s z_s \Rightarrow x_s =_s z_s \ , \\ x_s =_s y_s \Rightarrow y_s =_s x_s \end{gathered}$$

for variables $x_s, y_s, z_s \in X_s$. The **substitutivity axioms** over L are the Horn formulas

$$\bigwedge_{i=1}^{n} x_i =_{s_i} y_i \Rightarrow f(x_1, \ldots, x_m) =_s f(y_1, \ldots, y_m)$$

for all $f \in F$ with $\phi(f) = s_1 \cdots s_m s$ and variables $x_{s_i}, y_{s_i} \in X_{s_i}$, and

$$\bigwedge_{i=1}^{n} x_i =_{s_i} y_i \wedge p(x_1, \ldots, x_n) \Rightarrow p\langle y_1, \ldots, y_n \rangle$$

for all $p \in P$ with $\psi(p) = s_1 \cdots s_n$ and variables $x_{s_i}, y_{s_i} \in X_{s_i}$. The **congruence axioms** over L are the equivalence and substitutivity axioms over L.

A **Horn specification** $K = (L, E)$ consists of a language L and a set E of Horn formulas over L, called the **axioms** of K, which includes the congruence axioms for L. **Equational specifications** are Horn specifications over algebraic languages.

An L-structure A is called a **model** of K if all axioms in E are valid in A and A is normal. We denote by $\mathrm{MOD}(L)$ the class of all models of L, and by $\mathrm{GEN}(L)$ the class of term-generated models of L.

2.4 Quotients

Models for equational specifications can be constructed as quotients of the term algebra by suitable congruences. We generalise this notion to arbitrary structures.

Consider an L-structure A. A family $R = (p^R)_{p \in P}$ with $p^A \subseteq p^R \subseteq A_{\psi(p)}$ is called a **pre-congruence** on A. For a positive literal $p(t_1, \ldots, t_n)$, valuation $v \in A^X$ and pre-congruence R on A we define

$$v, R \models p(t_1, \ldots, t_n) \overset{\text{def}}{\Leftrightarrow} (\overline{v}(t_1), \ldots, \overline{v}(t_n)) \in p^R \ .$$

This relation is extended inductively to general first-order formulas as usual. Then R **satisfies** a first-order formula Φ iff $v, R \models \Phi$ for all valuations $v \in A^X$.

By $\mathbb{U}^A$ we denote the universal pre-congruence on an L-structure A with $p^{\mathbb{U}_A} \stackrel{\text{def}}{=} A_{\psi(p)}$ for all $p \in P$. We say that a set of pre-congruences on A is **lattice-forming** if it is closed under intersection and contains $\mathbb{U}^A$. Such a set forms a complete lattice under relation inclusion where the greatest lower bound of a set G of pre-conguences on A is their intersection $\cap G$. Fundamental for the theory of Horn specifications is

Theorem 2. *Consider an L-structure A and a set E of Horn formulas over L. The set of pre-congruences on A that satisfy E is lattice-forming.*

Taking $E = \emptyset$ we obtain that the set of pre-congruences on A is lattice-forming. Its least element is R^A with $p^{R^A} = p^A$ for all $p \in P$.

A pre-congruence R is called a **congruence** if it satisfies the congruence axioms. The set of pre-congruences on A is denoted by $\mathrm{CG}(A)$. By Theorem 2 it is lattice-forming. The **quotient** A/R of A by $R \in \mathrm{CG}(A)$ is defined by setting

1. $(A/R)_s \stackrel{\text{def}}{=} \{[x] : x \in A_s\}$ where $[x]$ is the equivalence class of x w.r.t. $=_s^R$;
2. $f^{A/R}([a_1], \ldots, [a_n]) \stackrel{\text{def}}{=} [f^A(a_1, \ldots, a_n)]$;
3. $([a_1], \ldots, [a_n]) \in p^{A/R} \stackrel{\text{def}}{\Leftrightarrow} (a_1, \ldots, a_n) \in p^R$.

The congruence axioms ensure well-definedness of this construction. It is clear that $\mathbb{U}^A$ is a congruence and that $A/\mathbb{U}^A \cong \mathbb{1}_H$. Moreover, we have

Lemma 3. *A Horn formula Q is valid in the quotient A/R iff R satisfies Q.*

For L-structures A and B and homomorphism $h : A \to B$ we define the **kernel** R_h on A by $(a_1, \ldots, a_n) \in p^{R_h} \stackrel{\text{def}}{\Leftrightarrow} (h(a_1), \ldots, h(a_n)) \in p^B$. Then R_h is a congruence on A.

Theorem 4. *Consider $A, B \in \mathrm{STR}(L)$.*

1. *If $h : A \to B$ is an epimorphism then $A/R_h \cong B$.*
2. *For $R_1, R_2 \in \mathrm{CG}(A)$ the inclusion $R_1 \subseteq R_2$ implies $A/R_1 \preceq A/R_2$.*
3. *$A \in \mathrm{GEN}(L)$ iff $A \cong WL/R$ for some $R \in \mathrm{CG}(WL)$. Moreover, then $R = R_{\iota^A}$.*
4. *If $A, B \in \mathrm{GEN}(L)$ we have $A \preceq B$ iff $R_{\iota^A} \subseteq R_{\iota^B}$. Thus, $\mathrm{GEN}(L)/\preceq$ is order-isomorphic to $(\mathrm{CG}(WL), \subseteq)$ and hence a complete lattice.*
5. *There is at most one homomorphism from one term-generated structure to another. If there is one, it is an epimorphism.*
6. *If $A, B \in \mathrm{GEN}(L)$ then $A \cong B$ iff $A \preceq B$ and $B \preceq A$.*
7. *$\mathrm{GEN}(L)$ is closed under quotients.*

We can use quotients to construct models:

Theorem 5. *Consider a Horn specification $K = (L, E)$.*

1. *K has an initial model. It is the quotient of WL by the least congruence on WL that satisfies E.*

2. *The set of isomorphism classes of term-generated models of L forms a complete lattice w.r.t. $\preceq$.*

A special case is the well-known result that the isomorphism classes of term-generated models of an equational specification form a complete lattice under the homomorphism ordering.

2.5 A Deductive Calculus

Consider now a Horn specification $K = (L, E)$ with $L = (S, F, P, \phi, \psi)$. For a term valuation $v \in WL(X)^X$ we extend $\overline{v}$ to Horn formulas by setting for literals $\overline{v}(p(t_1, \ldots, t_n)) \stackrel{\text{def}}{=} p(\overline{v}(t_1), \ldots, \overline{v}(t_n))$ and then $\overline{v}(\bigwedge_{i \in I} Q_i \Rightarrow Q) \stackrel{\text{def}}{=} \bigwedge_{i \in I} \overline{v}(Q_i) \Rightarrow \overline{v}(Q)$. Then the following inference rules are sound in $\mathrm{MOD}(L)$:

(Axiom) $\dfrac{}{\Phi}$ for $\Phi \in E$

(Instantiation) $\dfrac{\Phi}{\overline{v}(\Phi)}$ for $\Phi \in E$ and $v \in WL(X)^X$

(ModusPonens) $\dfrac{\bigwedge_{j \in J} Q_j \Rightarrow Q \qquad Q_i \ (i \in I)}{\bigwedge_{j \in J \setminus I} Q_j \Rightarrow Q}$

These rules form a complete calculus:

Theorem 6. *Let $K = (L, E)$ be a Horn specification, I an initial structure in $\mathrm{GEN}(K)$ and Q a literal over L. Then the following equivalences hold:*

1. *$L \vdash Q$ iff $\mathrm{MOD}(L) \models Q$.*
2. *If Q is a ground literal then $I \models Q$ iff $L \vdash Q$ iff $\mathrm{GEN}(K) \models Q$.*

This theorem shows also the distinction between term-generated models and arbitrary ones: In $\mathrm{GEN}(K)$ usually more formulas are valid than in $\mathrm{MOD}(K)$, since the principle of term-induction (see below) is available. Thus not all formulas valid in $\mathrm{GEN}(K)$ need be provable using only the above calculus. However, for ground terms provability and validity in $\mathrm{GEN}(K)$ coincide; moreover, they are equivalent to the validity in initial models.

To conclude this section, we now state the above-mentioned principle of **term induction**.

Theorem 7 (Term Induction). *Let L be a language and Φ be a first-order formula over L containing exactly one free variable $x_s \in X_s$ for some $s \in S$. Then Φ is valid in a class $K \subseteq \mathrm{GEN}(L)$ of term-generated structures for L iff the following property holds: For all $f \in F$ with $\phi(f) = s_1 \cdots s_n s$ the assumptions $K \models \Phi[x_j/x_s]$ $(j \in J)$ imply $K \models \Phi[u(x_1, \ldots, x_n)/x_s]$, where $x_i \in X_{s_i}$ $(1 \leq i \leq n)$ are variables not occurring in Φ and $J \stackrel{\text{def}}{=} \{i : s_i = s\}$.*

3 The Ordered Case

We turn now to the special case where the only predicate symbols besides the equality symbols are order relation symbols.

3.1 Pre-Orders

A **pre-order** is a reflexive and transitive binary relation $\leq$ on some set M; an **order** is an antisymmetric pre-order. E.g. in [5] one finds

Lemma 8. *For a pre-ordered set $(M, \leq)$ the relation $\sim_\leq$ defined by*

$$x \sim_\leq y \stackrel{\text{def}}{\Leftrightarrow} x \leq y \wedge y \leq x$$

is an equivalence relation on M.

We denote by $[x]_\leq$ the equivalence class of x w.r.t. $\sim_\leq$ and by $[M]_\leq$ the set $\{[x]_\leq : x \in M\}$. If no ambiguity arises, the index $\leq$ will be dropped. For a pre-ordered set $(M, \leq)$ the set $[M]$ together with the relation

$$[x] \leq [y] \stackrel{\text{def}}{\Leftrightarrow} x \leq y$$

is an ordered set called the **quotient** of $(M, \leq)$ by $\sim_\leq$; we denote it by $M/\leq$.

3.2 Ordered Specifications and Structures

A language $L = (S, F, P, \phi, \psi)$ is called **ordered** if $P = \{=_s, \leq_s : s \in S\}$ with $\psi(=_s) = \psi(\leq_s) = ss$. If L is ordered then all L-homomorphisms between L-structures are monotonic.

A Horn specification $K = (L, E)$ is an **ordered specification** if L is ordered and E contains the **order axioms**

$$\begin{gathered} x_s \leq_s x_s \ , \\ x_s \leq_s y_s \wedge y_s \leq_s z_s \Rightarrow x_s \leq_s z_s \ , \\ x_s \leq_s y_s \wedge y_s \leq_s x_s \Rightarrow x_s =_s y_s \end{gathered}$$

for all $s \in S$ and variables $x_s, y_s, z_s \in X_s$. The congruence and order axioms are called the **standard axioms** of K, whereas the other axioms are **non-standard axioms**. In the sequel we shall omit the indices of the equality and order predicate symbols.

Note that by the substitutivity axioms the antisymmetry axiom can be strengthened to

$$x \leq y \wedge y \leq x \Leftrightarrow x = y \ .$$

So in any L-structure A satisfying the order axioms $\leq^A$ is a pre-order and $=^A$ coincides with $\sim_{\leq^A}$ as defined in Lemma 8. If L is normal then $\leq^A$ is an order. Therefore we define: an **ordered L-structure** is a normal L-structure satisfying

the order axioms. Hence, again by Lemma 8 any quotient by a congruence that satisfies the order axioms is an ordered L-structure.

As a simple example for an ordered specification consider the one having as non-standard axioms only the formulas

$$x \leq y \Rightarrow y \leq x$$

for all sorts. The models of this specification are exactly the trivially ordered structures. Thus we retrieve here in a second way the above-mentioned result on isomorphism classes of models of equational specifications.

3.3 Groundedness and Strictness

In connection with programming language semantics frequently "undefined situations" are modelled by special $\perp$-elements in the respective domains; these elements are considered as carrying no information, i.e., as least elements in the information contents ordering (cf. [40]).

We call an ordered set **grounded** if it contains a least element. A language $L = (S, F, P, \phi, \psi)$ is **grounded** if for all $s \in S$ there are distinguished constants $\perp_s$ with $\phi(\perp_s) \stackrel{\text{def}}{=} s$. If L is grounded then all L-homomorphisms are **strict**: Since each $\perp_s$ is a special constant, we have for $A, B \in \mathrm{STR}(L)$ and $h : A \to B$ that $h_s(\perp_s^A) = \perp_s^B$.

We call a specification $K = (L, E)$ **grounded** if L is grounded and E contains the **groundedness axioms**

$$\perp_s \leq_s x_s$$

for all $s \in S$ and variables $x_s \in X_s$. To abbreviate our specification texts in examples, we use the keyword grounded; the operators $\perp_s$ as well as the groundedness axioms are then omitted from the specification text.

For many operations the concept of strictness in a certain argument is very important: Operationally speaking, this argument is crucial for computing the operation; as long as no information about this argument is available, the operation will be undefined. Given a grounded language L and an operator $f \in F$ with $\phi(f) = s_1 \cdots s_n s$ we can specify strictness in the kth argument by the axiom

$$f(x_{s_1}, \ldots, x_{s_{k-1}}, \perp_{s_k}, x_{s_{k+1}}, \ldots, x_{s_n}) \leq \perp_s.$$

3.4 Monotonicity

A second class of specifications is concerned with monotonic structures.

We call a specification $K = (L, E)$ **monotonic** if $L = (S, F, P, \phi, \psi)$ is ordered and E includes the **monotonicity axioms**

$$\bigwedge_{i=1}^{n} x_i \leq y_i \Rightarrow f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n).$$

for all $f \in F$ with with $\phi(f) = s_1 \cdots s_m s$ and variables $x_{s_i}, y_{s_i} \in X_{s_i}$. An L-structure is **monotonic** if it satisfies the monotonicity axioms. The **standard**

axioms for a monotonic specification are the congruence, order and monotonicity axioms. We use the keyword monotonic to mean that the respective specification implicitly contains the monotonicity axioms.

A congruence R on an L-structure A is **monotonic** if it satisfies the monotonicity axioms. From Lemma 3 we obtain

Corollary 9. *A/R is monotonic iff R is monotonic.*

The specialisation of Theorem 5 for monotonic specifications was obtained in [26, 28].

A special case of ordered specifications are **inequational** specifications, i.e., monotonic specifications in which the non-standard axioms are only of the form $u \leq w$ with $u, w \in L(X)$. For these we have the stronger result (see also [6])

Theorem 10. *Let $K = (L, E)$ be an inequational specification.*

1. $\mathrm{MOD}(L)$ *is closed under epimorphisms and hence under quotients.*
2. $\mathrm{GEN}(K)/\preceq$ *is a complete sublattice of* $\mathrm{GEN}(L)/\preceq$.

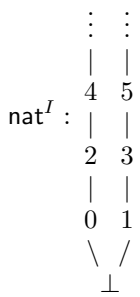
The specification with grounded language L requiring only the groundedness and monotonicity axioms has as its initial model the Herbrand structure WL ordered by the syntactic approximation relation $\sqsubseteq$ first described in [34]: For two ground terms u_1, u_2 we have $u_1 \sqsubseteq u_2$ iff u_2 results from u_1 by replacing zero or more occurrences of $\bot$ by other ground terms. This is due to the fact that the order in the initial model is the least order comprising the relation generated by the groundedness and monotonicity axioms.

As another example we give a specification of an ordering that will be used below:

Example 1. Consider

spec S
  grounded monotonic
  sort nat
  $0, 1$: nat
  plustwo : nat $\to$ nat
  axioms $\mathit{plustwo}(\bot) = \bot$
    $0 \leq \mathit{plustwo}(0)$
    $1 \leq \mathit{plustwo}(1)$
endspec

The initial model I of S has the carrier



□

3.5 Completeness and Continuity

We head now for continuous structures, i.e., structures in which fixpoint equations are solvable; they allow thus a mathematical semantics for recursion equations (see e.g. [15]).

Let us give the necessary order-theoretic notions. Consider two subsets $S, T \subseteq M$ of a pre-ordered set $(M, \leq)$. We write $S \leq T$ if $S \times T \subseteq \leq$, i.e., if $\forall\, x \in S : \forall\, y \in T : x \leq y$. For singletons we write $x \leq T$ and $S \leq y$ rather than $\{x\} \leq T$ and $S \leq \{y\}$.

The sets of **upper** and **lower** bounds of a subset $N \subseteq M$ are defined by

$$\mathsf{upb}_{\leq} N \stackrel{\text{def}}{=} \{y \in M : N \leq y\} \ , \qquad \mathsf{lwb}_{\leq} N \stackrel{\text{def}}{=} \{x \in M : x \leq N\} \ .$$

An upper (lower) bound of N that lies in N is called a **greatest (least)** element of N, and we set

$$\mathsf{gst}_{\leq} N \stackrel{\text{def}}{=} N \cap \mathsf{upb}_{\leq} N \ , \qquad \mathsf{lst}_{\leq} N \stackrel{\text{def}}{=} N \cap \mathsf{lwb}_{\leq} N \ .$$

Note that all greatest (least) elements of a subset are identified in $M/\leq$.

If the set of all upper (lower) bounds of N possesses least (greatest) elements, these elements are called **least upper bounds (greatest lower bounds)** of N. We set

$$\mathsf{lub}_{\leq} N \stackrel{\text{def}}{=} \mathsf{lst}_{\leq} \mathsf{upb}_{\leq} N \ , \qquad \mathsf{glb}_{\leq} N \stackrel{\text{def}}{=} \mathsf{gst}_{\leq} \mathsf{lwb}_{\leq} N \ .$$

In an ordered set, least upper bounds and greatest lower bounds are unique if they exist. In the sequel we shall omit the indices $\leq$ if $\leq$ is clear from the context.

A subset $D \subseteq M$ of a pre-ordered set $(M, \leq)$ is **directed** if $D \neq \emptyset$ and $D \cap \mathsf{upb}\,\{x, y\} \neq \emptyset$ for all $x, y \in D$. A pre-ordered set $(M, \leq)$ is **Δ-complete** if M is grounded and $\mathsf{lub}\, D \neq \emptyset$ for every directed subset $D \subseteq M$.

Let $(M_1, \leq_1)$, $(M_2, \leq_2)$ be pre-ordered sets. A function $f : M_1 \to M_2$ is **continuous** if for every directed $D \subseteq M_1$ we have $f(\mathsf{lub}_{\leq_1} D) \subseteq \mathsf{lub}_{\leq_2} f(D)$. In the case of an order we may replace the inclusion by an equality. A continuous function is monotonic.

The following well-known fixpoint theorem is the basis for the solution of recursion equations:

Theorem 11 (Knaster, Tarski, Kleene). *Let $(M, \sqsubseteq)$ be a Δ-complete ordered set and $f : M \to M$ be continuous. Then f has a least fixpoint, viz.*

$$\mathit{fix}(f) \stackrel{\text{def}}{=} \mathsf{lub}\, f^*(\bot) \ ,$$

where $f^*(x) \stackrel{\text{def}}{=} \{f^i(x) : i \in \mathbb{N}\}$.

Now we carry over these notions to structures. Contrary to groundedness, strictness and monotonicity they cannot be characterised by Horn formulas. An L-structure A is called

1. **pre-complete** if all its operations are continuous in all arguments;

2. **continuous** if it is pre-complete and its carrier is Δ-complete.

Note that a pre-complete structure is monotonic.

Lemma 12. *For a given language L the Herbrand structure WL is pre-complete.*

We consider now particular congruences that fit these specialised notions of structures. Consider an ordered language L and an ordered L-structure A. We call a congruence $R \in \mathrm{CG}(A)$

1. **pre-complete** if for every operation $f^A : A_w \to A_s$ and every $\leq^R$-directed subset $D \subseteq A_w$ we have $f^A(\mathsf{lub}_{\leq^R} D) \subseteq \mathsf{lub}_{\leq^R} f^A(D)$;
2. **lub-compatible** if for every $\leq^A$-directed subset $D \subseteq A_s$ we have $\mathsf{lub}_{\leq^A} D \subseteq \mathsf{lub}_{\leq^R} D$. This is equivalent to $\mathsf{lub}_{\leq^A} D \leq^R \mathsf{upb}_{\leq^R} D$.

Theorem 13. *Let R be a congruence on $A \in \mathrm{STR}(L)$.*

1. *A/R is pre-complete iff R is pre-continuous.*
2. *Let $h : A \to B$ be a homomorphism.*
   (a) *If h is continuous then R_h is lub-compatible.*
   (b) *If h is an epimorphism and R_h is lub-compatible then h is continuous. In particular, for lub-compatible R the canonical homomorphism $k : A \to A/R$ given by $k(a) \stackrel{\mathrm{def}}{=} [a]_{=^R}$ is continuous.*

For the proof see [11]. We denote by Mon^A and Pre^A the sets of monotonic and pre-continuous congruences on A.

Lemma 14. Mon^A *and* Pre^A *are lattice-forming.*

3.6 Completions

Our method of obtaining continuous structures is via completing term-generated monotonic structures. The requirement of monotonicity can be relaxed (see [33]), but for simplicity we shall here work assuming it.

Call a Δ-complete set M a **completion** of an ordered set L if

1. there is an order-embedding $\iota : L \to M$;
2. for every continuous function $h : L \to N$ into a Δ-complete set N there is a unique continuous function $\hat{h} : M \to N$ with $\hat{h} \circ \iota = h$, i.e., which extends h to M.

The completion is called **lub-preserving** if the embedding ι is continuous. Then it preserves the least upper bounds already existing in L. Thus, as pointed out in [11], there can only be a lub-preserving completion of A if A is pre-complete, i.e., if its operations are already continuous at the existing least upper bounds.

An important, however in general not lub-preserving, completion is the ideal completion (see e.g. [5, 12]). To define it, we need some auxiliary notions.

Consider a pre-ordered set $(M, \leq)$. The **downward closure** of a subset $N \subseteq M$ is

$$N^{\leq} \stackrel{\text{def}}{=} \{x \in M : \exists\, y \in N : x \leq y\} \ .$$

By a **cone** of M we mean a downward closed subset, i.e., a set $N \subseteq M$ such that $N = N^{\leq}$. An **ideal** of an ordered set $(M, \leq)$ is a directed cone in M. For a directed subset $D \subseteq M$ the cone $D^{\leq}$ is the **ideal generated by** D. Note that $\mathsf{lub}\, D = \mathsf{lub}\, D^{\leq}$. For $x \in M$ we write $x^{\leq}$ instead of $\{x\}^{\leq}$.

An element x of M is **finite** (**compact**) if for every directed set $D \subseteq M$ with $x \leq \mathsf{lub}\, D$ we have also $x \leq z$ for some $z \in D$. Equivalently, x is finite iff for every ideal $I \subseteq M$ with $x \leq \mathsf{lub}\, I$ we have $x \in I$. $(M, \leq)$ is **inductive** (**algebraic**) if every element of M is the least upper bound of a directed set of finite elements. A non-finite element of an inductive set is called a **limit point** or an **infinite element**.

Theorem 15. *Let $(M, \leq)$ be an ordered set and $I(M)$ be the set of ideals of M.*

1. *The set $(I(M), \subseteq)$ ordered by set inclusion is an inductive completion of M, the finite elements being the ideals $x^{\leq}$ for $x \in M$. M is embedded into $I(M)$ via the function $i : x \mapsto x^{\leq}$.*
2. *For every monotonic (not necessarily continuous) function $h : M \to N$ into a Δ-complete set $(N, \leq)$ there is a unique continuous function $\hat{h} : I(M) \to N$ extending h, i.e., with $\hat{h}(x^{\leq}) = h(x)$. $\hat{h}$ is given by $\hat{h}(I) = \mathsf{lub}\, h(I)$ for $I \in I(M)$; hence $\hat{h}(D^{\leq}) = \mathsf{lub}\, h(D)$ for directed D.*

For the proof see e.g. [11]. We call $(I(M), \subseteq)$ the **ideal completion** of M.

Let us now apply these notions to L-structures. A continuous L-structure B is a **completion** of a monotonic L-structure A if there is an order-embedding homomorphism $\iota : A \to B$ such that every continuous homomorphism $h : A \to C$ from A into a continuous L-structure C extends uniquely to a continuous homomorphism $\hat{h} : B \to C$ in the sense that $\hat{h} \circ \iota = h$. Again, the completion is called **lub-preserving** if ι is continuous.

Now every grounded monotonic L-structure A may be embedded into an inductive continuous L-structure A^{∞} using the ideal completion. Define

$$\begin{aligned} A^{\infty}_{\mathsf{t}} &\stackrel{\text{def}}{=} I(A_{\mathsf{t}}) && \text{ordered by inclusion,} \\ f^{A^{\infty}}(I_1, \ldots, I_n) &\stackrel{\text{def}}{=} f^{A}(I_1, \ldots, I_n)^{\leq^A} && (f \in F) \ . \end{aligned}$$

Theorem 16. *1. A^{∞} is an inductive completion of A.*

2. *Every* monotonic *(not necessarily continuous) homomorphism $h : A \to B$ into a continuous L-structure B extends uniquely into a continuous homomorphism $\hat{h} : A^{\infty} \to B$ such that $\hat{h} \circ \iota = h$.*

We call A^{∞} the **ideal completion** of A. Its construction may be applied to monotonic structures that are not pre-complete. It resolves non-continuities by introducing additional limit points into the carriers and by suitably extending

the functions to these limit points. The price for this is that in general it is not lub-preserving.

A different completion method which is based on techniques in [23] and [11] and which is lub-preserving is discussed in [30]. We do not describe it here, since the construction is fairly complicated and still does not give a satisfactory construction of continuous models, as we shall see below.

3.7 Fixpoint Structures

We now study structures in which fixpoint equations are solvable.

Let L be a grounded language and X be an S-indexed set of variables disjoint from L. A **system of recursion equations** over L and X is a valuation e of X in $WL(X)$. It may be viewed as the S-indexed set of equations

$$(\{x = e_s(x) : x \in X_s\})_{s \in S}.$$

Over any L-structure A such a system can be conceived of as a set of conditions about a family of elements of A, viz. about a valuation v of X in A. Thus we say that a valuation $v \in A^X$ is a **solution** of the system e if for all x we have $v_s(x) = \overline{v}(e_s(x))$.

Given an L-structure A, we now associate a **valuation transformer** $e_A : A^X \to A^X$ with a system e by setting for $v \in A^X$

$$e_A(v) \stackrel{\text{def}}{=} \overline{v} \circ e.$$

Thus the transformed valuation associates with each variable the value of the corresponding right hand side in e under the given valuation v. Then $v \in A^X$ is a solution of e iff v is a fixpoint of e_A.

To this end we need to order valuations. We use the standard pointwise order on functions. More generally, for a family $(M_i, \leq_i)_{i \in I}$ of pre-ordered sets, the product $\prod_{i \in I} M_i$ carries the **pointwise** pre-order given by $(x_i)_{i \in I} \leq (y_i)_{i \in I}$ iff $\forall\ i \in I : x_i \leq_i y_i$. For elements f, g of a function space $M \to N = \prod_{x \in M} N$ with pre-ordered set $(N, \leq)$ we therefore have $f \leq g \Leftrightarrow \forall\ x \in M : f(x) \leq g(x)$.

Lemma 17. *Consider a continuous L-structure A.*

1. *The set A^X is Δ-complete under the pointwise order on valuations.*
2. *For any system e of recursion equations over L and X the valuation transformer e_A is continuous.*

Hence by Theorem 11 $\mathit{fix}(e_A)$ is the least solution of e in A. If we set

$$\Omega^A(x) \stackrel{\text{def}}{=} \bot^A \text{ for all } x \in X,$$

we obtain

$$\mathit{fix}(e_A) = \mathsf{lub}\, e_A^*(\Omega^A)\ .$$

Thus, over continuous structures fixpoint equations are solvable. However, for the mere process of solving equations a continuous structure generally contains much too many "superfluous" least upper bounds. More liberally, we call a grounded and monotonic structure A a **fixpoint structure** if for every system e of recursion equations $\mathsf{lub}\, e_A^*(\Omega^A) \neq \emptyset$ and consists of a fixpoint of e_A. Note that, by monotonicity of A, it then is the least fixpoint of e_A.

Theorem 18. *1. Every continuous structure is a fixpoint structure.*

2. *Let A be a fixpoint structure and $h : A \to B$ be a strict continuous homomorphism. Then B is again a fixpoint structure. Moreover, for a system e of recursion equations we have $h \circ fix(e_A) = fix(e_B)$.*
3. *The class of fixpoint structures is closed under quotients modulo lub-compatible congruences.*

Proof. 1. is trivial.

2. For a system e of recursion equations and a valuation $v \in A^X$ we have, by Lemma 1, $h \circ e_A(v) = h \circ \overline{v} \circ e = \overline{h \circ v} \circ e = e_B(h \circ v)$. By induction it now follows that also $h \circ e_A^*(v) = e_B^*(h \circ v)$.
   Moreover, by strictness of h, $h \circ \Omega^A = \Omega^B$. Hence, by continuity of h,
$$h \circ fix(e_A) = h \circ \mathsf{lub}\, e_A^*(\Omega^A) \subseteq \mathsf{lub}\, h \circ e_A^*(\Omega^A) = \mathsf{lub}\, e_B^*(\Omega^B) = fix(e_B)\ .$$
3. is immediate from 2. and Theorem 13.3.

□

4 Adding Higher Order

4.1 Higher-Order Languages and Structures

We shall now, following [32, 25], introduce higher-order languages and structures as particular instances of the corresponding first-order notions. To keep the presentation simple, we do not introduce product types. The reason for this is that an n-ary product behaves exactly like a function type over a domain of cardinality n. So product types would lead to a duplication in all definitions without adding extra power. Moreover, in function arguments they can always be eliminated by currying. A formal treatment can be found in [25].

We first define (higher-order) types. Assume that a non-empty set B of **basic types** is given. The set $T(B)$ of **types** over B is the smallest set such that $B \subseteq T(B)$ and for $\mathsf{s}, \mathsf{t} \in T(B)$ also $\mathsf{s} \to \mathsf{t} \in T(S)$. As customary we assume that $\to$ associates to the right and hence abbreviate a type $\mathsf{s}_1 \to (\mathsf{s}_2 \to \cdots (\mathsf{s}_n \to \mathsf{s}_{n+1}) \cdots)$ by $\mathsf{s}_1 \to \mathsf{s}_2 \to \cdots \mathsf{s}_n \to \mathsf{s}_{n+1}$.

A **higher-order language (hol)** is an ordered language $H = (S, F, P, \phi, \psi)$ satisfying the following conditions:

1. $S \subseteq T(B)$ for some set B of basic types. This set can be uniquely reconstructed from S and is denoted by $B(H)$.

2. For every populated type $\mathsf{s} \to \mathsf{t} \in S$ there is an operator $\alpha^{\mathsf{s}\to\mathsf{t}} \in F$ with $\phi(\alpha^{\mathsf{s}\to\mathsf{t}}) = (\mathsf{s} \to \mathsf{t})\,\mathsf{s}\,\mathsf{t}$, called **application operator**.
3. All other operators in F are constants.
4. S is exactly the set of types populated in H.
5. All basic types are populated in H.
6. Whenever a type $\mathsf{s} \to \mathsf{t}$ is populated in H, so is s (and hence also t). Then for every term denoting a functional object there is also at least one term denoting an argument to that functional object. This can always be achieved by adding suitable constants of all types.

The latter three conditions serve to exclude empty sets in the carriers of H-structures. Note that a grounded language automatically satisfies these requirements.

In the sequel we shall drop the type superscripts as they will always be clear from the context. We will also use a simplified notation for terms over a hol: we abbreviate $\alpha(t, u)$ by $t\,u$ and assume, as customary, that juxtaposition associates to the left.

A hol is called **essentially of first order** if its constants are only of types of the form s or $\mathsf{s}_1 \to \cdots \mathsf{s}_n \to \mathsf{s}_{n+1}$, with $\mathsf{s}, \mathsf{s}_i \in B$, and its only non-constant operators are the respective application operators α. Such hols correspond to the first-order languages defined in Section 2.1: a constant $f : \mathsf{s}_1 \to \cdots \mathsf{s}_n \to \mathsf{s}_{n+1}$ represents an operator f with $\phi(f) = \mathsf{s}_1\mathsf{s}_2 \cdots \mathsf{s}_n\mathsf{s}_{n+1}$.

Consider now a hol H. In every H-structure A each element $f \in A_{\mathsf{s}\to\mathsf{t}}$ of a carrier of functional type $\mathsf{s} \to \mathsf{t}$ induces a function $[\![f]\!]^A : A_{\mathsf{s}} \to A_{\mathsf{t}}$ given by

$$[\![f]\!]^A(x) \stackrel{\text{def}}{=} \alpha^A(f, x)$$

for $x \in A_{\mathsf{s}}$; it is the curried form of α^A. Note that this function cannot be denoted by an operator in the language, since the corresponding "higher-order arity" does not exist. We shall frequently use this function instead of α^A; if A is clear from the context we shall omit the superscript ^{A}.

The **function order axioms** over a hol H are the formulas

$$f \leq g \;\Rightarrow\; f\,x \leq g\,x$$

with $f, g \in X_{\mathsf{s}\to\mathsf{t}}$ and $x \in X_{\mathsf{s}}$ for all populated functional types $\mathsf{s} \to \mathsf{t}$ in H. An H-structure A is called a **higher-order H-structure (H-hos)** if it is ordered and satisfies the function order axioms. Equivalently, A is an H-hos if all functions

$$[\![\_]\!]^A : A_{\mathsf{s}\to\mathsf{t}} \to (A_{\mathsf{s}} \to A_{\mathsf{t}})$$

are monotonic. This condition ensures that the ordering on carriers of functional types conforms to the pointwise function space ordering. The carriers A_{s} with $\mathsf{s} \in B(H)$ are called the **basic carriers** of A. The class of all H-hos is denoted by $\mathrm{HOS}(H)$. A similar notion occurs already in [37].

From our general setting we also obtain a notion of term-generated hoss. Note that a homomorphism between hoss also is a homomorphism w.r.t. $[\![\_]\!]$.

Of particular interest are monotonic (pre-complete) hoss: there all functions $[\![f]\!]$ are monotonic (continuous). In a continuous hos in addition the functions $[\![\_]\!]$ are continuous.

Finally, we study quotients of hoss. Consider a hol $H = (S, F, P, \phi, \psi)$ and an H-hos A. A congruence $R \in \mathrm{CG}(A)$ is called a **higher-order congruence (hop)** if it satisfies the order and the function order axioms so that all functions $[\![\_]\!]^A$ are monotonic w.r.t. $\leq^R$. The set of all hops on A is denoted by $\mathrm{HCG}(A)$. Since the function order axioms are Horn formulas, we obtain from Lemma 3 and Theorem 2

Corollary 19. *Consider $A \in \mathrm{STR}(H)$ and $R \in \mathrm{CG}(A)$.*

1. *$A/R \in \mathrm{HOS}(A)$ iff $R \in \mathrm{HCG}(A)$.*
2. *$\mathrm{HCG}(A)$ is lattice-forming.*

4.2 Extensionality

Consider a hol $H = (S, F, P, \phi, \psi)$. The **extensionality axioms** over H are the formulas

$$(\forall\ \mathsf{s}\ x : f\,x \leq g\,x) \Rightarrow f \leq g$$

with $f, g \in X_{\mathsf{s}\to\mathsf{t}}$ and $x \in X_{\mathsf{s}}$ for all populated functional types $\mathsf{s} \to \mathsf{t}$ in H. They are the reverse implications of the function order axioms. An H-hos A is called **extensional** if it satisfies the extensionality axioms for H. Equivalently, A is extensional if all functions $[\![\_]\!]^A$ are order-embeddings. In particular we have that

$$[\![f]\!]^A \leq^A [\![g]\!]^A \Rightarrow f \leq^A g\ .$$

Then the carriers of functional types are isomorphic to subsets of the respective function spaces. Since we allow subsets rather than always full function spaces, we are working in the framework of "general" models (cf. [18]) rather than "standard" or "full" models of higher type logic. This is the reason why we can still obtain a complete deductive calculus. Moreover, it allows a notion of term-generatedness for extensional hos, which in general would not be possible for full models, since for a countable language the Herbrand structure and therefore all term-generated models have countable carriers, whereas full funcion spaces over infinite domains are uncountable.

A first example of an extensional structure is given by

Lemma 20. *For a given hol H the Herbrand structure WH is extensional.*

Although the extensionality axioms are not Horn formulas, for a particular H-hos A we can replace them by equivalent infinite Horn formulas. To this end we extend the language H by an S-indexed set C of new constants, where $C_{\mathsf{t}} \stackrel{\mathrm{def}}{=} \{c_x : x \in A_{\mathsf{t}}\}$.Denote the extended language by $H(C)$. Then A is made into an $H(C)$-structure by defining $c_x^A \stackrel{\mathrm{def}}{=} x$. This means that each element of

the carrier of A is now denoted by a constant. Then the extensionality axiom for type $\mathsf{s} \to \mathsf{t} \in S$ can be replaced by the (in general infinite) Horn formula

$$\bigwedge_{c \in C_\mathsf{s}} f\,c \leq g\,c \;\Rightarrow\; f \leq g\ .$$

This observation is important in view of the results of [20, 42, 43]; it is the deeper reason for the nice structural properties of extensional specifications. If A is even term-generated, we can avoid the language extension and instead replace the extensionality axiom by the (again in general infinite) Horn formula

$$\bigwedge_{u \in WH_\mathsf{s}} f\,u \leq g\,u \;\Rightarrow\; f \leq g\ .$$

We call a congruence $R \in \mathrm{HCG}(A)$ **extensional** if it satisfies the extensionality axioms, i.e., if for every function type $\mathsf{s} \to \mathsf{t} \in S$ and all $f, g \in A_{\mathsf{s}\to\mathsf{t}}$ we have

$$[\![f]\!] \leq^R [\![g]\!] \;\Rightarrow\; f \leq^R g\ .$$

The set of extensional congruences on A is denoted by Ext^A .

From Lemma 3 and the above remark on Horn formulations of extensionality we have

Corollary 21. *Consider $A \in \mathrm{HOS}(H)$.*

1. *If $R \in \mathrm{HCG}(A)$ then A/R is extensional iff R is extensional.*
2. *Ext^A is lattice-forming.*

For $A \in \mathrm{HOS}(H)$ therefore $\cap\mathrm{Ext}^A$ is the least extensional congruence on A; it forms the "closest" extensional quotient of A. We now want to study the function $\mathrm{Ext} : A \mapsto A/\cap\,\mathrm{Ext}^A$ more thoroughly. First we note that Ext generally not only influences the carriers of types of higher order, but also the basic carriers.

Example 2. Consider a hol H with the basic types m and n and the constants

$$\begin{array}{ll} a & : \mathsf{m} \\ d, e & : \mathsf{n} \\ f, g & : \mathsf{m} \to \mathsf{m} \\ p & : (\mathsf{m} \to \mathsf{m}) \to \mathsf{n} \end{array}$$

and $A \in \mathrm{HOS}(H)$ with

$$\begin{array}{l} f(a)^A = g(a)^A = a^A \\ d^A \neq e^A \\ p(f)^A = d^A \\ p(g)^A = e^A \end{array}$$

Then in $B \stackrel{\mathrm{def}}{=} \mathrm{Ext}(A)$ we have $f^B = g^B$ and hence, by the congruence property of $\cap\mathrm{Ext}^A$, also $d^B = e^B$. □

In general, this is an undesired effect. Thus we call $A \in \mathrm{HOS}(H)$ **extensionally well-behaved** if $\leq_{\mathsf{t}}^{\cap\mathrm{Ext}^A} = \leq_{\mathsf{t}}^A$ for all basic types t of H. This means that the extensional view of A does not induce additional order relations or equalities on the basic elements of A.

Lemma 22. *Assume $A \in \mathrm{HOS}(H)$ for $H = (S, F, P, \phi, \psi)$.*

1. *If A is extensionally well-behaved then $\cap\mathrm{Ext}^A$ is inductively given by*
   (a) $\leq_{\mathsf{t}}^{\cap\mathrm{Ext}^A} = \leq_{\mathsf{t}}^A$ for $\mathsf{t} \in B(H)$.
   (b) For $f, g \in A_{\mathsf{s}\to\mathsf{t}}$ we have $f \leq_{\mathsf{s}\to\mathsf{t}}^{\cap\mathrm{Ext}^A} g$ iff $f =^A g$ or for all $a, b \in A_{\mathsf{s}}$ with $a =_{\mathsf{s}}^{\cap\mathrm{Ext}^A} b$ also $[\![f]\!](a) \leq_{\mathsf{t}}^{\cap\mathrm{Ext}^A} [\![g]\!](b)$.
2. *Let A be continuous and R be an extensional congruence on A that is lub-compatible on the basic carriers. Then R is lub-compatible. In particular, if A is extensionally well-behaved then $\cap\mathrm{Ext}^A$ is lub-compatible.*

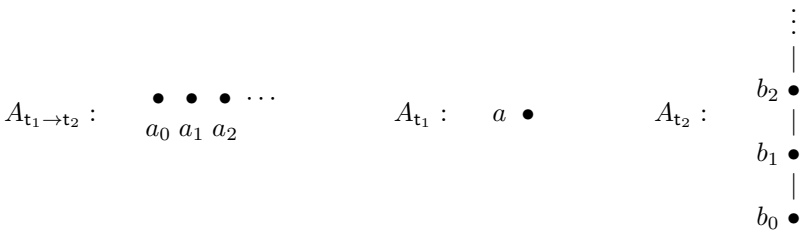
Proof. 1. Define a system R of relations on A inductively by
(a) $\leq_{\mathsf{t}}^R \stackrel{\mathrm{def}}{=} \leq_{\mathsf{t}}^A$ for $\mathsf{t} \in B(H)$,
(b) $f \leq_{\mathsf{s}\to\mathsf{t}}^R g$ iff $f =^A g$ or for all $a, b \in A_{\mathsf{s}}$ with $a =_{\mathsf{s}}^R b$ also $[\![f]\!](a) \leq_{\mathsf{t}}^R [\![g]\!](b)$, and $a =_{\mathsf{t}}^R b \stackrel{\mathrm{def}}{\Leftrightarrow} a \leq_{\mathsf{t}}^R b \wedge b \leq_{\mathsf{t}}^R a$. To prove 1. we show now that $R = \cap\mathrm{Ext}^A$. To this end we first prove that R is a congruence on R.
By construction $=^R$ and $\leq^R$ are reflexive and $=^R$ is symmetric and substitutive w.r.t. $\leq^R$. Next, we show that $=^R$ is also substitutive w.r.t. the application operations α. Assume $f =_{\mathsf{s}\to\mathsf{t}}^R g$ and $a =_{\mathsf{s}}^R b$. By definition of $=^R$ then $f \leq_{\mathsf{s}\to\mathsf{t}}^R g$ and $g \leq_{\mathsf{s}\to\mathsf{t}}^R f$. By definition of $\leq_{\mathsf{s}\to\mathsf{t}}^R$ and symmetry of $=_{\mathsf{s}}^R$ we therefore have $[\![f]\!](a) \leq_{\mathsf{t}}^R [\![g]\!](b)$ and $[\![f]\!](b) \leq_{\mathsf{t}}^R [\![g]\!](a)$, i.e., $[\![f]\!](a) =_{\mathsf{t}}^R [\![g]\!](b)$.
We now show transitivity of $\leq_{\mathsf{t}}^R$ by induction on t. For $\mathsf{t} \in B(H)$ this is clear from the assumption. Assume $f \leq_{\mathsf{s}\to\mathsf{t}}^R g \wedge g \leq_{\mathsf{s}\to\mathsf{t}}^R h$ for $f, g, h \in A_{\mathsf{s}\to\mathsf{t}}$. If $f =_{\mathsf{s}\to\mathsf{t}}^A g$ or $g =_{\mathsf{s}\to\mathsf{t}}^A h$ then $f \leq_{\mathsf{s}\to\mathsf{t}}^R h$ is immediate. Otherwise consider arbitrary $a, b \in A_{\mathsf{s}}$ with $a =_{\mathsf{s}}^R b$. By definition of $\leq_{\mathsf{s}\to\mathsf{t}}^R$ and reflexivity of $=_{\mathsf{s}}^R$ we get $[\![f]\!](a) \leq_{\mathsf{t}}^R [\![g]\!](a)$ and $[\![g]\!](a) \leq_{\mathsf{t}}^R [\![h]\!](b)$. Since by the induction hypothesis $\leq_{\mathsf{t}}^R$ is transitive this implies $[\![f]\!](a) \leq_{\mathsf{t}}^R [\![h]\!](b)$. Since a, b were arbitrary we conclude from the definition of $\leq_{\mathsf{s}\to\mathsf{t}}^R$ also $f \leq_{\mathsf{s}\to\mathsf{t}}^R h$.
Hence $\leq^R$ is a preorder and thus, by Lemma 8, $=^R$ is an equivalence.
By construction, R satisfies the function order axioms. We next show that $\leq_{\mathsf{t}}^A \subseteq \leq_{\mathsf{t}}^R$ by induction on t. For the basic types this is trivial. Assume now $f \leq_{\mathsf{s}\to\mathsf{t}}^A g$ and $a =_{\mathsf{s}}^R b$ for $a, b \in A_{\mathsf{s}}$. By substitutivity and reflexivity of $=^R$ we get $[\![f]\!](a) \leq_{\mathsf{t}}^R [\![f]\!](b)$. Since A satisfies the function order axioms we have $[\![f]\!](b) \leq_{\mathsf{t}}^A [\![g]\!](b)$. By the induction hypothesis this implies $[\![f]\!](b) \leq_{\mathsf{t}}^R [\![g]\!](b)$, and transitivity of $\leq_{\mathsf{t}}^R$ shows $[\![f]\!](a) \leq_{\mathsf{t}}^R [\![g]\!](b)$. Since a, b were arbitrary we conclude $f \leq_{\mathsf{s}\to\mathsf{t}}^R g$.
Since substitutivity was shown above, R thus is a congruence on A.
Next we show that R is extensional. Suppose $[\![f]\!](c) \leq_{\mathsf{t}}^R [\![g]\!](c)$ for all $c \in A_{\mathsf{s}}$. By substitutivity and reflexivity of $=^R$ we have for arbitrary $a, b \in A_{\mathsf{s}}$ with $a =_{\mathsf{s}}^R b$ that $[\![f]\!](a) =_{\mathsf{t}}^R [\![f]\!](b) \leq_{\mathsf{t}}^R [\![g]\!](b) =_{\mathsf{t}}^R [\![g]\!](b)$. Since a, b were arbitrary we conclude $f \leq_{\mathsf{s}\to\mathsf{t}}^R g$.

Let now R' be another extensional congruence on A. We show by induction on t that $\leq^R_\mathsf{t} \subseteq \leq^{R'}_\mathsf{t}$ for all $\mathsf{t} \in S$.
   (a) For $\mathsf{t} \in B(H)$ we have $\leq^R_\mathsf{t} = \leq^A_\mathsf{t} \subseteq \leq^{R'}_\mathsf{t}$.
   (b) Let $f \leq^R_{\mathsf{s}\to\mathsf{t}} g$. We have for all $a \in A_\mathsf{s}$ that $[\![f]\!](a) \leq^R_\mathsf{t} [\![g]\!](a)$, and hence, by the induction hypothesis, also $[\![f]\!](a) \leq^{R'}_\mathsf{t} [\![g]\!](a)$ showing $f \leq^{R'}_{\mathsf{s}\to\mathsf{t}} g$ by extensionality of R'.
2. The proof again uses induction on the types involved. On the basic types the lub-compatibility of R is assumed. Let now $D \subseteq A_{\mathsf{s}\to\mathsf{t}}$ be $\leq^A_{\mathsf{s}\to\mathsf{t}}$-directed. For all $a \in A_\mathsf{s}$ we have, by $\leq^A$-continuity of $[\![\_]\!]$, that $[\![\mathsf{lub}_{\leq^A_{\mathsf{s}\to\mathsf{t}}} D]\!](a) = \mathsf{lub}_{\leq^A_\mathsf{t}} [\![D]\!](a)$. By the induction hypothesis $\mathsf{lub}_{\leq^A_\mathsf{t}} [\![D]\!](a) \leq^R_\mathsf{t} \mathsf{upb}_{\leq^R_\mathsf{t}} [\![D]\!](a)$. Since R satisfies the function order axioms, $[\![\_]\!]$ is $\leq^R$-monotonic and so also $[\![\mathsf{lub}_{\leq^A_{\mathsf{s}\to\mathsf{t}}} D]\!](a) \leq^R_\mathsf{t} [\![\mathsf{upb}_{\leq^R_{\mathsf{s}\to\mathsf{t}}} D]\!](a)$. Now extensionality of R shows $\mathsf{lub}_{\leq^A_{\mathsf{s}\to\mathsf{t}}} D \leq^R_{\mathsf{s}\to\mathsf{t}} \mathsf{upb}_{\leq^R_{\mathsf{s}\to\mathsf{t}}} D$.

□

Property 2. will be instrumental for the existence of extensional structures that allow solving fixpoint equations. However, the structure $\mathrm{Ext}(A)$ need not be continuous even if A is:

Example 3. Let $A_{\mathsf{t}_1}, A_{\mathsf{t}_2}$ and $A_{\mathsf{t}_1\to\mathsf{t}_2}$ be given by



and define $[\![a_i]\!]^A(a) \stackrel{\text{def}}{=} b_i$. Then $A_{\mathsf{t}_1\to\mathsf{t}_2}$ is Δ-complete, but $\mathrm{Ext}(A)_{\mathsf{t}_1\to\mathsf{t}_2}$ is order-isomorphic to A_{t_2} which is not Δ-complete. □

4.3 Higher-Order and Extensional Specifications

A **higher-order specification** is an ordered specification $K = (H, E)$ with a higher-order signature $H = (S, F, P, \phi, \psi)$ such that the set E of axioms includes the function order axioms. K is **extensional** if E also includes the extensionality axioms. It is clear that all models of an extensional higher-order specification are extensional. We use the keyword extensional to express that the respective specification implicitly contains the extensionality axioms.

Consider now specifications that are **almost Horn**, i.e., higher-order specifications $K = (H, E)$ such that E consists of Horn axioms except possibly for extensionality axioms. Note that an almost Horn specification may even be Horn. As an immediate consequence of Theorem 2 and Horn expressibility of extensionality on term-generated algebras we obtain

Theorem 23. *For a higher-order specification K that is*
- almost Horn,
- almost Horn and grounded,
- almost Horn and monotonic,
- almost Horn, monotonic, and grounded,
the set $\mathrm{GEN}(K)/\preceq$ *of isomorphism classes forms a complete lattice. In particular, K has an initial term-generated model.*

For a specification with an extensionality axiom for a type $\mathtt{s} \to \mathtt{t} \in S$ the following additional inference rule is sound:

(Extensionality) $\dfrac{u\,x \leq w\,x}{u \leq w}$ for $u, w \in WH(X)_{\mathtt{s}\to\mathtt{t}}$ and $x \in X_{\mathtt{s}}$ not occurring in u, w

As shown in [30] and generalising the respective result in [25] we have the following completeness result:

Theorem 24. *Let $K = (H, E)$ be an almost Horn specification and $u, w \in WH(X)$. Then*

$$K \vdash u \leq w \text{ iff } \mathrm{MOD}(K) \models u \leq w.$$

Note that the analogue of Theorem 10 fails for almost Horn specifications.

4.4 Particular Higher-order Specifications

So far we have not used λ notation. This decision was taken to avoid the usual problems with bound variables. This does not imply a loss in power: A term $\lambda x.t$ can always be simulated by a new constant f with the axiom $f\,x = t$. Another way to realise λ notation is to include the classical combinators S, K, I as additional constants with the usual axioms and to replace λ terms by their combinator equivalents.

In the case of a higher-order specification $K = (H, E)$, we call K **grounded** if E comprises the groundedness axioms and all axioms of the form $\bot(x) = \bot$. Strictness is expressed as in the first-order case: for a hol $H = (S, F, P, \phi, \psi)$ and $f \in F$ with $\phi(f) = \mathtt{t}_1 \to \cdots \to \mathtt{t}_{k-1} \to \mathtt{t}_k \to \mathtt{t}_{k+1} \to \cdots \to \mathtt{t}_n \to \mathtt{t}$, strictness in the k-th argument is required by the axiom

$$f\,x_{\mathtt{t}_1}\,\cdots\,x_{\mathtt{t}_{k-1}}\,\bot_{\mathtt{t}_k}\,x_{\mathtt{t}_{k+1}}\,\cdots\,x_{\mathtt{t}_n} \leq \bot_{\mathtt{t}}.$$

4.5 Continuous Higher-Order Structures

Since hoss are just a special case of structures, we can apply completion techniques to monotonic hoss as well. However, the ideal completion may destroy extensionality as is shown by

Example 4. Let $\mathbb{N}_\bot$ be the flat domain of natural numbers extended by $\bot$ and define the functions $f_i : \mathbb{N}_\bot \to \mathbb{N}_\bot$ $(i \in \mathbb{N} \cup \{\infty\})$ by

$$f_i(x) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } 0 \leq x < i \\ \bot & \text{otherwise} \end{cases}$$
$$f_\infty(x) \stackrel{\text{def}}{=} x\ .$$

The set $M \stackrel{\text{def}}{=} \{f_i : i \in \mathbb{N} \cup \{\infty\}\}$ is extensional. The ideals of M are the sets $I_i \stackrel{\text{def}}{=} \{f_j : j \leq i\}$ $(i \in \mathbb{N} \cup \{\infty\})$ and $J \stackrel{\text{def}}{=} \{f_j : j < \infty\}$. We have $I_i \subseteq I_k$ iff $i \leq k$, $I_i \subseteq J$ iff $i < \infty$, and $J \subseteq I_\infty$. In order to extend function application in a continuous way to $(I(M), \subseteq)$ we need to set for $I \in I(M)$

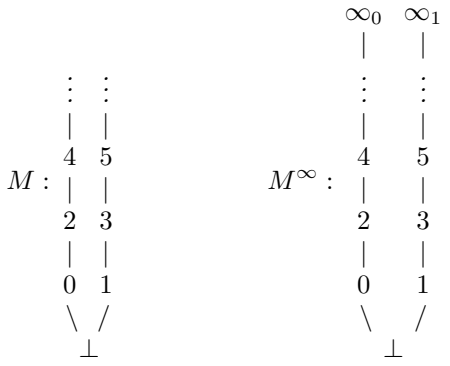
$$I(x) \stackrel{\text{def}}{=} \mathsf{lub}\,\{f(x) : f \in I\}\ .$$

We have for all $x \in \mathbb{N}_\perp$ that $I_\infty(x) = J(x)$, but $I_\infty \neq J$. Hence $I(M)$ is not extensional. □

This example shows also that the embedding of M into $I(M)$, in general, is not continuous; hence the ideal completion is not a lub-preserving completion.

It may seem that the loss of extensionality is due to the additional limit point, i.e., to the fact that the ideal completion is not lub-preserving. However, even under a lub-preserving completion extensionality may be lost. In fact, it turns out that there is *no* completion technique which is guaranteed to preserve extensionality. To prove this formally, we first study a particular ordered set:

Lemma 25. *Consider the following ordered set M and its completion M^∞:*



Then in every completion Q of M we have

$$\mathsf{lub}_Q\,\{\iota(n) : n \text{ even}\} \neq \mathsf{lub}_Q\,\{\iota(n) : n \text{ odd}\}$$

where $\iota : M \to Q$ is the embedding of M into Q.

Proof. Consider the embedding $h : M \to M^\infty$. Since Q is a completion of M and M^∞ is Δ-complete, h extends uniquely into a continuous function $\hat{h} : Q \to M^\infty$. We calculate, for $k \in \{0, 1\}$

$$\begin{aligned}
& \hat{h}(\mathsf{lub}_Q\,\{\iota(2i+k) : i \in \mathbb{N}\}) \\
= \quad & \{\![\ \text{continuity}\ ]\!\} \\
& \mathsf{lub}_{M^\infty}\,\{\hat{h}(\iota(2i+k)) : i \in \mathbb{N}\} \\
= \quad & \{\![\ Q \text{ completion}\ ]\!\}
\end{aligned}$$

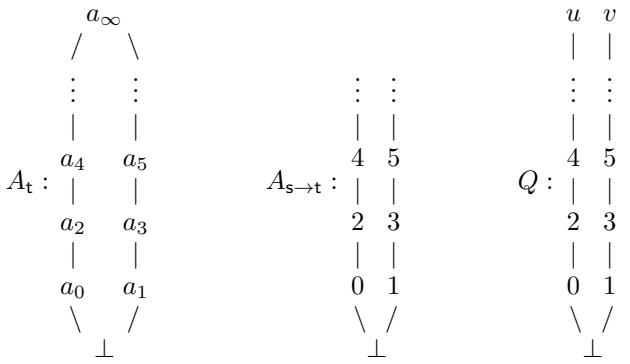
$$
\begin{aligned}
& \mathsf{lub}_{M^\infty}\,\{h(2i+k) : i \in \mathbb{N}\} \\
= \quad & \{\!\![\ \text{ideal completion}\ ]\!\!\} \\
& \infty_k\ .
\end{aligned}
$$

Since $\infty_0 \neq \infty_1$ and $\hat{h}$ is a function the claim follows. □

Now we can show

Theorem 26. *There is no completion technique which is guaranteed to preserve extensionality.*

Proof. Consider a hos A and types s, t with $A_\mathsf{s} = \{\bot\}$ and A_t and $A_{\mathsf{s}\to\mathsf{t}}$ given by the following diagrams.



Assume further that $[\![\bot]\!]^A(\bot) = \bot$ and $[\![i]\!]^A(\bot) = a_i$. Then $A_{\mathsf{s}\to\mathsf{t}}$ satisfies the extensionality axiom. By the previous lemma we know that in every completion A^∞ of A the carrier $A^\infty_{\mathsf{s}\to\mathsf{t}}$ has a subset of the shape of Q. For the unique continuous extension $[\![\_]\!]^{A^\infty}$ of $[\![\_]\!]^A$ we have $[\![u]\!]^{A^\infty}(\bot) = a_\infty = [\![u]\!]^{A^\infty}$ although $u \neq v$. Hence A^∞ is not extensional. □

A remedy for this unpleasant situation will be discussed in the following section.

4.6 Higher-Order Fixpoint Structures

We now want to apply the results of Section 3.7 to the case of higher-order structures. Given a hol H and a grounded structure $A \in \mathrm{GEN}(H)$, we can carry out the following construction:

1. Pass to the monotonic structure $B \stackrel{\text{def}}{=} \mathrm{Mon}(A)$.
2. Form the ideal completion $C \stackrel{\text{def}}{=} B^\infty$.
3. If $\cap\mathrm{Ext}^C$ is lub-compatible on the basic types, pass to the extensional structure $D \stackrel{\text{def}}{=} \mathrm{Ext}(C)$.

By Lemma 22 and Theorem 18.3 then D is an extensional fixpoint structure.

To conclude this section, we want to discuss the aspect of generatedness for this construction. Consider a term-generated structure $A \in \mathrm{GEN}(H)$. In the ideal completion A^∞ every carrier element is the least upper bound of a directed set of (injections of) elements of A, i.e., of term-denotable elements, and the behaviour of A^∞ is, through continuity, determined by this term-denotable set. Hence in the ideal completion A^∞ the infinite entities may just be viewed as a "way of speaking" about certain sets of finitely denotable elements. Note that this behaviour of A^∞ carries over to quotients by lub-compatible congruences. Thus, unlike in the approaches of [44, 1], no transfinite terms are involved here. This seems intuitively much more appealing, in particular, since we want to interpret the finitely denotable elements as approximations through which we handle infinite elements, as is the case on actual computers.

4.7 Continuous and Fixpoint Models of Higher-Order Specifications

In this section we want to construct **fixpoint models** of specifications, i.e., models that are fixpoint structures. In keeping with the idea of approximability, we want to use the construction of the previous section. To ensure that, starting from a model, this construction leads again to a model, the axioms of the specifications should remain valid in the completed structure.

From [28] we know

Lemma 27. *Consider an ordered and grounded language L and a formula Φ of the shape $\Phi = (u_1 \leq u_2)$ or $\Phi = (u_1 = u_2)$. Then for any grounded and monotonic ordered structure $A \in \mathrm{STR}(L)$ we have $A \models \Phi$ iff $A^\infty \models \Phi$.*

Together with the principle of term-induction this gives a very powerful means of proving properties of completions of term-generated models of an inequational specification. Further classes of axioms that are preserved under the ideal completion are discussed in [28]. As we shall, however, see below, more general axioms are not useful in the higher-order case.

Example 5. Consider the specification S from Example 1. The ideal completion I^∞ of the initial model I of S has a carrier which is order-isomorphic to the set M^∞ in Lemma 25. □

For the inequational specification with grounded language L and groundedness and monotonicity axioms only (cf. Section 3.4) we get as a continuous model the structure $WL^\infty \stackrel{\mathrm{def}}{=} (WL/\sqsubseteq)^\infty$. It can be thought of as the completion of the term structure WL by "infinite terms". For a grounded hol H that is essentially of first order, it is isomorphic to the free Δ-complete F-magma defined in [10]. WL^∞ can thus be interpreted as a generalised term structure comprising finite and infinite L-terms. In the case of a general hol we may also have infinite terms of functional type.

Finally, we need to consider the case where a quotient has to be taken after completion. To this end, the class of models of the respective specification should

be closed under quotients. This is the case for inequational specifications, as we know from Theorem 10.1. Hence we have

Corollary 28. *Let K be a grounded and monotonic inequational specification and assume $A \in \mathrm{GEN}(\mathrm{Mon}(K))$. If $\cap\mathrm{Ext}^{A^\infty}$ is lub-compatible on the basic types then $\mathrm{Ext}(A^\infty)$ is an extensional fixpoint model of K.*

5 Examples

In this section we give a number of examples illustrating various uses of higher-order specifications. For modularisation, we use the phrase include SPC′ within a specification SPC to indicate that SPC′ is a subspecification of SPC.

On some occasions we use overloading of operator identifiers. Also, we freely use mixfix notation; the positions of actual parameters are marked by underscores.

We assume grounded and monotonic specifications BOOL and NAT that define the basic type bool with the truth values *true* and *false* and the standard operations on them, and the basic type nat with 0, successor *succ*, predecessor *pred*, zero test *iszero*, addition *add*, equality test *eq* and the doubling function *double*. All these operators are supposed to be strict.

Finally, for every populated type s we assume a constant

$$\mathsf{if}\ \_\,\mathsf{then}\ \_\,\mathsf{else}\ \_\,\mathsf{fi} : \mathsf{bool} \to \mathsf{s} \to \mathsf{s} \to \mathsf{s}$$

with the axioms

$$\begin{aligned}&\mathsf{if}\ \bot\,\mathsf{then}\ a\,\mathsf{else}\ b\ \mathsf{fi} = \bot,\\ &\mathsf{if}\ \mathit{true}\,\mathsf{then}\ a\,\mathsf{else}\ b\ \mathsf{fi} = a,\\ &\mathsf{if}\ \mathit{false}\,\mathsf{then}\ a\,\mathsf{else}\ b\ \mathsf{fi} = b.\end{aligned}$$

5.1 A Small Functional Programming Language

In this example we specify a small language showing some of the essential features of Backus's FP (cf. [2]), viz. constant function $\underline{\ }$, lifting $\hat{\ }$ of basic functions and predicates, function composition $\_ \circ \_$, conditional $\_ \to \_;\_$ and function application $\_ : \_$ We restrict ourselves to unary functions on natural numbers and assume a superspecification OPS of NAT that provides the primitive functions and predicates from which to build the functional programs.

spec FUNCT
grounded monotonic
include OBS
basic type funct, bfunct
$\underline{\ }$: nat → funct
$\hat{\ }$: (bool → nat) → bfunct
$\hat{\ }$: (nat → nat) → funct
$\_ \circ \_$: funct → funct → funct

$\_ \to \_;\_ : \mathsf{bfunct} \to \mathsf{funct} \to \mathsf{funct} \to \mathsf{funct}$
$\_ : \_ : \mathsf{bfunct} \to \mathsf{nat} \to \mathsf{bool}$
$\_ : \_ : \mathsf{funct} \to \mathsf{nat} \to \mathsf{nat}$
axioms $\bot : x = \bot$
$\overline{y} : x = y$
$\hat{p} : x = p\,x$
$\hat{f} : x = f\,x$
$(f \circ g) : x = f : (g : x)$
$(p \to f\,; g) : x = \mathsf{if}\ p : x\ \mathsf{then}\ f : x\ \mathsf{else}\ g : x\ \mathsf{fi}$
endspec

The only higher-order objects are the combining forms $=$, $\hat{\_}$, $\_ \circ \_$ and $\_ \to \_;\_$ whereas all functional programs are basic objects. Therefore questions of extensionality do not arise.

Unlike the first-order approaches in [9, 27], the present framework allows a proper treatment of the lifting operators $\hat{\_}$ that assign to every operation of type $\mathsf{nat} \to \mathsf{bool}$ and $\mathsf{nat} \to \mathsf{nat}$ a functional program denoting that operation. In the approaches mentioned, this had to be done *outside* the respective specifications by adding for each operator of type $\mathsf{s}_1 \to \mathsf{s}_2$ a new constant of sort $\mathsf{funct}\ \mathsf{s}_1\ \mathsf{s}_2$.

The specification is inequational and hence our completion techniques apply. A typical recursion equation over FUNCT is

$$\mathsf{funct}\ \mathit{zero} = \hat{\mathit{iszero}} \to \overline{0}\,; (\mathit{zero} \circ \hat{\mathit{pred}})$$

defining the recursive constant zero function.

5.2 Function Composition

As a building block for further specifications we define a generic composition operator for functions. Note that this is quite different from the purely syntactic composition operator in the functional language of the previous example. This specification by itself does not have a hol in the sense defined in Section 4.1. However, we only want to use it as a building block for larger specifications which then will have proper hols.

spec COMP
$\_ \circ \_ : (\mathsf{t}_1 \to \mathsf{t}_2) \to (\mathsf{t}_2 \to \mathsf{t}_3) \to (\mathsf{t}_1 \to \mathsf{t}_3)$
axioms $(f \circ g)\,x = f\,(g\,x)$
endspec

We have

$$\mathrm{COMP} \vdash (f \circ (g \circ h))\,x = ((f \circ g) \circ h)\,x$$

with variables f, g, h, and x. In extensional models A of specifications that include COMP we have therefore also

$$A \models f \circ (g \circ h) = (f \circ g) \circ h,$$

i.e., composition is associative in A.

5.3 Fixpoints and Function Iteration

We now define a fixpoint operation as used in the semantics of recursively routines or objects; contrary to Section 3.7, we do this *within* rather than *over* a specification. We compare the fixpoint with the elements obtained by functional iteration starting from $\bot$.

spec FIX
grounded
include NAT
include COMP
$\mathit{fix} : (\mathsf{t} \to \mathsf{t}) \to \mathsf{t}$
$\mathit{id} : \mathsf{t} \to \mathsf{t}$
$\_^{-} : (\mathsf{t} \to \mathsf{t}) \to \mathsf{nat} \to (\mathsf{t} \to \mathsf{t})$
axioms $\mathit{fix}\, f = f\,(\mathit{fix}\, f)$
$x = f\, x \Rightarrow \mathit{fix}\, f \leq x$
$\mathit{id}\, x = x$
$f^0 = \mathit{id}$
$f^{\mathit{succ}\, n} = f \circ f^n$
endspec

Note that the specification of *fix* is not complete relative to the other functions. Therefore, in an initial model of FIX, the interpretation of $\mathit{fix}(f)$ will not coincide with that of any other term; it provides a proper extension of the carrier of type $\mathsf{t} \to \mathsf{t}$ (cf. [16]).

Using the proof rules from Section 2.5 we can deduce $f^i \bot \leq \mathit{fix}\, f$. Hence $\mathit{fix}\, f$ is an upper bound of the $f^i \bot$. However, since we have no assertion of continuity (in fact not even a notion of continuity at this level), we cannot be sure that it is the least upper bound of the $f^i \bot$. Note that this property cannot even be specified by a Horn axiom.

This unsatisfactory behaviour is the reason why we consider recursion equations *over* specifications.

5.4 Communicating Sequential Processes

In [19] already an algebraic approach to the specification of communicating sequential processes is taken. We want to show with this Example, how that description fits into the present framework, and how higher-order concepts can be used to smoothen the description.

The ordering on processes used in the specification is the refinement ordering: It is intended that $p \leq q$ should hold iff q is a refinement of p, i.e., iff the behaviour of q is at least as determinate and defined as that of p. In the terminology of [19], $p \leq q$ states that q is at least as reliable as p. Actually, we define the ordering conversely to the one used in [19] to make it fit in with the ideal completion.

We assume a primitive specification DATA that defines the basic type data of data values to be used in the communications. Based on DATA, we want to

build our description of CSP in several levels to bring out the structure of that language and its semantics more clearly. First we define the totally unreliable and underspecified process $\top$ that is refined by any process.

Moreover, we define the output construct $!d \to p$ that describes a process that outputs the element d of type data and then behaves like the process p.

The sort of processes is denoted by pcs. The corresponding specification reads

spec TRACES
monotonic
include DATA
basic type pcs
$\top$: pcs
$!\_ \to \_$: data $\to$ pcs $\to$ pcs
axioms $\top \leq p$
endspec

The only terms of type pcs in TRACES are of the form

$$!d_1 \to (!d_2 \to \cdots (!d_k \to \top) \cdots)$$

with primitive terms d_i of type data. Since the only axioms besides leastness of $\top$ are monotonicity axioms, in an initial model I of TRACES we have

$$!d_1 \to (!d_2 \to \cdots (!d_m \to \top) \cdots)^I \leq^I !e_1 \to (!e_2 \to \cdots (!e_n \to \top) \cdots)^I$$

iff $m \leq n$ and for all $i \leq m$ then $d_i^I = e_i^I$. Thus the carrier pcs^I is order-isomorphic to the set of all finite sequences of data elements under the prefix ordering. Hence, for the completion I^∞, pcs^{I^∞} is order-isomorphic to the set of all finite and infinite sequences of data elements, again under the prefix ordering. In the terminology of CSP, these sequences are also called traces, whence the name of our specification. More frequently, these sequences are also called streams.

In the next step, we add the operator $\_ \vee \_$ of disjunction of processes. It is used to form processes that offer alternative behaviour. The operator $\_ \vee \_$ is associative, commutative, and idempotent; moreover, a disjunction is at most as reliable as either of its alternatives.

spec TRACESETS
monotonic
include TRACES
$\_ \vee \_$: pcs $\to$ pcs $\to$ pcs
axioms $(p \vee q) \vee r = p \vee (q \vee r)$
$p \vee q = q \vee p$
$p \vee p = p$
$p \vee q \leq p$
$(!d \to p) \vee (!d \to q) \leq !d \to (p \vee q)$
endspec

The last axiom, together with the derivable property

$$!d \to (p \vee q) \leq (!d \to p) \vee (!d \to q)$$

implies

$$!d \to (p \vee q) = (!d \to p) \vee (!d \to q),$$

i.e., that output distributes over disjunction, as is stipulated in [19]. This property allows us to write every process as a disjunction of traces.

Another derivable property is

$$p \leq q \Leftrightarrow p \vee q = p\ . \tag{1}$$

In [19] this property is taken as the definition of the refinement ordering, since there the ordering is a derived concept, whereas in the present framework it is, of course, a basic notion. This property allows omitting extensions of traces in a disjunction from that disjunction without changing the result. Together with the associativity, commutativity, and idempotence of choice this implies that the carrier pcs^I in an initial model of TRACESETS is isomorphic to the set of finite sets of traces that are not prefixes of each other. Hence in the completion I^∞ the carrier pcs^{I^∞} is isomorphic to the Smyth power domain (cf. [41]) over the set of finite and infinite traces.

From (1) one obtains immediately

$$\top \vee q = \top\ , \tag{2}$$

which states that disjunction is "demonic", i.e., that a process offering a totally unreliable alternative is totally unreliable itself. Note that (2) depends crucially on the axiom $p \vee q \leq p$. By using the dual axiom $p \leq p \vee q$ we would obtain an "angelic" disjunction with $\top \vee p = p$.

One can also show that the disjunction of two processes is the greatest lower bound of these processes w.r.t. the refinement ordering; hence in all models of TRACESET the carrier of sort pcs is a lower semilattice.

In the next step we introduce an input command. In CSP this is coupled with a binding mechanism: The input value is bound to an identifier under which it is available throughout the remaining process. However, to keep the description simple, we want to avoid the introduction of identifiers and the problems of binding. Rather, we use a combinator variant of input and represent a process depending on a free identifier for data elements as a function from data to pcs. The input command may only be applied to such a parameterised process pp; after input of a data value d the process then behaves like $pp(d)$. We extend disjunction pointwise to parameterised processes. The operator ©. lifts a process to a constant parameterised process.

```
spec INPUT
    monotonic
    include TRACESETS
    ? → _ : (data → pcs) → pcs
```

$\_ \vee \_ : (\mathsf{data} \to \mathsf{pcs}) \to (\mathsf{data} \to \mathsf{pcs}) \to (\mathsf{data} \to \mathsf{pcs})$
$\copyright\_ : \mathsf{pcs} \to (\mathsf{data} \to \mathsf{pcs})$
axioms $? \to (pp \vee qq) = (? \to pp) \vee (? \to qq)$
$(pp \vee qq)(d) = pp(d) \vee qq(d)$
$(\copyright p)(d) = p$
endspec

Let us now specify a process representing an unbounded buffer. In this, we follow [7]. Since a buffer process internally maintains the sequence of buffered values, we first specify sequences of data values.

[] denotes the empty sequence, [_] is the singleton sequence former, and $\_ + \_$ denotes concatenation.

spec SEQU =
include DATA
basic type sequ
[] : sequ
[_] : data → sequ
$\_ + \_$: sequ → sequ → sequ
axioms $[] + s = s$
$s + [] = s$
$r + (s + t) = (r + s) + t$
endspec

spec BUFFER
monotonic
include SEQU
$buf\_ : \mathsf{sequ} \to \mathsf{pcs}$
$enterbuf\_ : \mathsf{sequ} \to (\mathsf{data} \to \mathsf{pcs})$
axioms $buf_{[]} = ? \to enterbuf_{[]}$,
$buf_{s+[x]} = (? \to enterbuf_{s+[x]}) \vee (!x \to buf_s)$
$enterbuf_t\, y = buf_{[y]+t}$
endspec

These axioms may be interpreted as follows: A buffer with empty internal sequence can only input and then behave like the enterbuf process with empty internal sequence. A buffer with non-empty internal sequence either may input another value and pass it to the enterbuf process or output the last buffer value and behave like a buffer with a shortened internal sequence. The enterbuf process merely attaches the input value to the front of its internal sequence and then behaves like a buffer. This specification shows how, using higher-order concepts, we can give the use of indices in [7] a precise foundation. Further CSP constructs such as chaining and interleaving can be introduced similarly following [19].

Since the whole specification is inequational, grounded (with $\top$ instead of $\bot$), and monotonic, the constructions of Sections 4.6 and 4.7 yield fixpoint models of

PCSCOMB in which solutions to these recursion equations exist. As an example for such a recursion equation consider

$$\mathsf{pcs}\ \mathit{dinf}\ = !d \to \mathit{dinf}$$

for some fixed $d \in \mathsf{data}$. Its least solution is the process that emits an infinite number of ds. However, this equation is of first order and so already the techniques of [26, 28] would have sufficed to give its semantics. A higher-order recursion is

$$\mathsf{pcs}\ \mathit{finf}\ = ? \to \lambda x.\ !f\ x \to \mathit{finf}$$

for some fixed $f \in \mathsf{data} \to \mathsf{data}$. This is an infinitary version of the apply-to-all operation from functional languages: an infinite process fed into *finf* by chaining (see [19]) gets all its output transformed by f.

Finally it should be remarked that the description given leads to problems with fairness when infinite processes are considered. However, this is due to the model chosen in [19] and not to the underlying framework of higher-order specification. The present example was meant to show how the semi-formal treatment of [7, 19] can be made fully formal within the proposed specification style; it is not an attempt at giving a new semantics for parallelism.

6 Conclusion and Outlook

The approach to higher-order specifications using a generation principle also for higher types has proved to lead to a framework which preserves the simplicity and clarity of the first-order case. Higher-order specifications can be formulated in a straightforward way; for Horn axioms the existence of extensional models is guaranteed, so that in this case one can use the higher-order framework quite naively and actually conceive of the higher-order objects as functions. We deem this of high importance for the practical acceptance of the idea of formal specification in general and of higher-order specifications in particular. Only if the framework is simple, programmers will actually want to use it.

Although the construction of fixpoint models turns out to be more awkward than in the first-order case, the way how specifications need to be written is still fairly straightforward.

It will be interesting to see whether from Kleene's approximation sequence for the solution of recursion equations over specifications one can derive busy and lazy operational semantics as was done in [27] for the first-order case. Such an investigation should go hand in hand with a search for deduction-oriented sufficient criteria for hierarchy-completeness extending the ones in [17] to the higher-order case. Similarly, deduction-oriented sufficient criteria for relative consistency w.r.t. subspecifications should be developed.

Of course, a large field of further research is the investigation of wider classes of axioms that admit extensional and fixpoint models.

Acknowledgement. This work has been initiated by discussions with my colleagues from the project CIP and with members of IFIP WG 2.1. I am most grateful to F.L. Bauer, M. Broy, W. Dosch, H. Ehler, K. Meinke, F. Nickl, H. Partsch, O. Paukner, A. Poigné, A. Tarlecki, and M. Wirsing for their valuable comments.

References

1. J. Adámek, A.H. Mekler, E. Nelson, J. Reiterman: On the logic of continuous algebras. Notre Dame J. Formal Logic **29**, 365–380 (1988)
2. J. Backus: Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. Commun. ACM **21**, 613-641 (1978)
3. F.L. Bauer, H. Wössner: Algorithmic language and program development. Berlin: Springer 1982
4. F.L. Bauer, R. Berghammer, M. Broy, W. Dosch, F. Geiselbrechtinger, R. Gnatz, E. Hangel, W. Hesse, B. Krieg-Brückner, A. Laut, T. Matzner, B. Möller, F. Nickl, H. Partsch, P. Pepper, K. Samelson, M. Wirsing, H. Wössner: The Munich project CIP. Volume I: The wide spectrum language CIP-L. Lecture Notes in Computer Science **183**. Berlin: Springer 1985
5. G. Birkhoff: Lattice theory, 3rd edition. American Mathematical Society Colloquium Publications, Vol. XXV. Providence, R.I.: AMS 1967
6. S. Bloom: Varieties of ordered algebras. J. Computer Syst. Sci. **13**, 200–212
7. S.D. Brookes, C.A.R. Hoare, A.W. Roscoe: A theory of communicating sequential processes. J. ACM **31**, 560-599 (1984)
8. M. Broy: Partial interpretations of higher order algebraic types. In: M. Broy (ed.): Logic of Programming and Calculi of Discrete Design. NATO ASI Series. Series F: Computer and Systems Sciences, Vol. 36. Berlin: Springer 1987, 185–241
9. M. Broy, M. Wirsing: Initial versus terminal algebra semantics for partially defined abstract types. Institut für Informatik der TU München, TUM-I8018, 1980
10. B. Courcelle, M. Nivat: Algebraic families of interpretations. 17th Annual IEEE Symposium on Foundations of Computer Science, 1976, 137-146
11. B. Courcelle, J.-C. Raoult: Completions of ordered magmas. Fundamenta Informaticae **3**, 105-111 (1980)
12. B.A. Davey, H.A. Priestley: Introduction to lattices and order. Cambridge: Cambridge University Press 1990
13. P. Dybjer: Category-theoretic logics and algebras of programs. Chalmers University of Technology at Göteborg, Dept. of Computer Science, Ph.D. Thesis, 1983
14. H.B. Enderton: A mathematical introduction to logic. New York: Academic Press 1972
15. J.A. Goguen, J. Thatcher, E.G. Wagner, J.B. Wright: Initial algebra semantics and continuous algebras. J. ACM **24**, 68-95 (1977)
16. T. Grünler: Spezifikationen höherer Ordnung. Fakultät für Mathematik und Informatik der Universität Passau, Dissertation 1990
17. J.V. Guttag: The specification and application to programming of abstract data types. PH. D. Thesis, University of Toronto, Department of Computer Science, Rep. CSRG-59, 1975
18. L. Henkin: Completeness in the theory of types. J. Symbolic Logic **15**, 81-91 (9150)

19. C.A.R. Hoare: Algebraic specifications and proofs for communicating sequential processes. In: M. Broy (ed.): Logic of programming and calculi of discrete design. NATO ASI Series. Series F: Computer and Systems Sciences, Vol. 36. Berlin: Springer 1987, 277–300
20. B. Mahr, J.A. Makowsky: Characterizing specification languages which admit initial semantics. Theoretical Computer Science **31**, 49-59 (1984)
21. T.S.E. Maibaum, C.J.Lucena: Higher order data types. International Journal of Computer and Information Sciences **9**, 31-53 (1980)
22. A.I. Mal'cev: Ob umnoženii klassov algebraičeskih sistem. Sibir. Mat. Ž. **8**, 346–365 (1967). English translation: A few remarks on quasi varieties of algebraic systems. In A.I. Mal'cev: The metamathematics of algebraic systems. Collected papers 1936–1967. Studies in Logic and the Foundations of Mathematics, Vol. 66. Amsterdam: North-Holland 1971, 416–421
23. G. Markowsky: Chain-complete posets and directed sets with applications. Algebra Universalis **6**, 53-68 (1976)
24. K. Meinke: A recursive second order initial algebra specification of primitive recursion. Report CSR 8-91, Dept. of Computer Science, University College of Swansea, 1991. Also: Acta Informatica (to appear)
25. K. Meinke: Universal algebra in higher types. Theoretical Computer Science **100**, 385–417 (1992)
26. B. Möller: Unendliche Objekte und Geflechte. Fakultät für Mathematik und Informatik der TU München, Dissertation, Report TUM-I8213, 1982
27. B. Möller: An algebraic semantics for busy (data-driven) and lazy (demand-driven) evaluation and its application to a functional language. In: J. Diaz (ed.): Automata, languages and programming. Lecture Notes in Computer Science **154**. Berlin: Springer 1983, 513-526
28. B. Möller: On the algebraic specifications of infinite objects - Ordered and continuous models of algebraic types. Acta Informatica **22**, 537-578 (1985)
29. B. Möller: Algebraic specifications with higher-order operators. In L.G.L.T. Meertens (ed.): Program Specification and Transformation. Amsterdam: North-Holland 1987, 367–398
30. B. Möller: Higher-order algebraic specifications. Fakultät für Mathematik und Informatik der TU München, Habilitationsschrift 1987
31. B. Möller, A. Tarlecki, M. Wirsing: Algebraic specifications with built-in domain constructions. In M. Dauchet, M. Nivat (eds.): Proc 13th CAAP, Nancy, March 21–24, 1988. Lecture Notes in Computer Science **299**. Berlin: Springer 1988, 132–148
32. B. Möller, A. Tarlecki, M. Wirsing: Algebraic specifications of reachable higher-order algebras. In: D. Sannella, A. Tarlecki (eds.): Recent trends in data type specification. Lecture Notes in Computer Science **332**. Berlin: Springer 1988, 154–169
33. F. Nickl: Algebraic specifications for domain theory. In: B. Monien, R. Cori (eds.): Theoretical aspects of computer science. Lecture Notes in Computer Science **349**. Berlin: Springer 1989, 360–375. Extended version: Algebraic specifications of domain constructions. Fakultät für Mathematik und Informatik der Universität Passau, Report MIP-8815, 1988
34. M. Nivat: On the interpretation of recursive polyadic program schemes. Istituto Nazionale di Alta Matematica, Symposia Matematica XV. London: Academic Press 1975, 255-281

35. P. Padawitz: computing in Horn clause theories. EATCS Monographs on Theoretical Computer Science **16**. Berlin: Springer 1988
36. K. Parsaye-Ghomi: Higher-order abstract data types. Dept. of Computer Science, University of California at Los Angeles, Ph.D. Thesis, 1981
37. G.D. Plotkin: LCF considered as a programming language. Theoretical Computer Science **5**, 223-255 (1977)
38. A. Poigné: Higher-order data structures - cartesian closure versus λ-calculus. In: M. Fontet, K. Mehlhorn (eds.): Theoretical aspects of computer science. Lecture Notes in Computer Science **166**. Berlin: Springer 1984, 174-185
39. A. Poigné: On specifications, theories, and models with higher types. Information and Control **68**, 1-46 (1986)
40. D.S. Scott: Outline of a mathematical theory of computation. Proc. 4th Annual princeton Conference on Information Sciences and Systems, 1970, 169-176, and Technical Monograph PRG-2, Oxford University, Computing Laboratory, Programming Research Group, 1970
41. M.B. Smyth: Power domains. J. Computer Syst. Sci. **16**, 23-36 (1978)
42. A. Tarlecki: On the existence of free models in abstract algebraic institutions. Theoretical Computer Science **37**, 269–304 (1985)
43. A. Tarlecki: Quasi-varieties in abstract algebraic institutions. Journal of Computer and System Sciences **33**, 333–360 (1986)
44. A. Tarlecki, M. Wirsing: Continuous abstract data types. Fundamenta Informaticae **9**, 95-125 (1986)
45. Q. Zhenyu: An algebraic semantics of higher-order types with subtypes. Acta Informatica **30**, 569–607 (1993)