A Constructor- and Destructor-based Framework for Monotonic (Co-) Inductive Specification

Claus-Peter Wirth

Brandenburger Str. 42, D-65582 Diez, Germany
`wirth@logic.at`
`http://www.ags.uni-sb.de/~cp`

!!! !!! !!! !!! preliminary version !!! !!! !!! !!! !!!
July 23, 2008

Abstract: ???

Contents

1 Plan

1.1 Title

Notions of (Co) Inductive Validity for Constructor- and Destructor-based Specifications with Partially Defined Operations

1.2 Structure

Part 1 Introduction + (co-) inductive validities + example. 15 p..

Part 2 Monotonicity. 5 p..

Part 3 Relatedness of notions + Discussion + Meta-physics. 15 p..

Follow-up Paper Criteria for the confluence required in part 2. 40 p..

1.3 What the paper is about

CADE'94 paper + co inductive data domains.

1.4 What the paper is NOT about

1.4.1 On purpose

Co-Signatures This is fucking in the presence of terms because it renders the notion of a well-formed term to be a semantical one.

Co-Algebras

Finality

- The final objects can be proper classes and thus cannot provide something like a computational model.
- Congruence modulo bisimularity is not always given, e.g. for labeled state transition systems, for a choice-function on sets, or for some funny stacks of Joseph Goguen.

μ- and ν-Predicates

Termination and Sufficient Completeness Not intended as admissibility conditions.

Sorts that have constructors and destructors By this I mean sorts that have inductive as well as co-inductive objects and where the constructors are having no confusion. (I would not call the constructors of hidden sort of Peter Padawitz constructors because they have confusion.) Sorts in this sense, which are both visible and hidden may be nice in practice but theoretically hard to treat and not really necessary.

1.4.2 Due to lack of space

Proving

Higher-Order

1.5 Admissibility condition

The heads of the positive/negative-conditional rules may have the forms $\mathbb{D}^*\mathbb{K}\mathbb{C}^*$ where $\mathbb{D}$ are the destructor function symbols of the hidden sorts, $\mathbb{C}$ the constructors for the visible sorts, and $\mathbb{K}$ the other function symbols.

2 Matter

We assume the reader to be familiar with the basics of term rewriting (cf. e.g. Avenhaus & Madlener (1990), Klop (1992)) and algebraic specification (cf. e.g. Ehrig & Mahr (1985)).

Since our approach is based on the rigorous syntactic distinction of constructors and destructors, we have to be quite explicit about signatures, terms, substitutions, and algebras.

Definition 2.1 (Variable-System, $\mathcal{V}()$)
A *variable-system* for a set $\mathbb{S}$ of sorts and a set $\mathbb{K}$ of function symbols is a $\{\mathrm{GEN}, \mathrm{DAT}\}{\times}\mathbb{S}$-sorted family $\mathrm{V} = (\mathrm{V}_{\varsigma,s})_{(\varsigma,s)\in\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}}$ of classes $\mathrm{V}_{\varsigma,s}$ that are mutually disjoint and disjoint from $\mathbb{K}$. The variables from $\mathrm{V}_{\mathrm{DAT},s}$ and $\mathrm{V}_{\mathrm{GEN},s}$ are called the *data* and *general variables of sort* s, resp..
By abuse of notation we write V to denote $\bigcup_{(\varsigma,s)\in\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}} \mathrm{V}_{\varsigma,s}$, &c..
For k being any of these kinds of variables and for a term, formula Γ &c., '$\mathcal{V}_k(\Gamma)$' denotes the set of variables from V_k occurring in Γ.

Definition 2.2 (Signature)
A *signature*

$$\Sigma = (\mathbb{S}, \mathbb{V}, \mathbb{H}, \mathbb{K}, \mathbb{C}, \mathbb{D}, \alpha, \mathrm{V}, \bullet, C, \lesssim)$$

consists of

- a set $\mathbb{S}$ of sorts that partitions (i.e. $\mathbb{S} = \mathbb{V} \uplus \mathbb{H}$) into two disjoint sets $\mathbb{V}$ and $\mathbb{H}$ of *visible* and *hidden sorts*, resp.,
- a set $\mathbb{K}$ of function symbols with two disjoint subsets $\mathbb{C}$ and $\mathbb{D}$ of *constructors* and *destructors*, resp.,
- an arity function $\alpha : \mathbb{K} \to \mathbb{S}^+$,
- a variable-system V for $\mathbb{S}$ and $\mathbb{K}$ where $\mathrm{V}_{\varsigma,s}$ is infinite for each $(\varsigma, s) \in \{\mathrm{GEN}\}{\times}\mathbb{S} \cup \{\mathrm{DAT}\}{\times}\mathbb{H}$ and $\mathrm{V}_{\mathrm{DAT},s} = \emptyset$ for each $s \in \mathbb{H}$,

and of the following items that will be further explained later, cf. Definition 6.3,

- a function $\bullet$ with $\mathrm{dom}(\bullet) = \mathbb{H}$ and $\bullet_s \in \mathrm{V}_{\mathrm{GEN},s}$ for all $s \in \mathbb{H}$,
- a set C of *destructor patterns*,
- and a wellfounded quasi-ordering $\lesssim$ on $\mathbb{S}$.

When α is clear from the context, we write $f : s_0 \to \cdots \to s_n$ to denote $\alpha(f) = s_0 \ldots s_n$ for $f \in \mathbb{K}$, $n \in \mathbf{N}$, and $s_0, \ldots, s_n \in \mathbb{S}$.

Example 2.3 (Signature)
Let bool be the visible sort of Boolean values, nat be the visible sort of natural numbers, stream the hidden sort of finite and infinite streams of natural numbers, and set and list be the sorts of sets and finite lists of these streams, resp..
The constructors for bool are $\mathsf{true}, \mathsf{false} : \mathsf{bool}$. The constructors for nat are $\mathsf{0} : \mathsf{nat}$ and $\mathsf{s} : \mathsf{nat} \to \mathsf{nat}$. The constructors for list are $\mathsf{nil} : \mathsf{list}$ and $\mathsf{cons} : \mathsf{stream} \to \mathsf{list} \to \mathsf{list}$. We have the destructors $\mathsf{null} : \mathsf{stream} \to \mathsf{bool}$, $\mathsf{hd} : \mathsf{stream} \to \mathsf{nat}$, $\mathsf{tl} : \mathsf{stream} \to \mathsf{stream}$, and

$\mathsf{in} : \mathsf{stream} \to \mathsf{set} \to \mathsf{bool}$.
We have the three destructor patterns ($S \in \mathrm{V}_{\mathrm{GEN},\mathsf{stream}}$)

$$\begin{array}{lll} \mathsf{null}(\bullet_{\mathsf{stream}}) & & \\ \mathsf{hd}(\bullet_{\mathsf{stream}}) & \longleftarrow & \mathsf{null}(\bullet_{\mathsf{stream}})=\mathsf{false} \\ \mathsf{tl}(\bullet_{\mathsf{stream}}) & \longleftarrow & \mathsf{null}(\bullet_{\mathsf{stream}})=\mathsf{false} \\ \mathsf{in}(S, \bullet_{\mathsf{set}}) & \longleftarrow & \mathrm{Def}\ S \end{array}$$

They say that the behavior of a stream object is given by its value under null and, if this is false (i.e. the stream is not empty), also by its value under hd (returning the first element) and tl (returning the tail of the stream beyond that element). The behavior of a set is given the elements it contains; more precisely, by the value of the functions $\mathsf{in}(S) : \mathsf{set} \to \mathsf{bool}$ for $S \in \mathrm{V}_{\mathrm{GEN},\mathsf{stream}}$. Now, if S is a partially defined or undefined stream, we cannot expect $\mathsf{in}(S)$ to return a defined value, which we need, however, e.g. in order to find out that two singleton sets containing the same defined stream are behaviorally equal. Therefore, we have added the condition Def S to the above destructor pattern.
Finally, for our wellfounded quasi-ordering we take $\mathsf{bool} \lesssim \mathsf{stream}$ because the first destructor pattern has sort bool and also because the condition literal of the second and third destructor pattern has visible sort bool, we take $\mathsf{nat} \lesssim \mathsf{stream}$ because the head of the second destructor pattern has sort nat, we take $\mathsf{stream} < \mathsf{set}$ because stream is the sort of the condition literal of the fourth destructor pattern, and $\mathsf{stream} < \mathsf{list}$ because the constructor cons has hidden argument sort stream. Note that the quasi-ordering $\lesssim$ will be used to guarantee that visible sorts are not constructed from hidden sorts whose behavior they define, which would be a vicious circle.

Definition 2.4 (Terms)
Let $\Sigma = (\mathbb{S}, \mathbb{V}, \mathbb{H}, \mathbb{K}, \mathbb{C}, \mathbb{D}, \alpha, \mathrm{V}, \bullet, C, \lesssim)$ be a signature, X be a variable-system for $\mathbb{S}$ and $\mathbb{K}$, and $s \in \mathbb{S}$.
The set $\mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ of *terms of sort s* denotes the well-sorted first-order terms of fixed arity α over $\mathbb{K}$ and X of sort s. This means that $\mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ are the smallest classes such that $\mathrm{X}_{\mathrm{GEN},s} \uplus \mathrm{X}_{\mathrm{DAT},s} \subseteq \mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ and if $f : s_0 \to \cdots \to s_n$ $(n \in \mathbf{N})$ and, for each $i \prec n$, $t_i \in \mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s_i}$ then $f(t_0, \ldots, t_{n-1}) \in \mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s_n}$.
If $s \in \mathbb{V}$, then the terms in $\mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ are called *visible*. Otherwise, if $s \in \mathbb{H}$, they are *hidden*.
The set $\mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{DAT},s}$ of *data terms of sort s* denotes the subset of terms from $\mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ that only contain function symbols from $\mathbb{C}$ and variables from $(\mathrm{X}_{\mathrm{DAT},s})_{s \in \mathbb{S}}$.
The set $\mathcal{VGT}(\Sigma, \mathrm{X})_s$ of *visible-ground terms of sort s* denotes the subset of terms from $\mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ that only contain function symbols from $\mathbb{C}$ and variables of hidden sort.

Note that the visible-ground terms generalize the constructor ground terms of the constructor-based approach. The difference is that we cannot disallow hidden variables because there are no hidden constructor function symbols that could take their place.

Definition 2.5 (Condition Literal)
Let $\Sigma = (\mathbb{S}, \mathbb{V}, \mathbb{H}, \mathbb{K}, \mathbb{C}, \mathbb{D}, \alpha, \mathrm{V}, \bullet, C, \lesssim)$ be a signature, X be a variable-system for $\mathbb{S}$ and $\mathbb{K}$, and $s \in \mathbb{S}$.
A *condition literal over* Σ, X *of sort s* has one of the three forms 'Def u', '$u{=}u'$', or '$u{\neq}u'$' with $u, u' \in \mathcal{T}(\Sigma, \mathrm{X})_s$.

Definition 2.6 (Sensible Signature, Destructor Pattern, $\lhd$)
Let $\Sigma = (\mathbb{S}, \mathbb{V}, \mathbb{H}, \mathbb{K}, \mathbb{C}, \mathbb{D}, \alpha, \mathrm{V}, \bullet, C, \lesssim)$ be a signature.
Let $<$ be the ordering of $\lesssim$. We define the wellfounded ordering $\lhd$ on $\mathbb{S}$ by ($s, s' \in \mathbb{S}$):

$$s \lhd s' \quad \text{if} \quad s < s' \quad \text{or} \quad s \lesssim s' \wedge s \in \mathbb{V} \wedge s' \in \mathbb{H}.$$

Now, Σ is *sensible* if

- for each constructor $c \in \mathbb{C}$ with say $c : s_0 \to \cdots \to s_n$ ($n \in \mathbf{N}$) we have $s_n \in \mathbb{V}$ and for each $i \prec n$ we have $(s_i \in \mathbb{V} \Rightarrow s_i \lesssim s_n)$ and $(s_i \in \mathbb{H} \Rightarrow s_i < s_n)$.
- for each destructor $d \in \mathbb{D}$ with say $d : s_0 \to \cdots \to s_n$ ($n \in \mathbf{N}$) there is some $i \prec n$ with $s_i \in \mathbb{H}$ and for each $j \prec n$ with $s_j \in \mathbb{V}$ we have $s_j \lesssim s_i$; and
- each destructor pattern from C has the form $d(x_0, \ldots, x_n) \longleftarrow L_0, \ldots, L_{m-1}$ where $n, m \in \mathbf{N}$ (for $m = 0$ we omit the '$\longleftarrow$'), $d \in \mathbb{D}$ is a destructor with $d : s_0 \to \cdots \to s_{n+1}$, $\{x_0, \ldots, x_n\} \subseteq \mathrm{V}$, $d(x_0, \ldots, x_n)$ is a linear term, there is exactly one $i \preceq n$ with $x_i \in \mathrm{ran}(\bullet)$ (we speak of a *destructor pattern for the sort s_i and of sort s_{n+1}*), and for this i we have $s_{n+1} \lesssim s_i$ and for each $j \prec m$, L_j is a condition literal over Σ, V of sort l_j with $\mathcal{V}(L_j) \subseteq \{x_0, \ldots, x_n\}$ and $l_j \lhd s_i$, and
- for each $s \in \mathbb{S}$ we have $\mathcal{VGT}(\Sigma, \mathrm{V})_s \neq \emptyset$.

Note that the quasi-ordering $\lesssim$ guarantees that visible sorts are not constructed from hidden sorts whose behavior they define, which would be a vicious circle. Thus, the result sort of a constructor must be visible and its argument sorts must be somehow smaller, whereas the hidden argument sorts of a destructor must be somehow bigger than its result sort, the sorts of its visible arguments, and the sorts of the condition literals of its destructor patterns. Note that in the absence of hidden sorts we can choose $\lesssim$ to be $\mathbb{V} \times \mathbb{V}$. This renders the conditions imposed on a sensible signature to be satisfied, so that Definition 2.6 is really a generalization of the standard constructor-based notion of a sensible (or inhabited) many-sorted signature to one with hidden sorts and destructors.

Furthermore, note that the signature of Example 2.3 is sensible.

Definition 2.7 (Substitutions)
Let $\Sigma = (\mathbb{S}, \mathbb{V}, \mathbb{H}, \mathbb{K}, \mathbb{C}, \mathbb{D}, \alpha, \mathrm{V}, \bullet, C, \lesssim)$ be a signature. Let X be a variable-system for $\mathbb{S}$ and $\mathbb{K}$. Let $T = (T_{\varsigma,s})_{(\varsigma,s) \in \{\mathrm{GEN},\mathrm{DAT}\} \times \mathbb{S}}$ be any $\{\mathrm{GEN}, \mathrm{DAT}\} \times \mathbb{S}$-sorted family of classes. With $\mathrm{X} \to T$ we denote the $\{\mathrm{GEN}, \mathrm{DAT}\} \times \mathbb{S}$-sorted (total) functions from X to T. If T is a family of terms, then $\mathrm{X} \to T$ is a class of *substitutions*. E.g., if $\sigma \in \mathrm{X} \to \mathcal{T}(\Sigma, \mathrm{X})$ (or simply written $\sigma : \mathrm{X} \to \mathcal{T}(\Sigma, \mathrm{X})$), then, for any $X \in \mathrm{X}_{\mathrm{GEN},s}$, $\sigma(X)$ is a term of sort s, and for any $x \in \mathrm{X}_{\mathrm{DAT},s}$, $\sigma(x)$ is a data term of sort s.
For a substitution σ we denote with '$\Gamma\sigma$' the result of replacing in Γ each occurrence of a variable $x \in \mathrm{dom}(\sigma)$ with $\sigma(x)$.

3 Modeling Partial Functions with Total Structures

Traditionally, algebras or structures are total in the sense that they assign total functions to the function symbols of their signature.

Contrary to this, most working mathematicians would be very unhappy if they were not able to use partial functions. Similarly, some people working on algebraic specification are very fond of partial functions: The idea that a function symbol denotes a partial function results in so-called *partial* algebras, cf. the fascinating book of Reichel (1987), which, however, already makes clear that partial algebras are not as simple as total ones. Similarly, category theorists deal with *co* algebras, where partiality can easily be modeled because the co domain of a function symbol of a co signature is a sum into that $1 = \{\bot\}$ can nicely be included without the usual inconvenience of an extra sort for possibly erroneous values, cf. e.g. Jacobs & Rutten (1997), Padawitz (2000).

Again contrary to this, total algebras are preferred by computer scientists working on theorem proving and algebra in implementations that deal with a very big number of terms. This is because the soundness of these implementations requires that each occurring term object really is a term, and it is practically not possible to prove this again and again. Note that e.g. in a partial or co algebra containing the natural numbers nat and the predecessor function $\mathsf{p} : \mathsf{nat} \rightsquigarrow \mathsf{nat}$ or $\mathsf{p} : \mathsf{nat} \to \mathsf{nat} + 1$, resp., and a ground term t of sort nat, whether $\mathsf{p}(t)$ is defined and whether $\mathsf{p}(\mathsf{p}(t))$ is a term or not depends on the evaluation of t in the algebra and therefore is a semantical notion instead of a purely syntactical one.

Our solution to this dilemma is to model partial functions with total algebras, cf. Wirth & Gramlich (1994b). In this paper, we also include predicates besides (characteristic) functions because they play an essential role as destructors of co inductive data domains. For a predicate $p : s_0, \ldots, s_{n-1}$ (with $s_0, \ldots, s_{n-1} \in \mathbb{S}$ being sorts of the signature Σ) we, however, will write $p : s_0, \ldots, s_{n-1} \rightsquigarrow \mathsf{pred}$ (with $\mathsf{pred} \notin \mathbb{S}$) and include it into the set $\mathbb{K}$ of constant (= non-variable) function and predicate symbols of the signature Σ, i.e. $p \in \mathbb{K}$.

Consider a partial algebra $\mathcal{D}$ that satisfies $\mathsf{pred}^{\mathcal{D}} = \{\mathsf{TRUE}\}$ and assigns to each sort $s \in \mathbb{S}$ and each constant $f : s_0, \ldots, s_{n-1} \rightsquigarrow s_n$ of its signature Σ a non-empty set say $s^{\mathcal{D}}$ and a (strict) (possibly) partial function $f^{\mathcal{D}} : s_0^{\mathcal{D}} \times \cdots \times s_{n-1}^{\mathcal{D}} \rightsquigarrow s_n^{\mathcal{D}}$, resp.. We model this partial algebra with a class K of (total) Σ-*structures* $\mathcal{A}$ of the following form: $\mathcal{A}$ assigns to each sort $s \in \mathbb{S}$ and each constant $f : s_0, \ldots, s_{n-1} \rightsquigarrow s_n$ of Σ, resp., two non-empty sets say $(\mathrm{DAT}, s)^{\mathcal{A}}$ and $(\mathrm{GEN}, s)^{\mathcal{A}}$ with $(\mathrm{DAT}, s)^{\mathcal{A}} = s^{\mathcal{D}}$ and $(\mathrm{DAT}, s)^{\mathcal{A}} \subseteq (\mathrm{GEN}, s)^{\mathcal{A}}$, and in case of $s_n \neq \mathsf{pred}$ a total function

$$f^{\mathcal{A}} : (\mathrm{GEN}, s_0)^{\mathcal{A}} \times \cdots \times (\mathrm{GEN}, s_{n-1})^{\mathcal{A}} \to (\mathrm{GEN}, s_n)^{\mathcal{A}}$$

with $f^{\mathcal{D}} \subseteq f^{\mathcal{A}}$, and in case of $s_n = \mathsf{pred}$ a relation

$$f^{\mathcal{A}} \subseteq (\mathrm{GEN}, s_0)^{\mathcal{A}} \times \cdots \times (\mathrm{GEN}, s_{n-1})^{\mathcal{A}}$$

with $\mathrm{dom}(f^{\mathcal{D}}) \subseteq f^{\mathcal{A}}$. We call $(\mathrm{GEN}, s)^{\mathcal{A}}$ the *general universe of $\mathcal{A}$ for the sort s*, whereas $(\mathrm{DAT}, s)^{\mathcal{A}}$ is called the *data universe of $\mathcal{A}$ for the sort s.*

Let us define a *variable-system* for a set $\mathbb{S}$ of sorts and a set $\mathbb{K}$ of constants (= non-variables) to be a $\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}$-sorted family $\mathrm{V} = (\mathrm{V}_{\varsigma,s})_{(\varsigma,s)\in\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}}$ of sets $\mathrm{V}_{\varsigma,s}$ that are mutually disjoint and disjoint from $\mathbb{K}$. The variables from $\mathrm{V}_{\mathrm{DAT},s}$ and $\mathrm{V}_{\mathrm{GEN},s}$ are called the *data* and *general variables of sort* s, resp.. By abuse of notation we sometimes write V to denote $\bigcup_{(\varsigma,s)\in\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}} \mathrm{V}_{\varsigma,s}$, &c.. For k being any of these kinds of variables and for a term, formula Γ &c., '$\mathcal{V}_k(\Gamma)$' denotes the set of variables from V_k occurring in Γ. The set $\mathcal{T}(\Sigma,\mathrm{V})_{\mathrm{GEN},s}$ of *terms of sort* s denotes the well-sorted first-order terms of fixed arity over $\mathbb{K}$ and V of sort $s\in\mathbb{S}$, where $\mathbb{S}$ and $\mathbb{K}$ are the set of sorts and the set of constants given by Σ, resp.. This means that, for $s\in\mathbb{S}$, the $\mathcal{T}(\Sigma,\mathrm{V})_{\mathrm{GEN},s}$ are the smallest sets such that $\mathrm{V}_{\mathrm{GEN},s}\uplus\mathrm{V}_{\mathrm{DAT},s}\subseteq\mathcal{T}(\Sigma,\mathrm{V})_{\mathrm{GEN},s}$ and if $f : s_0,\ldots,s_{n-1}\rightsquigarrow s_n$ $(n\in\mathbf{N};\ s_0,\ldots,s_n\in\mathbb{S})$ and, for each $i\prec n$, $t_i\in\mathcal{T}(\Sigma,\mathrm{V})_{\mathrm{GEN},s_i}$ then $f(t_0,\ldots,t_{n-1})\in\mathcal{T}(\Sigma,\mathrm{V})_{\mathrm{GEN},s_n}$. The set $\mathcal{T}(\Sigma,\mathrm{V})_{\mathrm{DAT},s}$ of *data terms of sort* s denotes the subclass of terms from $\mathcal{T}(\Sigma,\mathrm{X})_{\mathrm{GEN},s}$ that only contain variables from $\mathrm{V}_{\mathrm{DAT}}$ and constants from a special set $\mathbb{C}\subseteq\mathbb{K}$ of *constructors* that is given by the signature Σ. For such a constructor $c\in\mathbb{C}$ with $c : s_0,\ldots,s_{n-1}\rightarrow s_n$ we require $s_n\neq\mathsf{pred}$ and any Σ-structure $\mathcal{A}$ must satisfy

$$\forall i\prec n.\ d_i\in(\mathrm{DAT},s_i)^{\mathcal{A}} \quad\Rightarrow\quad c^{\mathcal{A}}(d_0,\ldots,d_{n-1})\in(\mathrm{DAT},s_n)^{\mathcal{A}}.$$

Let $T = (T_{\varsigma,s})_{(\varsigma,s)\in\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}}$ be any $\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}$-sorted family of sets. With $\mathrm{V}\rightarrow T$ we denote the ($\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}$-sorted) (total) functions from V to T. On the one hand, if T is a family of sets of terms, then the elements of $\mathrm{V}\rightarrow T$ are *substitutions*. E.g., if $\sigma : \mathrm{V}\rightarrow\mathcal{T}(\Sigma,\mathrm{V})$, then, for any $X\in\mathrm{V}_{\mathrm{GEN},s}$, $\sigma(X)$ is a term of sort s, and for any $x\in\mathrm{V}_{\mathrm{DAT},s}$, $\sigma(x)$ is a data term of sort s. For a substitution σ we denote with '$\Gamma\sigma$' the result of replacing in Γ each occurrence of any variable $x\in\mathrm{dom}(\sigma)$ with $\sigma(x)$. On the other hand, if T is a Σ-structure $\mathcal{A}$, then we use $\mathcal{A}$ to denote $((\varsigma,s)^{\mathcal{A}})_{(\varsigma,s)\in\{\mathrm{GEN},\mathrm{DAT}\}\times\mathbb{S}}$ and the elements of $\mathrm{V}\rightarrow\mathcal{A}$ are the (total) valuations from V to $\mathcal{A}$.

For $\tau : \mathrm{V}\rightarrow\mathcal{A}$ we denote with '$\mathcal{A}\uplus\tau$' the extension of $\mathcal{A}$ to the variables of V which are then treated as nullary constants. More precisely, we assume the existence of some evaluation function 'eval' such that $\mathrm{eval}(\mathcal{A}\uplus\tau)$ maps any term over $\Sigma\uplus\mathrm{V}$ into the universe of $\mathcal{A}$ (respecting kinds) such that for all $x\in\mathrm{V}$: $\mathrm{eval}(\mathcal{A}\uplus\tau)(x)=\tau(x)$. Moreover, $\mathrm{eval}(\mathcal{A}\uplus\tau)$ maps any formula B over $\Sigma\uplus\mathrm{V}$ to TRUE or FALSE, such that B is valid in $\mathcal{A}\uplus\tau$ iff $\mathrm{eval}(\mathcal{A}\uplus\tau)(B)=\mathsf{TRUE}$. We assume that the *Substitution-Lemma* (cf. e.g. Enderton (1973), p. 127, or Fitting (1996), p. 120) holds in the sense that, for any substitution σ, Σ-structure $\mathcal{A}$, valuation $\tau : \mathrm{V}\rightarrow\mathcal{A}$, and term or formula B over $\Sigma\uplus\mathrm{V}$:

$$\mathrm{eval}(\mathcal{A}\uplus\tau)(B\sigma) = \mathrm{eval}\big(\ \mathcal{A}\ \uplus\ (\ \sigma\ \uplus\ _{\mathrm{V}\setminus\mathrm{dom}(\sigma)}\!\upharpoonright\!\mathrm{id}\ )\ \circ\ \mathrm{eval}(\mathcal{A}\uplus\tau)\ \big)\big(\ B\ \big).$$

Note that we assume the operator '$\circ$' to have higher priority than the operators '$\cup$' and '$\uplus$'.

Now, when the partial algebra $\mathcal{D}$ satisfies a first-order specification where all terms are defined and when we consider all variables occurring in this specification to be from $\mathrm{V}_{\mathrm{DAT}}$, then the specification is satisfied by each Σ-structure $\mathcal{A}$ in our above class K. Note that the variables from $\mathrm{V}_{\mathrm{GEN}}$ are useful for specifying non-strict partial functions:

Example 3.1
Let $\mathsf{true}, \mathsf{false} : \mathsf{bool}$ be the constructors for the sort bool of Boolean values. Let $\mathsf{or} : \mathsf{bool}, \mathsf{bool} \to \mathsf{bool}$ be from $\mathbb{K}\backslash\mathbb{C}$. Let $X : \mathrm{V}_{\mathrm{GEN},\mathsf{bool}}$ and $x : \mathrm{V}_{\mathrm{DAT},\mathsf{bool}}$. Let t be a ground term of sort bool that does not evaluate to the data universe of sort bool given by the values of true and false under the evaluation function. Now specifying or via $\mathsf{or}(x, \mathsf{false}){=}x$ and $\mathsf{or}(x, \mathsf{true}){=}\mathsf{true}$ has the same syntactical and semantical effect as specifying it via $\mathsf{or}(\mathsf{true}, \mathsf{true}){=}\mathsf{true}$, $\mathsf{or}(\mathsf{false}, \mathsf{true}){=}\mathsf{true}$, $\mathsf{or}(\mathsf{true}, \mathsf{false}){=}\mathsf{true}$, $\mathsf{or}(\mathsf{false}, \mathsf{false}){=}\mathsf{false}$. Moreover, adding $\mathsf{or}(\mathsf{true}, x){=}\mathsf{true}$ has no effect anymore. Contrariwise, $\mathsf{or}(X, \mathsf{false}){=}X$, $\mathsf{or}(X, \mathsf{true}){=}\mathsf{true}$ also specifies that $\mathsf{or}(t, \mathsf{true}){=}t$ and adding $\mathsf{or}(\mathsf{true}, X){=}\mathsf{true}$ has the additional effect of implying $\mathsf{or}(\mathsf{true}, t){=}\mathsf{true}$. By the way, the readers familiar with rewriting systems may note that all these equations do not produce a single proper critical pair.

Continuing our modeling of partiality, we add a predicate $\mathrm{Def}_s : s \leadsto \mathsf{pred}$ (index omitted when clear from context) for each sort s of Σ that is interpreted by any Σ-structure $\mathcal{A}$ as $(\mathrm{DAT}, s)^{\mathcal{A}}$. Now, if a ground term t over Σ is defined w.r.t. $\mathcal{D}$, then $\mathrm{Def}\, t$ holds in each Σ-structure $\mathcal{A}$ of our above class K. The converse is also true, provided that K contains all possible Σ-structures of the above kind. To wit, include the Σ-structure given by $(\mathrm{GEN}, s)^{\mathcal{A}} = (\mathrm{DAT}, s)^{\mathcal{A}} \uplus \{\bot\}$ and the functions evaluating to $\bot$ when undefined in $\mathcal{D}$ or when applied to $\bot$. Considering this Σ-structure, the question arises what the possibility to choose $(\mathrm{GEN}, s)^{\mathcal{A}}$ different from $(\mathrm{DAT}, s)^{\mathcal{A}} \uplus \{\bot\}$ is good for. The simple answer is that, on one hand, we want to be able to model non-strict functions like or in Example 3.1 above and, on the other hand, we do not want silly theorems to hold, like $\frac{\mathsf{0}}{\mathsf{0}}{=}\mathsf{p}(\mathsf{0})$ (due to $\bot{=}\bot$) that becomes invalid when we add $\mathsf{p}(\mathsf{0}) = \mathsf{0}$ to the specification of natural numbers.

Well-known equality notions like (in order of increasing logical strength) *weak*, *strong*, and *existential* equality of two terms t, t' can now be expressed via $t{=}t' \Leftarrow (\mathrm{Def}\, t \wedge \mathrm{Def}\, t')$, $(t{=}t' \wedge \mathrm{Def}\, t \wedge \mathrm{Def}\, t') \vee (\neg\mathrm{Def}\, t \wedge \neg\mathrm{Def}\, t')$, and $t{=}t' \wedge \mathrm{Def}\, t \wedge \mathrm{Def}\, t'$, resp., but this is not our subject here.

Since, in our modeling, the notion of a term is a purely syntactical one and for reasons we will see later, this modeling of partial functions with total Σ-structures makes the computer scientists perfectly happy. People working on partial or co algebras in practice should also be interested in it. Finally, the working mathematicians are confident with our modeling because they are not really interested in the logical formalization of partial functions, at least as long as there is one and someone who could principally formalize the proofs containing partial functions.

4 Basic Notions for Σ-Structures

Definition 4.1 (Enrichment, Reduct)
Let Σ be a signature with $\mathbb{S}$, $\mathbb{K}$, $\mathbb{C}$ as sets of sorts, constants, and constructors, resp..
Σ' is a *an enrichment* of Σ if Σ' is a signature whose sets of sorts, constants, and constructors are a superclasses of the resp. sets of Σ and the arities of the common constants are identical in both signatures.
When Σ' is an enrichment of Σ and $\mathcal{B}$ is a Σ'-structure, then the Σ-*reduct* of $\mathcal{B}$ is the Σ-structure $\mathcal{A}$ given by $(\varsigma, s)^{\mathcal{A}} = (\varsigma, s)^{\mathcal{B}}$ for all $(\varsigma, s) \in \{\mathrm{GEN}, \mathrm{DAT}\} \times \mathbb{S}$ and $f^{\mathcal{A}} = f^{\mathcal{B}}$ for all $f \in \mathbb{K}$.
When Σ'' is the sub-signature of Σ that contains only the constants from $\mathbb{C}$ and $\mathcal{A}$ is a Σ-structure, then the *data reduct of* $\mathcal{A}$ is the standard total algebra $\mathcal{D}$ for Σ'' given by $s^{\mathcal{D}} = (\mathrm{DAT}, s)^{\mathcal{A}}$ for all $s \in \mathbb{S}$ and $c^{\mathcal{D}} = c^{\mathcal{A}}$ for all $c \in \mathbb{C}$.

Note that the following definition differs from the standard one.

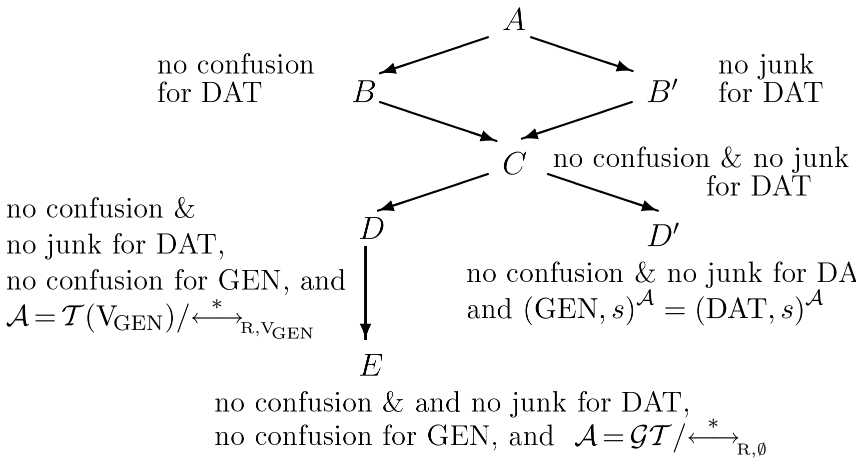
Definition 4.2 (Σ-Homomorphism)
Let Σ be a signature, with $\mathbb{S}$ and $\mathbb{K}$ as sets of sorts and constants, resp..
A Σ-*homomorphism from* $\mathcal{A}$ *to* $\mathcal{B}$ is a triple written $h{::}\mathcal{A}{\to}\mathcal{B}$ of two Σ-structures $\mathcal{A}$ and $\mathcal{B}$ *whose data reducts are equal (!!!)* and an $\mathbb{S}$-sorted family $h = (h_s)_{s \in \mathbb{S}}$ of functions $h_s : (\mathrm{GEN}, s)^{\mathcal{A}} \to (\mathrm{GEN}, s)^{\mathcal{B}}$ that is compatible with Σ in the following sense:
For $f \in \mathbb{K}$ with $f : s_0, \ldots, s_{n-1} \rightsquigarrow s_n$ and for $a_0 \in (\mathrm{GEN}, s_0)^{\mathcal{A}}, \ \ldots, \ a_{n-1} \in (\mathrm{GEN}, s_{n-1})^{\mathcal{A}}$
in case of $s_n \neq \mathsf{pred}$ we have $h_{s_n}(f^{\mathcal{A}}(a_0, \ldots, a_{n-1})) = f^{\mathcal{B}}(h_{s_0}(a_0), \ldots, h_{s_{n-1}}(a_{n-1}))$, and
in case of $s_n = \mathsf{pred}$ we have $(a_0, \ldots, a_{n-1}) \in f^{\mathcal{A}} \Rightarrow (h_{s_0}(a_0), \ldots, h_{s_{n-1}}(a_{n-1})) \in f^{\mathcal{B}}$, and
for all $s \in \mathbb{S}$ and $d \in (\mathrm{DAT}, s)^{\mathcal{A}}$ we have $h_s(d) = d$.

Definition 4.3 (Initial, Free)
Let X be a variable-system for $\mathbb{S}$ and $\mathbb{K}$ of the signature Σ.
Let $\mathcal{A}$ be a Σ-structure and K a class of Σ-structures.
$\mathcal{A}$ is *initial in* K if $\mathcal{A}$ is in K and, for each $\mathcal{B}$ in K, there is a unique Σ-homomorphism $h{::}\mathcal{A}{\to}\mathcal{B}$.
$\mathcal{A}$ is *free in* K *over* X if there is some $\iota : \mathrm{X} \to \mathcal{A}$ such that $\mathcal{A}$ is free in K over X w.r.t. ι.
$\mathcal{A}$ is *free in* K *over* X *w.r.t.* ι if $\mathcal{A}$ is in K and $\mathcal{A}$ is free for K over X w.r.t. ι.
$\mathcal{A}$ is *free for* K *over* X *w.r.t.* ι if for each $\mathcal{B}$ in K and $\tau : \mathrm{X} \to \mathcal{B}$ there is a unique Σ-homomorphism $h{::}\mathcal{A}{\to}\mathcal{B}$ with $\tau_{\varsigma,s} = \iota_{\varsigma,s} \circ h_s$ for each $(\varsigma, s) \in \{\mathrm{GEN}, \mathrm{DAT}\} \times \mathbb{S}$.

5 Four Notions of Validity

Please! Never again write a paper with seven notions of validity!

— Deepak Kapur, 1996



In Wirth & Gramlich (1994b) and still in Wirth (1997), we had seven notions of (inductive) validity. After application and consideration for years, we would now say that one or two (type C and maybe type D) are really useful, two more are important because they are still common (type E, D'), and the remaining three (type A, B, and B') are only interesting for the history of inductive validity and will not be treated in this paper, although they were once used by other authors as in Kounalis & Rusinowitch (1990) and Bouhoula &al. (1992). The names of the types were chosen such that, roughly speaking, if T precedes T' in the alphabet, then type-T validity of a formula implies type-T' validity of it.

*Specifications should express only what is important for the specifier.
A proper semantics for such partial specifications must be based on
a class of different models instead of a single one.
A free or initial model can serve for a minimal implementation
but is not adequate for defining the semantics of partial specifications.*

— Robert Bookhade, 1999

Definition 5.1 (Specification, Validity)
Let Σ be a signature with $\mathbb{S}$ and $\mathbb{K}$ as sets of sorts and constants, resp.. Let $\mathcal{D}$ be the data reduct of some Σ-structure. Let R be a set of formulas over Σ. We will call $(\Sigma,\mathcal{D},R)$ a *specification*. Let $\mathcal{A}$ be a Σ-structure such that R is valid in $\mathcal{A}$.
Now, $\mathcal{A}$ is a *model of* $(\Sigma,\mathcal{D},R)$ *of type*

C if the data reduct of $\mathcal{A}$ is $\mathcal{D}$.

D if for some variable-system X for $\mathbb{S}$, $\mathbb{K}$ with $\forall s\in\mathbb{S}.\ \mathrm{X}_{\mathrm{DAT},s}=\emptyset$, $\mathcal{A}$ is free in the class of models of $(\Sigma,\mathcal{D},R)$ of type C over X.

E if $\mathcal{A}$ is initial in the class of models of $(\Sigma,\mathcal{D},R)$ of type C.

D' if $\mathcal{A}$ is a model of $(\Sigma,\mathcal{D},R)$ of type C, and $(\mathrm{DAT},s)^{\mathcal{A}}=(\mathrm{GEN},s)^{\mathcal{A}}$ for each $s\in\mathbb{S}$; i.e. the general universes coincide with the data universes.

Finally, for T being any of the above types, a formula is *type-T valid w.r.t.* $(\Sigma,\mathcal{D},R)$ if it is valid in all models of $(\Sigma,\mathcal{D},R)$ of type T.

Corollary 5.2 *If a formula is type-C valid w.r.t.* $(\Sigma, \mathcal{D}, R)$*, then it is also type-D, type-E, and type-*D' *valid w.r.t.* $(\Sigma, \mathcal{D}, R)$*.*

Definition 5.3 (Extension)
$(\Sigma', \mathcal{D}', R')$ is an *extension* of of the specification $(\Sigma, \mathcal{D}, R)$ if $(\Sigma', \mathcal{D}', R')$ is a specification such that Σ' is a an enrichment of Σ, $s^{\mathcal{D}'} = s^{\mathcal{D}}$ and $c^{\mathcal{D}'} = c^{\mathcal{D}}$ for all $s \in \mathbb{S}$ and $c \in \mathbb{C}$ where $\mathbb{S}$ and $\mathbb{C}$ are the sets of sorts and constructor symbols of Σ, resp., and $R \subseteq R'$.
Such an extension is called *type-T consistent* if the Σ-reduct of any model of $(\Sigma', \mathcal{D}', R')$ of type-T is a model of $(\Sigma, \mathcal{D}, R)$ of type-T.

Note that consistent extensions (where the reducts of new models must be old ones) are somehow the converse of the well-known *conservative extensions*, where old models must be extensible to new models.

6 Data Universes

In this section we discuss two ways of establishing the data universes of the common data reduct $\mathcal{D}$ of the Σ-structures of our classes of models. In order to do this, we first add some more slots to our signature objects. We assume that each signature Σ contains a wellfounded quasi-ordering $\lesssim$ on $\mathbb{S} \uplus \{\mathsf{pred}\}$ with pred as strict minimum (i.e. $\forall s \in \mathbb{S}.\ \mathsf{pred} < s$, where $< := \lesssim \backslash \gtrsim$ is the (strict) ordering of $\lesssim$), and two disjoint sets $\mathbb{V}$ and $\mathbb{H}$ of *visible* and *hidden sorts*, resp., that partition the set of sorts $\mathbb{S}$, i.e. $\mathbb{S} = \mathbb{V} \uplus \mathbb{H}$. The set $\mathbb{H}$ of hidden sorts again partitions into the sets $\mathbb{H}_{\neq}$ and $\mathbb{H}_{\mathsf{pred}}$ because we cannot use both inequality and predicates for defining a hidden sort, cf. § 6.2.

Moreover, we require that the equivalence classes of the quasi-ordering $\lesssim$ (i.e. those of the equivalence $\sim := \lesssim \cap \gtrsim$) are subclasses of $\mathbb{V}$, $\mathbb{H}_{\neq}$, $\mathbb{H}_{\mathsf{pred}}$, or $\{\mathsf{pred}\}$. Now, the data universes of such an equivalence class of sorts are built up in parallel, assuming that the data universes for all sorts strictly smaller in the ordering $<$ have already been established. The manner in that this is done depends on whether the equivalence class is a subclass of $\mathbb{V}$ or of $\mathbb{H}$, as discussed in the following two subsections. While visible sorts are standard, hidden sorts are still not generally understood.

6.1 Visible Sorts and Constructors

For any constructor $c \in \mathbb{C}$ with say $c : s_0, \ldots, s_{n-1} \to s_n$ $(n \in \mathbf{N})$ we require that $s_n \in \mathbb{V}$ and $\forall i \prec n.\ s_i \lesssim s_n$, and call c a *constructor for* s_n. Let V be the equivalence class of s_n w.r.t. $\sim$. For any $d_i \in s_i^{\mathcal{D}}$ $(i \prec n)$, the term $c(d_0, \ldots, d_{n-1})$ becomes an element in $s_n^{\mathcal{D}}$. While the $s_i^{\mathcal{D}}$ with $s_i < s_n$ are assumed to be already given, all $v^{\mathcal{D}}$ with $v \in V$ are being defined as the smallest sets closed under the above term construction with all constructors for v. Moreover, in order to avoid empty data universes, we require that, for each $v \in V$, there is a term of sort v that only consists of constructors for the sorts in V and variables of sorts say s' with $s' < s_n$. Finally, in the above context, we define $c^{\mathcal{D}}(d_0, \ldots, d_{n-1}) := c(d_0, \ldots, d_{n-1})$, which immediately gives us the required $c^{\mathcal{D}}(d_0, \ldots, d_{n-1}) \in s_n^{\mathcal{D}}$.

Example 6.1
The and/or and or/and trees over bool can be realized with the visible sorts and and or, resp., when we set $\mathsf{bool} < \mathsf{or} \sim \mathsf{and}$ and take as constructors

$$\begin{array}{lll} [\_]^{\mathsf{or}}_{\mathsf{bool}} : \mathsf{bool} \to \mathsf{or}, & [\_]^{\mathsf{or}}_{\mathsf{and}} : \mathsf{and} \to \mathsf{or}, & \mathsf{cons}_{\mathsf{or}} : \mathsf{and}, \mathsf{or} \to \mathsf{or}, \\ [\_]^{\mathsf{and}}_{\mathsf{bool}} : \mathsf{bool} \to \mathsf{and}, & [\_]^{\mathsf{and}}_{\mathsf{or}} : \mathsf{or} \to \mathsf{and}, & \mathsf{cons}_{\mathsf{and}} : \mathsf{or}, \mathsf{and} \to \mathsf{and}. \end{array}$$

An and/or tree like $\wedge(a, \vee(b, c, \wedge(d, \vee(e)), f))$ is then represented as the term

$$\mathsf{cons}_{\mathsf{and}}([a]^{\mathsf{or}}_{\mathsf{bool}}, [\mathsf{cons}_{\mathsf{or}}([b]^{\mathsf{and}}_{\mathsf{bool}}, \mathsf{cons}_{\mathsf{or}}([c]^{\mathsf{and}}_{\mathsf{bool}}, \mathsf{cons}_{\mathsf{or}}(\mathsf{cons}_{\mathsf{and}}([d]^{\mathsf{or}}_{\mathsf{bool}}, [[e]^{\mathsf{or}}_{\mathsf{bool}}]^{\mathsf{and}}_{\mathsf{or}}), [f]^{\mathsf{or}}_{\mathsf{bool}})))]^{\mathsf{and}}_{\mathsf{or}}).$$

While the presentation of this term to human beings cries for some syntactic sugaring, the term can well serve as an implementation and has the advantage that all children of the root of an and/or-tree are guaranteed to be or/and-trees, provided that the term is well-sorted.

Note that, theoretically, any data universe can be established with a visible sort simply by introducing a nullary constructor symbol for exactly each element of the intended data universe.

Summing up, the data universes for visible sorts are built up the following way:

No Junk

One starts with the empty set and keeps adding terms that apply a constructor. This process will come to an end at level ω. Another way to say this is that the data universes are the least fixpoints of the finitary and monotonic functor that returns the set of terms that apply a constructor to its argument set. The buzzword "no junk" emphasizes that the data universes contain nothing but what is generated by the constructors as useful data.

No Confusion

All elements that have a different construction process are considered to be different. The buzzword "no confusion" actually only means that there is as little equality between the elements as possible or, stated differently, that the universe can be homomorphically mapped to any other universe that satisfies some required axioms. E.g., when we used lists to represent finite sets, we would need the axioms of associativity, commutativity, and idempotence. In former papers we used to allow such axioms in the form of positive conditional constructor equations. Since these constructor equations produced a lot of trouble regarding consistency and theorem proving we did not include them in the implementation of the tactic-based inductive theorem prover QuodLibet, cf. Kühler (2000), and by now we also do not want to have them in our theoretical considerations anymore because we can have sets as a hidden sort and the finite sets are simply those where the function $\mathsf{card} : \mathsf{set} \rightsquigarrow \mathsf{nat}$ is defined.

In Wirth & Gramlich (1994b) we even had types of validity where junk or confusion could occur in the data universes for visible sorts. Today we do not consider this to be appropriate anymore.

6.2 Hidden Sorts and Destructors

Definition 6.2 (Condition Literal)
Let Σ be a signature, X be a variable-system for $\mathbb{S}$,$\mathbb{K}$, and $s \in \mathbb{S} \uplus \{\mathsf{pred}\}$.
A *condition literal over* Σ,X *of sort* s has one of the four forms

$\mathrm{Def}_s u$ where $u \in \mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ and $s \in \mathbb{S}$.
This literal is not *conjugatible* and its *terms* are the single term u.

$u_0 {=_s} u_1$ where $u_0, u_1 \in \mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ and $s \in \mathbb{S}$.
This literal is *conjugatible* to its *conjugate* $u_0 {\neq_s} u_1$ if $s \in \mathbb{V} \uplus \mathbb{H}_{\neq}$. Its *terms* are u_0, u_1.

$u_0 {\neq_s} u_1$ where $u_0, u_1 \in \mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s}$ and $s \in \mathbb{V} \uplus \mathbb{H}_{\neq}$.
This literal is *conjugatible* to its *conjugate* $u_0 {=_s} u_1$. Its *terms* are u_0, u_1.

$p(u_0, \ldots, u_{n-1})$
with $s = \mathsf{pred}$, $p \in \mathbb{K}$ with $p : s'_0, \ldots, s'_{n-1} \rightsquigarrow \mathsf{pred}$, and $\forall i \prec n.\ u_i \in \mathcal{T}(\Sigma, \mathrm{X})_{\mathrm{GEN},s'_i}$.
This literal is not *conjugatible* and its *terms* are $u_0, \ldots, u_{n-1}$.

The sort indices are omitted when clear from the context. The conjugate of a conjugatible condition literal is denoted by $\overline{L}$.

Let $H \subseteq \mathbb{H}$ be the equivalence class of hidden sorts whose data universes we want to establish and O the set of sorts from $\mathbb{S}$ whose data universes we assume to be already established because they are smaller than the sorts in H w.r.t. $<$.

We first add some more slots to our signature objects. We assume that each signature Σ contains a set $\mathbb{D} \subseteq \mathbb{K}\backslash\mathbb{C}$ of *destructors*, a function $\bullet$ with $\mathrm{dom}(\bullet) = \mathbb{H}$ and $\bullet_h \in \mathrm{V}_{\mathrm{GEN},h}$ for all $h \in \mathbb{H}$, and a set $\mathbb{P}$ of *destructor patterns*.

Definition 6.3 (Destructor Pattern)
Each *destructor pattern* from $\mathbb{P}$ has the form $d(x_0, \ldots, x_n) \longleftarrow L_0, \ldots, L_{m-1}$ where $n, m \in \mathbf{N}$ (for $m = 0$ we omit the '$\longleftarrow$'); $d \in \mathbb{D}$ is a destructor with $d : s_0, \ldots, s_n \rightsquigarrow s_{n+1}$; $x_0, \ldots, x_n \in \mathrm{V}_{\mathrm{DAT}}$ are mutually different data variables; and there is an $i \preceq n$ with $\mathrm{ran}(\bullet) \cap \{x_0, \ldots, x_n\} = \{\bullet_{s_i}\}$, $s_{n+1} = \mathsf{pred} \Rightarrow s_i \in \mathbb{H}_{\mathsf{pred}}$, and $\forall k \preceq n{+}1.\ s_k \lesssim s_i$, and for each $j \prec m$, L_j is a conjugatable condition literal over the constructors and destructors for sorts from $O \uplus H$ and over $x_0, \ldots, x_n$ and of a sort from $O \uplus \{\mathsf{pred}\}$. In this case, we speak of a destructor pattern *for s_i and of sort s_{n+1}*, denote the set of these patterns with $\mathbb{P}_{s_i}^{s_{n+1}}$, and call d a destructor *for s_i and of sort s_{n+1}*.

While a destructor pattern can be for only one sort, a destructor can be for different sorts (in different patterns), but we require that these sorts are congruent w.r.t. $\sim$.

Note that the wellfounded quasi-ordering $\lesssim$ guarantees that visible sorts are not constructed from hidden sorts whose behavior they define, which would be a vicious circle. Thus, the result sort of a constructor must be visible and its argument sorts must be smaller, whereas the hidden argument sorts of a destructor must be somehow bigger than its result sort, the sorts of its arguments, and the sorts of the condition literals of its destructor patterns. Note that in the absence of hidden sorts we can choose $\lesssim$ to be $\mathbb{V} \times \mathbb{V}$. This renders the conditions imposed on a signature to be satisfied, so that our signatures really are a generalization of the standard constructor-based notion of a sensible (or inhabited) many-sorted signature to one with hidden sorts and destructors.

Assume that $\mathcal{D}$ assigns to each sort in O and its constructors the already established universes and total functions. Moreover, assume that $\mathcal{D}$ assigns to any destructor for hidden sorts in O already established partial functions. The step we have to perform is to let $\mathcal{D}$ assign a data universe to each sort in H and a partial function to each destructor for a sort in H. Suppose we have a candidate $\mathcal{D}$ for such an assignment.

Firstly, we need three requirements for $\mathcal{D}$. The first is needed soon, the other two must be formulated later.

Requirement 6.4 (Definedness)
For each destructor pattern $\big(\ d(x_0, \ldots, x_n) \longleftarrow L_0, \ldots, L_{m-1}\ \big) \in \mathbb{P}_h^s$ $(h \in H)$ *we require that for each valuation* $\tau : \{x_0, \ldots, x_n\} \to \mathcal{D}$,

- *for each* $j \prec m$ *the terms of* L_j *are defined w.r.t.* $\mathcal{D} \uplus \tau$ *if the literals* $L_0, \ldots, L_{j-1}$ *are true w.r.t.* $\mathcal{D} \uplus \tau$ *and*
- $d(x_0, \ldots, x_n)$ *is defined w.r.t.* $\mathcal{D} \uplus \tau$ *if the literals* $L_0, \ldots, L_{m-1}$ *are true w.r.t.* $\mathcal{D} \uplus \tau$ *and* $s \neq \mathsf{pred}$.

Secondly, we define the following:

Definition 6.5 (Bisimulation)
A $\mathbb{P}$-*bisimulation for* $\mathcal{D}$ is an $O \uplus H$-sorted family $\approx \; = \; (\approx_s)_{s \in O \uplus H}$ of binary relations $\approx_s$ on $s^{\mathcal{D}}$ that has the following properties:

1. For $o \in O$ we have $a \approx_o a'$ iff $a = a' \in o^{\mathcal{D}}$.

2. For $h \in H$, if $a \approx_h a'$ and $\left(\; t \longleftarrow \; L_0, \ldots, L_{m-1} \; \right) \in \mathbb{P}_h^s$, for $Y := \mathcal{V}(t) \backslash \{\bullet_h\}$ and $\forall y \in Y.\; y \in \mathrm{V}_{\mathrm{DAT}, s'_y}$,

   if $s \neq \mathsf{pred}$, **then**
   - for any $\pi : Y \to \mathcal{D}$ such that
     all literals in $L_0, \ldots, L_{m-1}$ are true w.r.t. $\mathcal{D} \uplus \{\bullet_h \mapsto a\} \uplus \pi$
     there is a $\pi' : Y \to \mathcal{D}$ with $\forall y \in Y.\; \pi(y) \approx_{s'_y} \pi'(y)$ such that
     all literals in $L_0, \ldots, L_{m-1}$ are true w.r.t. $\mathcal{D} \uplus \{\bullet_h \mapsto a'\} \uplus \pi'$
     and $\mathrm{eval}(\mathcal{D} \uplus \{\bullet_h \mapsto a\} \uplus \pi)(t) \approx_s \mathrm{eval}(\mathcal{D} \uplus \{\bullet_h \mapsto a'\} \uplus \pi')(t)$, and
   - for any $\pi' : Y \to \mathcal{D}$ such that
     all literals in $L_0, \ldots, L_{m-1}$ are true w.r.t. $\mathcal{D} \uplus \{\bullet_h \mapsto a'\} \uplus \pi'$
     there is a $\pi : Y \to \mathcal{D}$ with $\forall y \in Y.\; \pi(y) \approx_{s'_y} \pi'(y)$ such that
     all literals in $L_0, \ldots, L_{m-1}$ are true w.r.t. $\mathcal{D} \uplus \{\bullet_h \mapsto a\} \uplus \pi$
     and $\mathrm{eval}(\mathcal{D} \uplus \{\bullet_h \mapsto a\} \uplus \pi)(t) \approx_s \mathrm{eval}(\mathcal{D} \uplus \{\bullet_h \mapsto a'\} \uplus \pi')(t)$, and

   if $s = \mathsf{pred}$, **then**
   - for any $\pi : Y \to \mathcal{D}$ such that
     all literals in $L_0, \ldots, L_{m-1}, t$ are true w.r.t. $\mathcal{D} \uplus \{\bullet_h \mapsto a\} \uplus \pi$
     there is a $\pi' : Y \to \mathcal{D}$ with $\forall y \in Y.\; \pi(y) \approx_{s'_y} \pi'(y)$ such that
     all literals in $L_0, \ldots, L_{m-1}, t$ are true w.r.t. $\mathcal{D} \uplus \{\bullet_h \mapsto a'\} \uplus \pi'$, and
   - for any $\pi' : Y \to \mathcal{D}$ such that
     all literals in $L_0, \ldots, L_{m-1}, t$ are true w.r.t. $\mathcal{D} \uplus \{\bullet_h \mapsto a'\} \uplus \pi'$
     there is a $\pi : Y \to \mathcal{D}$ with $\forall y \in Y.\; \pi(y) \approx_{s'_y} \pi'(y)$ such that
     all literals in $L_0, \ldots, L_{m-1}, t$ are true w.r.t. $\mathcal{D} \uplus \{\bullet_h \mapsto a\} \uplus \pi$.

Definition 6.6 (Experiment)
Let Σ be a signature. For $h \in \mathbb{H}$ and $s \in \mathbb{V} \uplus \{\mathsf{pred}\}$ the sets $\mathbb{E}_h^s$ of *experiments over* Σ *for* h *of sort* s are inductively defined as follows:

1. $\mathbb{P}_h^s \subseteq \mathbb{E}_h^s$.
2. If, for $h' \in \mathbb{H}$,
   $\left(\; e \longleftarrow \; L_0, \ldots, L_{m-1} \; \right) \in \mathbb{E}_{h'}^s$,
   $\left(\; e' \longleftarrow \; L'_0, \ldots, L'_{m'-1} \; \right) \in \mathbb{P}_h^{h'}$,
   $\sigma : \mathcal{V}(e) \to \mathcal{T}(\Sigma, \mathrm{V})$ satisfies $\sigma(\bullet_{h'}) = e'$ and renames $\mathcal{V}(e) \setminus \{\bullet_{h'}\}$ away from $\mathcal{V}(e')$ and $\mathrm{ran}(\bullet)$ (i.e. ${}_{\mathcal{V}(e) \backslash \{\bullet_{h'}\}}\!\upharpoonright\!\sigma : (\mathcal{V}(e) \backslash \{\bullet_{h'}\}) \to (\mathrm{V} \backslash \mathcal{V}(e') \backslash \mathrm{ran}(\bullet))$ is injective), then
   $\left(\; e\sigma \longleftarrow \; L'_0, \ldots, L'_{m'-1}, L_0\sigma, \ldots, L_{m-1}\sigma \; \right) \in \mathbb{E}_h^s$.

Note that for any experiment $e \longleftarrow \; L_0, \ldots, L_{m-1}$ for h of sort s the following holds: $s = \mathsf{pred} \Rightarrow h \in \mathbb{H}_{\mathsf{pred}}$, the term e is linear, $\mathcal{V}(L_0, \ldots, L_{m-1}) \subseteq \mathcal{V}(e)$, and there is a unique path from the root of e to the variable $\bullet_h$. On this path, all proper subterms are hidden and all symbols are destructors, whereas all symbols not on this path are variables. Furthermore, $s<h$ and $l_j<h$ for the sorts l_j of L_j.

Now, an $\mathbb{E}$-*bisimulation* can be defined just like a $\mathbb{P}$-bisimulation in Definition 6.5 and the following corollary holds for $\mathbb{P}$- as well as for $\mathbb{E}$-bisimulations.

Corollary 6.7
Let $\mathcal{D}$ satisfy Requirement 6.4. Now:

1. $({}_{s^{\mathcal{D}}}\!\upharpoonright\!\mathrm{id})_{s\in O\uplus H}$ *is a bisimulation for $\mathcal{D}$.*
2. *If $(A_s)_{s\in O\uplus H}$ and $(B_s)_{s\in O\uplus H}$ are bisimulations for $\mathcal{D}$, then $({A_s}^{-1})_{s\in O\uplus H}$ and $(A_s\circ B_s)_{s\in O\uplus H}$ are bisimulations for $\mathcal{D}$, too.*
3. *A non-empty union over bisimulations for $\mathcal{D}$ is a bisimulation for $\mathcal{D}$.*
4. *There is a maximum bisimulation for $\mathcal{D}$, which is given as the union over all bisimulations for $\mathcal{D}$ and which is a $\mathcal{D}$-equivalence.*

Corollary 6.8 *If $\mathcal{D}$ satisfies Requirement 6.4, then every $\mathbb{P}$-bisimulation is an $\mathbb{E}$-bisimulation.*

As the maximum $\mathbb{E}$-bisimulation can easily be shown to satisfy the contrapositive of condition (2) of the definition of a $\mathbb{P}$-bisimulation, we get:

Corollary 6.9 *If $\mathcal{D}$ satisfies Requirement 6.4, then the maximum $\mathbb{E}$-bisimulation for $\mathcal{D}$ is the maximum $\mathbb{P}$-bisimulation for $\mathcal{D}$ and will be denoted with $\approx_{\mathcal{D}}$.*

The second requirement on $\mathcal{D}$ is the opposite of "no confusion":

Requirement 6.10 (Extensionality)
For any two elements $a, a' \in h^{\mathcal{D}}$ of a data universe for a sort $h \in H$, if $a \approx_{\mathcal{D},h} a'$ then $a = a'$.

The third requirement on $\mathcal{D}$ is quite the opposite of "no junk":

Requirement 6.11 (Maximality)
If we extend the data universes for H and the partial functions for the destructors for H, yielding a new structure $\mathcal{D}'$ for that Requirement 6.4 is satisfied, then for each new element $a \in h^{\mathcal{D}'} \setminus h^{\mathcal{D}}$ of a data universe for $h \in H$, we have $a \approx_{\mathcal{D}',h} a'$ for some old element $a' \in h^{\mathcal{D}}$.

Now it is time to see some examples.

Example 6.12 (Infinite Streams)
The infinite streams over some sort s can be established as follows. Let $\mathsf{infstream} \in \mathbb{H}$ with $s < \mathsf{infstream}$ and hd, $\mathsf{tl} \in \mathbb{D}$ with $\mathsf{hd} : \mathsf{infstream} \rightsquigarrow s$ and $\mathsf{tl} : \mathsf{infstream} \rightsquigarrow \mathsf{infstream}$. Set $\mathbb{P} := \{\mathsf{hd}(\bullet_{\mathsf{infstream}}),\ \mathsf{tl}(\bullet_{\mathsf{infstream}})\}$. We get $\mathbb{E} = \{\ \mathsf{hd}(\mathsf{tl}^k(\bullet_{\mathsf{infstream}}))\ \mid\ k \in \mathbf{N}\ \}$.
The above definedness requirement says that $\mathsf{hd}^{\mathcal{D}}$ and $\mathsf{tl}^{\mathcal{D}}$ must be total on $\mathsf{infstream}^{\mathcal{D}}$.
The condition (2) of Definition 6.5 says (due to $Y = \emptyset$) that $a \approx_{\mathcal{D},\mathsf{infstream}} a'$ implies $\mathsf{hd}^{\mathcal{D}}(a) = \mathsf{hd}^{\mathcal{D}}(a')$ and $\mathsf{tl}^{\mathcal{D}}(a) \approx_{\mathcal{D},\mathsf{infstream}} \mathsf{tl}^{\mathcal{D}}(a')$.

Thus, the above extensionality requirement says that the data universe for $\mathsf{infstream}$ can be viewed as a subset of $\mathbb{E} \to s^{\mathcal{D}}$ (with simple changes for more general experiments (cf. Example 6.17), this is always the case if the experiments are unconditional and do not include variables of sorts from H besides $\mathrm{ran}(\bullet)$) or as a subset of $\mathbf{N} \to s^{\mathcal{D}}$. Indeed, the infinite streams are isomorphic to this set of functions.

Finally, the maximality requirement says that we need all functions from $\mathbf{N} \to s^{\mathcal{D}}$ in the data universe for $\mathsf{infstream}$, or at least those we can talk about. By this we mean that we only need those streams that can be comprehended with our language, logic, and (co) recursive definition mechanism. Such a "Henkin" or "general" model can usually be guaranteed to exist, to be countable, and to have the same theory as the standard model (here: of all streams) by a proof similar to the completeness proof for Q_0 in Andrews (1986). While countable data universes may be preferred for philosophical reasons, for infinite streams there seems to be no real need to deviate from the standard model. This will be different in Example 6.16 below.

The co algebraic view on infinite streams is the destructor

$$(\mathsf{hd}, \mathsf{tl}) : \mathsf{infstream} \to s \times \mathsf{infstream}$$

and the set of infinite streams is the greatest fixpoint of the monotonic functor $F(X) = s^{\mathcal{D}} \times X$ in the non-wellfounded set theory of Aczel (1988).

Example 6.13 (Streams)

The (finite and infinite) streams over some sort s can be established as follows. Let $\mathsf{stream} \in \mathbb{H}$ with $s < \mathsf{stream}$ and null, hd, tl $\in$ $\mathbb{D}$ with $\mathsf{null} : \mathsf{stream} \rightsquigarrow \mathsf{bool}$, $\mathsf{hd} : \mathsf{stream} \rightsquigarrow s$, and $\mathsf{tl} : \mathsf{stream} \rightsquigarrow \mathsf{stream}$. Set

$$\mathbb{P} := \left\{ \begin{array}{lcl} \mathsf{null}(\bullet_{\mathsf{stream}}); & & \\ \mathsf{hd}(\bullet_{\mathsf{stream}}) & \longleftarrow & \mathsf{null}(\bullet_{\mathsf{stream}}){=}\mathsf{false}; \\ \mathsf{tl}(\bullet_{\mathsf{stream}}) & \longleftarrow & \mathsf{null}(\bullet_{\mathsf{stream}}){=}\mathsf{false} \end{array} \right\}.$$

We get $\mathbb{E} :=$

$$\left\{ \begin{array}{lcl} \mathsf{null}(\mathsf{tl}^k(\bullet_{\mathsf{stream}})) & \longleftarrow & \mathsf{null}(\mathsf{tl}^0(\bullet_{\mathsf{stream}})){=}\mathsf{false},\ \ldots,\ \mathsf{null}(\mathsf{tl}^{k-1}(\bullet_{\mathsf{stream}})){=}\mathsf{false}; \\ \mathsf{hd}(\mathsf{tl}^k(\bullet_{\mathsf{stream}})) & \longleftarrow & \mathsf{null}(\mathsf{tl}^0(\bullet_{\mathsf{stream}})){=}\mathsf{false},\ \ldots,\ \mathsf{null}(\mathsf{tl}^{k}(\bullet_{\mathsf{stream}})){=}\mathsf{false} \end{array} \middle|\ k \in \mathbf{N} \right\}.$$

The above definedness requirement says that $\mathsf{null}^{\mathcal{D}}$ must be total on $\mathsf{stream}^{\mathcal{D}}$ and that $\mathsf{hd}^{\mathcal{D}}$ and $\mathsf{tl}^{\mathcal{D}}$ must be defined on any non-null stream.

The condition (2) of Definition 6.5 says that $a \approx_{\mathcal{D},\mathsf{stream}} a'$ implies $\mathsf{null}^{\mathcal{D}}(a) = \mathsf{null}^{\mathcal{D}}(a')$ and if $\mathsf{null}^{\mathcal{D}}(a) = \mathsf{false}^{\mathcal{D}}$ then $\mathsf{hd}^{\mathcal{D}}(a) = \mathsf{hd}^{\mathcal{D}}(a')$ and $\mathsf{tl}^{\mathcal{D}}(a) \approx_{\mathcal{D},\mathsf{stream}} \mathsf{tl}^{\mathcal{D}}(a')$.

Thus, the above extensionality requirement says that the data universe for stream can be viewed as a subset of $\bigcup_{\alpha \preceq \omega} (\alpha \to s^{\mathcal{D}})$. Indeed, the streams are isomorphic to this set of functions.

Finally, the maximality requirement says that we need all functions from this set in the data universe for stream, or at least those we can talk about.

The co algebraic view on infinite streams is the destructor

$$(\mathsf{hd}, \mathsf{tl}) : \mathsf{stream} \to s \times (\mathsf{stream} + 1)$$

and the set of streams minus the empty stream is the greatest fixpoint of the monotonic functor $F(X) = s^{\mathcal{D}} \times (X \uplus 1)$ in the non-wellfounded set theory of Aczel (1988).

Example 6.14 (Characteristic Functions)
The characteristic functions over some sort s can be established as follows. Let $\mathsf{chara} \in \mathbb{H}$ with $s < \mathsf{chara}$ and $\mathsf{in} \in \mathbb{D}$ with $\mathsf{in} : s, \mathsf{chara} \rightsquigarrow \mathsf{bool}$.
Set $\mathbb{P} := \{\mathsf{in}(e, \bullet_{\mathsf{chara}})\}$. We get $\mathbb{E} = \mathbb{P}$.
The above definedness requirement says that $\mathsf{in}^{\mathcal{D}}$ must be total on $s^{\mathcal{D}} \times \mathsf{chara}^{\mathcal{D}}$.
The condition (2) of Definition 6.5 says that $a \approx_{\mathcal{D},\mathsf{chara}} a'$ implies $\mathsf{in}^{\mathcal{D}}(e, a) = \mathsf{in}^{\mathcal{D}}(e, a')$ for any $e \in s^{\mathcal{D}}$.
Thus, the above extensionality requirement says that the data universe for chara can be viewed as a subset of $s^{\mathcal{D}} \to \mathsf{bool}^{\mathcal{D}}$. Indeed, the characteristic functions are isomorphic to this set of functions.
Finally, the maximality requirement says that we need all functions from this set in the data universe for chara, or at least those we can talk about.
The co algebraic view on characteristic functions is the destructor $\mathsf{in} : \mathsf{chara} \to s \to 2$ and the set of characteristic functions is the greatest fixpoint of the monotonic functor $F(X) = s^{\mathcal{D}} \to \mathsf{bool}^{\mathcal{D}}$.

Example 6.15 (Sets)
Some people would say that the characteristic functions of Example 6.14 are isomorphic to predicates or sets. Other people, however, use to think predicates as μ-predicates, i.e. sets where only positive information is given and the intended meaning is the smallest consistent set. This will be important for the labeled transition systems in the example below. We will call these μ-predicates simply “sets” in the current example.
The sets over some sort s can be established as follows. Let $\mathsf{set} \in \mathbb{H}_{\mathsf{pred}}$ with $s < \mathsf{set}$ and $\mathsf{in} \in \mathbb{D}$ with $\mathsf{in} : s, \mathsf{set} \rightsquigarrow \mathsf{pred}$.
Set $\mathbb{P} := \{\mathsf{in}(e, \bullet_{\mathsf{set}})\}$. We get $\mathbb{E} = \mathbb{P}$.
Note that we cannot put set into $\mathbb{H}_{\neq}$ because we have a pattern of the sort pred. Thus, we have to put it into $\mathbb{H}_{\mathsf{pred}}$ instead, which means that there are no inequality condition literals for the sort pred.
The above definedness requirement does not say anything about the definedness of $\mathsf{in}^{\mathcal{D}}$. Definedness of predicates is no reasonable notion anyway.
The condition (2) of Definition 6.5 says that $a \approx_{\mathcal{D},\mathsf{set}} a'$ implies, for any $e \in s^{\mathcal{D}}$, that $\mathsf{in}^{\mathcal{D}}(e, a)$ iff $\mathsf{in}^{\mathcal{D}}(e, a')$.
Thus, the above extensionality requirement says that the data universe for set can be viewed as a subset of $\mathfrak{P}(s^{\mathcal{D}})$ or as a subset of $s^{\mathcal{D}} \rightsquigarrow \{\mathsf{true}^{\mathcal{D}}\}$. Indeed, the sets over $s^{\mathcal{D}}$ are isomorphic to any of these sets. Moreover, they are even isomorphic to $s^{\mathcal{D}} \to \mathsf{bool}^{\mathcal{D}}$ (i.e. $s^{\mathcal{D}} \to \{\mathsf{true}^{\mathcal{D}}, \mathsf{false}^{\mathcal{D}}\}$) of Example 6.14, which seems to justify the view that characteristic functions and sets are isomorphic. The last isomorphism, however, depends on two premises that both will not be satisfied in Example 6.16, namely that $s^{\mathcal{D}}$ is a set (i.e. not a proper class) and that $s < \mathsf{set}$.
Finally, the maximality requirement says that we need all elements from one of these sets in the data universe for set, or at least those we can talk about.
The co algebraic view on sets is the destructor $\mathsf{in} : \mathsf{set} \to \mathfrak{P}(s)$ and the set of sets over $s^{\mathcal{D}}$ is the greatest fixpoint of the monotonic functor $F(X) = \mathfrak{P}(s^{\mathcal{D}})$.

Example 6.16 (Labeled Transition Systems)

The labeled transition systems over bool as alphabet can be established as follows. Let $\mathsf{lts},\ \mathsf{set} \in \mathbb{H}_{\mathsf{pred}}$ with $\mathsf{bool} < \mathsf{lts} \sim \mathsf{set}$ and $\mathsf{next},\ \mathsf{in} \in \mathbb{D}$ with $\mathsf{next} : \mathsf{lts}, \mathsf{bool} \rightsquigarrow \mathsf{set}$ and $\mathsf{in} : \mathsf{lts}, \mathsf{set} \rightsquigarrow \mathsf{pred}$. Set $\mathbb{P} := \{\mathsf{next}(\bullet_{\mathsf{lts}}, b),\ \mathsf{in}(a, \bullet_{\mathsf{set}})\}$, where the former returns the set of all successor states labeled with b and the latter tests whether a is in the set. We get $\mathbb{E} = \{\mathsf{in}(a, \mathsf{next}(\bullet_{\mathsf{lts}}, b)),\ \mathsf{in}(a, \bullet_{\mathsf{set}})\}$.
The above definedness requirement says that $\mathsf{next}^{\mathcal{D}}$ must be total on $\mathsf{lts}^{\mathcal{D}}$.
The condition (2) of Definition 6.5 says that $a \approx_{\mathcal{D},\mathsf{lts}} a'$ implies $\mathsf{next}^{\mathcal{D}}(a,b) \approx_{\mathcal{D},\mathsf{set}} \mathsf{next}^{\mathcal{D}}(a',b)$ for all $b \in \mathsf{bool}^{\mathcal{D}}$. It also says that $s \approx_{\mathcal{D},\mathsf{set}} s'$ implies that for any $a \in \mathsf{lts}^{\mathcal{D}}$ with $\mathsf{in}^{\mathcal{D}}(a,s)$ there is an $a' \in \mathsf{lts}^{\mathcal{D}}$ with $a \approx_{\mathsf{lts}} a'$ and $\mathsf{in}^{\mathcal{D}}(a',s')$.

The co algebraic view on infinite streams is the destructor $\mathsf{next} : \mathsf{lts} \to \mathsf{bool} \to \mathfrak{P}(\mathsf{lts})$ and the proper class of labeled transition systems is the greatest fixpoint of the monotonic functor $F(X) = \mathsf{bool}^{\mathcal{D}} \to \mathfrak{P}(X)$ in the non-wellfounded set theory of Aczel (1988). We, however, have two problems with this fixpoint:
The first problem is that it is a proper class. This means we cannot work properly with it in set theory. E.g., every total function defined on it is a proper class, too, and thus any class of such functions is empty.
The second problem is that for new nullary function symbols say $\mathsf{a} : \mathsf{lts}$ and $\mathsf{all} : \mathsf{set}$ defined by $\mathsf{next}(\mathsf{a}, b) = \mathsf{all}$ and $\mathsf{in}(a, \mathsf{all})$ we have the problem that $\mathsf{a}^{\mathcal{D}}$ is not bisimular to any element of that fixpoint, simply because $\mathsf{all}^{\mathcal{D}}$ functions as a proper class and therefore cannot be an element of that fixpoint. This means that that fixpoint does not satisfy our above maximality requirement.
Thus, we need another fixpoint (1) in another set theory or (2) for another functor.

1. In order to be consistent, a set theory has to put some restrictions on set formation in order to avoid sets like the famous class named after Bertrand Russell, which is similar to the one below. The ingenious solution to this problem found by John von Neumann and others had been sufficient for half a century but then turned out to be not too well-suited for some tasks like ours here. While the non-wellfounded set theory of Aczel (1988) is simply an improvement of the standard one (i.e. it is just as consistent but has more sets), other set theories might deprive us of certain sets by restricting the axioms of comprehension, power-set, union, separation, subset, &c., but give us new sets that may be important for our work. The only results on new set theories I know are Forti &al. (1996) and others of the same authors who do not seem to have any interest in the practical usefulness of their theories or in explaining them to anybody interested in their practical application. It should be clear that we cannot settle these very sophisticated problems here.

2. The clearer solution is to choose a different functor. E.g., in this example, we could use the functor
$$F(X) = \{\ f : \mathsf{bool}^{\mathcal{D}} \to \mathfrak{P}(X) \mid f \text{ is “definable” with the elements of } X\ \},$$
which is finitary and monotonic and thus will have a least fixpoint at level ω, which is countable and therefore a set. While this is not the place to define “definable”, the reader should also cf. the discussion on “Henkin” or “general” models in Example 6.12.

Note that in general it may be well the case that no such fixpoint exists in any set theory. To see this, suppose that the signature of in would be $\mathsf{in} : \mathsf{lts}, \mathsf{set} \rightsquigarrow \mathsf{bool}$. Now, consider $\mathsf{next}(\mathsf{r},b)=\mathsf{Russell}$ and $\mathsf{in}(a,\mathsf{Russell})=\mathsf{true}\longleftarrow\ \mathsf{in}(a,\mathsf{next}(a,\mathsf{true}))=\mathsf{false}$. This implies $\mathsf{in}(\mathsf{r},\mathsf{next}(\mathsf{r},\mathsf{true}))=\mathsf{true}\longleftarrow\ \mathsf{in}(\mathsf{r},\mathsf{next}(\mathsf{r},\mathsf{true}))=\mathsf{false}$. Since the destructor pattern for in would require totality of $\mathsf{in}^{\mathcal{D}}$, this would lead to a contradiction. Thus, in this case, there would be no data universe that satisfies our definedness and maximality requirements, unless we excluded the above definition of r from the admissible ones, which renders r "undefinable". But of course, this destructor pattern would have completely different effects: Consider

$$\begin{tikzcd} \mathsf{a} \arrow[d, bend right, "\mathsf{true}"'] \arrow[rrrd, "\mathsf{true}"] & & & \mathsf{c} \arrow[d, bend right, "\mathsf{true}"'] \\ \mathsf{b} \arrow[u, bend right, "\mathsf{false}"'] & & & \mathsf{d} \arrow[u, bend right, "\mathsf{false}"'] \end{tikzcd}$$

which can be specified via

$$\begin{array}{ccc} \mathsf{in}(\mathsf{b},\mathsf{next}(\mathsf{a},\mathsf{true})) & & \mathsf{in}(\mathsf{d},\mathsf{next}(\mathsf{c},\mathsf{true})) \\ & \mathsf{in}(\mathsf{d},\mathsf{next}(\mathsf{a},\mathsf{true})) & \\ \mathsf{in}(\mathsf{a},\mathsf{next}(\mathsf{b},\mathsf{false})) & & \mathsf{in}(\mathsf{c},\mathsf{next}(\mathsf{d},\mathsf{false})) \end{array}$$

and renders $\mathsf{a}\approx_{\mathcal{D},\mathsf{lts}}\mathsf{c}$ and $\mathsf{b}\approx_{\mathcal{D},\mathsf{lts}}\mathsf{d}$. For the alternative signature of $\mathsf{in}:\mathsf{lts},\mathsf{set}\rightsquigarrow\mathsf{bool}$, however, the above specification would not be sufficient because in would have to be total, and we would need something like:

$$\begin{array}{l} \mathsf{in}(a,\mathsf{next}(\mathsf{a},b))=\mathsf{and}(\mathsf{or}(\mathsf{eq}(a,\mathsf{b}),\mathsf{eq}(a,\mathsf{d})),\mathsf{eq}(b,\mathsf{true})) \\ \mathsf{in}(a,\mathsf{next}(\mathsf{b},b))=\mathsf{and}(\mathsf{eq}(a,\mathsf{a}),\mathsf{eq}(b,\mathsf{false})) \\ \mathsf{in}(a,\mathsf{next}(\mathsf{c},b))=\mathsf{and}(\mathsf{eq}(a,\mathsf{d}),\mathsf{eq}(b,\mathsf{true})) \\ \mathsf{in}(a,\mathsf{next}(\mathsf{d},b))=\mathsf{and}(\mathsf{eq}(a,\mathsf{c}),\mathsf{eq}(b,\mathsf{false})) \end{array}$$

But then the maximum bisimulation would be the identity because in case of $\mathsf{a}\approx_{\mathcal{D},\mathsf{lts}}\mathsf{c}$, due to $\mathsf{in}(\mathsf{b},\mathsf{next}(\mathsf{c},\mathsf{true}))=\mathsf{false}$ there must be some $a\approx_{\mathcal{D},\mathsf{lts}}\mathsf{b}$ with $\mathsf{in}(a,\mathsf{next}(\mathsf{a},\mathsf{true}))=\mathsf{false}$, but the only possible solutions $a=\mathsf{a}$ and $a=\mathsf{c}$ are contradictory.

Example 6.17 (Total Functions)

The simple function spaces of simply typed higher-order logic can be established stepwise in the following fashion. For $\alpha,\beta\in\mathbb{S}$, in order to get the total functions from α to β, let $(\alpha\rightarrow\beta)\in\mathbb{H}$ with $\alpha,\beta<(\alpha\rightarrow\beta)$ and $\mathsf{app}\in\mathbb{D}$ with $\mathsf{app}:(\alpha\rightarrow\beta),\alpha\rightsquigarrow\beta$. Set $\mathbb{P}:=\{\mathsf{app}(\bullet_{(\alpha\rightarrow\beta)},a)\}$. We get $\mathbb{E}=\mathbb{P}$.
The above definedness requirement says that $\mathsf{app}^{\mathcal{D}}$ and must be total on $(\alpha\rightarrow\beta)^{\mathcal{D}}\times\alpha^{\mathcal{D}}$.
The condition (2) of Definition 6.5 says (due to $Y=\emptyset$) that $f\ \approx_{\mathcal{D},(\alpha\rightarrow\beta)}\ f'$ implies $\mathsf{app}^{\mathcal{D}}(f,a)=\mathsf{app}^{\mathcal{D}}(f',a)$ for any $a\in\alpha^{\mathcal{D}}$. Moreover, the extensional equality of higher-order logic is—mutatis mutandis—exactly our $\approx_{\mathcal{D},(\alpha\rightarrow\beta)}$.
Now, the above extensionality requirement says that the data universe for $(\alpha\rightarrow\beta)$ can be viewed as a subset of $\{\ \mathsf{app}(\bullet_{(\alpha\rightarrow\beta)},a)\mid a\in\alpha^{\mathcal{D}}\ \}\rightarrow\beta^{\mathcal{D}}$ or as a subset of $\alpha^{\mathcal{D}}\rightarrow\beta^{\mathcal{D}}$. Indeed, the total functions from α to β are isomorphic to these sets of functions.
Finally, the maximality requirement says that we need all functions from $\alpha^{\mathcal{D}}\rightarrow\beta^{\mathcal{D}}$ in the data universe for $(\alpha\rightarrow\beta)$, or at least those we can talk about.
The co algebraic view on infinite streams is the destructor

$$\mathsf{app}:(\alpha\rightarrow\beta)\rightarrow\alpha\rightarrow\beta$$

and the set of total functions is the greatest fixpoint of the monotonic functor $F(X)=\alpha^{\mathcal{D}}\rightarrow\beta^{\mathcal{D}}$.

Our pragmatic solution to the problems with the establishment of hidden sorts is simply to assume that for the destructor patterns and the admissible definitions we want to have there is a set satisfying our three requirements of definedness, extensionality, and maximality from above, and that there is a way to choose one of them as the intended one. In case that this choice should—contrary to our assumptions—turn out to be crucial for our notions of validity, then we have to add more requirements, and finally, when the field has become more common and better understood, we will have to solve the problem of whether and what data universes exist for given destructor patterns and given admissibility of definitions by some general means.

7 Fulfilledness and Rewriting

Note that the following definition is recursive on the sort of condition literals in the well-founded ordering $<$ introduced at the beginning of § 6.

Definition 7.1 (Fulfilledness)
Let Σ be a signature with $\mathbb{S}$, $\mathbb{V}$, $\mathbb{H}$, $\mathbb{K}$, $\mathbb{C}$ as sets of sorts, visible sorts, hidden sorts, constant symbols, constructors, resp.. Let $\mathcal{D}$ be the data reduct of some Σ-algebra. Let X be a variable-system for $\mathbb{S}$, $\mathbb{K}$ with $\forall s\in\mathbb{S}.\ \mathrm{X}_{\mathrm{DAT},s}=\emptyset$. Let Σ' denote the signature that differs from Σ insofar as that the sets $\mathbb{K}$ and $\mathbb{C}$ of constants and constructors are extended to the sets $\mathbb{K}'$ and $\mathbb{C}'$ with a new nullary constant symbol $d:\ \to h$ for any hidden sort $h\in\mathbb{H}$ and any $d\in h^{\mathcal{D}}$. Note that the signature Σ' violates the rules of § 6 because it has constructors for hidden sorts. Let $\longrightarrow$ be a sort-invariant relation on $\mathcal{T}(\Sigma',\mathrm{X})$ and H be a set of condition literals of sort pred over Σ',X.
A list of condition literals over Σ',X is said to be *fulfilled w.r.t.* $\longrightarrow, H$ if each of its literals is.
A condition literal over Σ',X is said to be *fulfilled w.r.t.* $\longrightarrow, H$ if for all $s\in\mathbb{S}$ and $u,u'\in\mathcal{T}(\Sigma',\mathrm{X})_{\mathrm{GEN},s}$ if it has the form

Def u**:**

If $s\in\mathbb{V}$**:** There is a term t whose function symbols are constructors from $\mathbb{C}'$ and whose variables are of hidden sort, and a substitution $\sigma:\mathcal{V}(t)\to\mathcal{T}(\Sigma',\mathrm{X})$ such that $u\overset{*}{\longrightarrow}t\sigma$ and $\mathrm{Def}(x\sigma)$ is fulfilled w.r.t. $\longrightarrow, H$ for each $x\in\mathcal{V}(t)$.

If $s\in\mathbb{H}$**:** For any $v\in\mathbb{V}$, any experiment $\left(\ e\longleftarrow\ L_0,\ldots,L_{m-1}\ \right)\in\mathbb{E}^v_s$ over Σ,V where w_j,w'_j are the terms of L_j, and any $\sigma:\mathcal{V}(e)\to\mathcal{T}(\Sigma',\mathrm{X})$ with $\sigma(\bullet_s)=u$ and $\forall x\in\mathcal{V}(e)\backslash\{\bullet_s\}.\ \sigma(x)\in\mathcal{GT}(\Sigma')_{\mathrm{DAT}}$, there either is some $i\prec m$ such that
$\left(\ \mathrm{Def}\ w_0,\mathrm{Def}\ w'_0,L_0,\ldots,\mathrm{Def}\ w_{i-1},\mathrm{Def}\ w'_{i-1},L_{i-1},\mathrm{Def}\ w_i,\mathrm{Def}\ w'_i,\overline{L_i}\ \right)\sigma$
is fulfilled w.r.t. $\longrightarrow, H$ or otherwise
$\left(\ \mathrm{Def}\ w_0,\mathrm{Def}\ w'_0,L_0,\ldots,\mathrm{Def}\ w_{m-1},\mathrm{Def}\ w'_{m-1},L_{m-1}\ \right)\sigma$
is fulfilled w.r.t. $\longrightarrow, H$ and, unless $v=\mathsf{pred}$, $\mathrm{Def}(e\sigma)$ is fulfilled w.r.t. $\longrightarrow, H$.

$u{=}u'$**:**

If $s\in\mathbb{V}$**:** There is a term t whose function symbols are constructors from $\mathbb{C}'$ and whose variables are of hidden sort, and two substitutions $\sigma,\sigma':\mathcal{V}(t)\to\mathcal{T}(\Sigma',\mathrm{X})$ such that $u\overset{*}{\longrightarrow}t\sigma$, $u'\overset{*}{\longrightarrow}t\sigma'$, and $x\sigma{=}x\sigma'$ is fulfilled w.r.t. $\longrightarrow, H$ for each $x\in\mathcal{V}(t)$.

If $s\in\mathbb{H}$**:** For any $v\in\mathbb{V}$, any experiment $\left(\ e\longleftarrow\ L_0,\ldots,L_{m-1}\ \right)\in\mathbb{E}^v_s$ over Σ,V where w_j,w'_j are the terms of L_j, and any $\sigma,\sigma':\mathcal{V}(e)\to\mathcal{T}(\Sigma',\mathrm{X})$ with $\sigma(\bullet_s)=u$, $\sigma'(\bullet_s)=u'$, and $\forall x\in\mathcal{V}(e)\backslash\{\bullet_s\}.\ \sigma(x)=\sigma'(x)\in\mathcal{GT}(\Sigma')_{\mathrm{DAT}}$, there either is some $i\prec m$ such that
$\left(\ \mathrm{Def}\ w_0,\mathrm{Def}\ w'_0,L_0,\ldots,\mathrm{Def}\ w_{i-1},\mathrm{Def}\ w'_{i-1},L_{i-1},\mathrm{Def}\ w_i,\mathrm{Def}\ w'_i,\overline{L_i}\ \right)\sigma$
or
$\left(\ \mathrm{Def}\ w_0,\mathrm{Def}\ w'_0,L_0,\ldots,\mathrm{Def}\ w_{i-1},\mathrm{Def}\ w'_{i-1},L_{i-1},\mathrm{Def}\ w_i,\mathrm{Def}\ w'_i,\overline{L_i}\ \right)\sigma'$
is fulfilled w.r.t. $\longrightarrow, H$ or otherwise
$\left(\ \mathrm{Def}\ w_0,\mathrm{Def}\ w'_0,L_0,\ldots,\mathrm{Def}\ w_{m-1},\mathrm{Def}\ w'_{m-1},L_{m-1}\ \right)\sigma$
and

$\big($ Def w_0, Def $w'_0, L_0, \ldots,$ Def w_{m-1}, Def w'_{m-1}, L_{m-1} $\big)\sigma'$
are fulfilled w.r.t. $\longrightarrow, H$ and, in case of $v \neq \mathsf{pred}$, $e\sigma{=}e\sigma'$ is fulfilled w.r.t. $\longrightarrow, H$, and, in case of $v \neq \mathsf{pred}$, $e\sigma, e\sigma'$ is fulfilled w.r.t. $\longrightarrow, H$.

$u{\neq}u'$**:**

If $s \in \mathbb{V}$**:** There are terms t, t' whose function symbols are constructors from $\mathbb{C}'$ and whose variables are of hidden sort, and two substitutions $\sigma : \mathcal{V}(t) \to \mathcal{T}(\Sigma', \mathrm{X})$, $\sigma' : \mathcal{V}(t') \to \mathcal{T}(\Sigma', \mathrm{X})$, such that $u \overset{*}{\longrightarrow} t\sigma$, $u' \overset{*}{\longrightarrow} t'\sigma'$, and either t and t' are clashing or else t and t' are equal and there is some $x \in \mathcal{V}_{\mathrm{GEN},h}(t)$ such that $h \in \mathbb{H}_{\neq}$ and $x\sigma{\neq}x\sigma'$ is fulfilled w.r.t. $\longrightarrow, H$.

If $s \in \mathbb{H}_{\neq}$**:** For any $v \in \mathbb{V}$, any experiment $\big(\ e \longleftarrow\ L_0, \ldots, L_{m-1}\ \big) \in \mathbb{E}^v_s$ over Σ, V where w_j, w'_j are the terms of L_j, and any $\sigma, \sigma' : \mathcal{V}(e) \to \mathcal{T}(\Sigma', \mathrm{X})$ with $\sigma(\bullet_s) = u$, $\sigma'(\bullet_s) = u'$, and $\forall x \in \mathcal{V}(e) \setminus \{\bullet_s\}.\ \sigma(x) = \sigma'(x) \in \mathcal{GT}(\Sigma')_{\mathrm{DAT}}$, there either is some $i \prec m$ such that
$\big($ Def w_0, Def $w'_0, L_0, \ldots,$ Def w_{i-1}, Def w'_{i-1}, L_{i-1}, Def w_i, Def $w'_i, \overline{L_i}$ $\big)\sigma$
or
$\big($ Def w_0, Def $w'_0, L_0, \ldots,$ Def w_{i-1}, Def w'_{i-1}, L_{i-1}, Def w_i, Def $w'_i, \overline{L_i}$ $\big)\sigma'$
is fulfilled w.r.t. $\longrightarrow, H$ or otherwise
$\big($ Def w_0, Def $w'_0, L_0, \ldots,$ Def w_{m-1}, Def w'_{m-1}, L_{m-1} $\big)\sigma$
and
$\big($ Def w_0, Def $w'_0, L_0, \ldots,$ Def w_{m-1}, Def w'_{m-1}, L_{m-1} $\big)\sigma'$
are fulfilled w.r.t. $\longrightarrow, H$ and (note $v \neq \mathsf{pred}$) $e\sigma{\neq}e\sigma'$ is fulfilled w.r.t. $\longrightarrow, H$.

$p(w_0, \ldots, w_{n-1})$**:** If there is some $p(w'_0, \ldots, w'_{n-1}) \in H$ such that for each $j \prec n$ either w_i is equal to w'_i or $(w_i{=}w'_i)$ is fulfilled w.r.t. $\longrightarrow, H$.

Notes

References

Peter Aczel (1988). *Non-Well-Founded Sets.* CSLI Lecture Notes 14, Stanford Univ..

Peter B. Andrews (1986). *An Introduction to Mathematical Logic and Type Theory.* Academic Press (Elsevier).

Jürgen Avenhaus, Klaus Madlener (1990). *Term Rewriting and Equational Reasoning.* In: R. B. Banerji (ed.). *Formal Techniques in Artificial Intelligence*, pp. 1–43, Elsevier.

Matthias Baaz, Christian G. Fermüller (1995). *Non-elementary Speedups between Different Versions of Tableaux.* 4th TABLEAUX 1995, LNAI 918, pp. 217–230, Springer.

Matthias Baaz, Uwe Egly, Christian G. Fermüller (1997). *Lean Induction Principles for Tableaux.* 6th TABLEAUX 1997, LNAI 1227, pp. 62–75, Springer.

Leo Bachmair (1988). *Proof By Consistency in Equational Theories.* 3rd IEEE Symposium on Logic In Computer Sci., pp. 228–233, IEEE Press.

Peter Baumgartner, Ulrich Furbach, Frieder Stolzenburg (1997). *Computing Answers with Model Elimination.* Artificial Intelligence **90**, pp. 135–176.

Bernhard Beckert, Reiner Hähnle (1998). *Analytic Tableaux.* In: Bibel & Schmitt (1998), Vol. 1, pp. 11–41.

Bernhard Beckert, Reiner Hähnle, Peter H. Schmitt (1993). *The Even More Liberalized δ-Rule in Free Variable Semantic Tableaux.* Kurt Gödel Colloquium, LNCS 713, pp. 108-119, Springer.

Wolfgang Bibel (1987). *Automated Theorem Proving.* 2nd rev. ed., Vieweg, Braunschweig.

Wolfgang Bibel, Peter H. Schmitt (eds.) (1998). *Automated Deduction — A Basis for Applications.* Kluwer (Springer).

Susanne Biundo, Birgit Hummel, Dieter Hutter, Christoph Walther (1986). *The Karlsruhe Induction Theorem Proving System.* 8th CADE 1986, LNCS 230, pp. 672–674, Springer.

Adel Bouhoula, Michaël Rusinowitch (1995). *Implicit Induction in Conditional Theories.* J. Automated Reasoning **14**, pp. 189–235, Kluwer (Springer).

Adel Bouhoula, Emmanuël Kounalis, Michaël Rusinowitch (1992). *Automated Mathematical Induction.* Technical Report 1636, INRIA.

Nicolas Bourbaki (1954). *Livre I — Théorie des Ensembles.* Hermann, Paris.

Robert S. Boyer, J S. Moore (1979). *A Computational Logic.* Academic Press (Elsevier).

Robert S. Boyer, J S. Moore (1988). *A Computational Logic Handbook.* Academic Press (Elsevier).

Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux (1993). *On Skolemization in Constrained Logics.* DFKI Research Report RR–93–06, Saarbrücken. Also in: Annals of Mathematics and Artificial Intelligence **18**, pp. 95-131, Balzer Sci. Publ., 1996.

Ricardo Caferra, Gernot Salzer (eds.) (2000). *Automated Deduction in Classical and Non-Classical Logics.* LNAI 1761, Springer.

Domenico Cantone, Marianna Nicolosi-Asmundo (2000). *A Further and Effective Liberalization of the δ-Rule in Free Variable Semantic Tableaux.* In: Caferra & Salzer (2000), pp. 109–125.

Hartmut Ehrig, Bernd Mahr (1985). *Fundamentals of Algebraic Specification 1.* Springer.

Herbert B. Enderton (1973). *A Mathematical Introduction to Logic.* 2nd printing, Academic Press (Elsevier).

Melvin C. Fitting (1996). *First-Order Logic and Automated Theorem Proving.* 2nd extd. ed., Springer.

Marco Forti, Furio Honsell, Marina Lenisa (1996). *Axiomatic Characterizations of Hyperuniverses and Applications.* Annals of the New York Academy of Sci., vol.80, pp. 140–163.

Dov M. Gabbay, C. J. Hogger, J. A. Robinson (eds.) (1993 ff.). *Handbook of Logic in Artificial Intelligence and Logic Programming.* Clarendon Press.

Gerhard Gentzen (1935). *Untersuchungen über das logische Schließen.* Mathematische Zeitschrift **39**, pp. 176-210, 405–431.

Gerhard Gentzen (1938). *Die gegenwärtige Lage in der mathematischen Grundlagenforschung – Neue Fassung des Widerspruchsfreiheitsbeweises für die reine Zahlentheorie.* Forschungen zur Logik und zur Grundlegung der exakten Wissenschaften, Folge 4, Leipzig.

Alfons Geser (1995). *A Principle of Non-Wellfounded Induction.* In: Tiziana Margaria (ed.). Kolloquium Programmiersprachen und Grundlagen der Programmierung, MIP–9519, pp. 117-124, Univ. Passau.

Martin Giese (1998). *Integriertes automatisches und interaktives Beweisen: die Kalkülebene.* Master's thesis, Univ. Karlsruhe. `http://i11www\netdotira\netdotuka\netdotde\netslash\~{}giese\netslashda\psgz (May 09, 2000).`

Martin Giese, Wolfgang Ahrendt (1999). *Hilbert's ε-Terms in Automated Theorem Proving.* 8th TABLEAUX 1999, LNAI 1617, pp. 171–185, Springer.

Kurt Gödel (1986 ff.). *Collected Works.* Solomon Feferman (ed.), Oxford Univ. Press.

Bernhard Gramlich, Wolfgang Lindner (1991). *A Guide to* UNICOM*, an Inductive Theorem Prover Based on Rewriting and Completion Techniques.* SEKI-Report SR–91–17 (SFB), FB Informatik, Univ. Kaiserslautern. `http://agent\netdotinformatik\netdotuni-kl\netdotde\netslashseki\netslash1991\netslashLindner\netdotSR-91-17\psgz (May 09, 2000).`

Reiner Hähnle, Peter H. Schmitt (1994). *The Liberalized δ-Rule in Free Variable Semantic Tableaux.* J. Automated Reasoning **13**, pp. 211–221, Kluwer (Springer).

David Hilbert, Paul Bernays (1968/70). *Grundlagen der Mathematik.* 2nd ed., Springer.

Paul Howard, Jean E. Rubin (1998). *Consequences of the Axiom of Choice.* Math. Surv. and Monographs, Vol. 58, American Math. Society.

Bart Jacobs, Jan Rutten (1997). *A Tutorial on (Co)Algebras and Co(Induction).* Bull. EATCS **62**, pp. 222–259. `http://www\netdotcs\netdotkun\netdotnl\netslash\~{}bart\netslashPAPERS\netslashJR\netdotps\netdotZ (Oct. 15, 1999).`

Stig Kanger (1963). *A Simplified Proof Method for Elementary Logic.* In: Siekmann & Wrightson (1983), Vol. 1, pp. 364-371.

Deepak Kapur, Hantao Zhang (1989). *An Overview of Rewrite Rule Laboratory (*RRL*).* 3rd RTA 1989, LNCS 355, pp. 559–563, Springer.

Manfred Kerber (1998). *Proof Planning: A Practical Approach to Mechanized Reasoning in Mathemantics.* In: Bibel & Schmitt (1998), Vol. 3, pp. 77–95.

Jan Willem Klop (1992). *Term Rewriting Systems.* In: S. Abramsky, Dov M. Gabbay, T. S. E. Maibaum (eds.). *Handbook of Logic in Computer Science, Vol. 2*, Clarendon Press.

Donald E. Knuth (1997 f.). *The Art of Computer Programming.* 3rd ed., Addison-Wesley.

Michaël Kohlhase (1995). *Higher-Order Tableaux.* 4th TABLEAUX 1995, LNAI 918, pp. 294–309, Springer. Revised version is: Kohlhase (1998).

Michaël Kohlhase (1998). *Higher-Order Automated Theorem Proving.* In: Bibel & Schmitt (1998), Vol. 1, pp. 431–462.

Emmanuël Kounalis, Michaël Rusinowitch (1990). *Mechanizing Inductive Reasoning.* 8th AAAI 1990, pp. 240–245, MIT Press.

Georg Kreisel (1965). *Mathematical Logic.* In: T. L. Saaty (ed.). Lectures on Modern Mathematics, Vol. III, pp. 95–195, John Wiley & Sons, New York.

Christoph Kreitz, Brigitte Pientka (2000). *Matrix-based Inductive Theorem Proving.* 9th TABLEAUX 2000, LNAI 1847, pp. 294–308, Springer.

Ulrich Kühler (2000). *A Tactic-Based Inductive Theorem Prover for Data Types with Partial Operations.* Ph.D. thesis, Infix, Akademische Verlagsgesellschaft Aka GmbH, Sankt Augustin, Berlin.

Ulrich Kühler, Claus-Peter Wirth (1996). *Conditional Equational Specifications of Data Types with Partial Operations for Inductive Theorem Proving.* SEKI-Report SR–96–11, FB Informatik, Univ. Kaiserslautern. Short version in: 8th RTA 1997, LNCS 1232, pp. 38–52, Springer. `http://\wwwcppubladdressrta97\netslashwelcome\netdothtml (May 09, 2000).`

Al(bert) C. Leisenring (1969). *Mathematical Logic and Hilbert's ε-Symbol.* Gordon and Breach, New York.

Vladimir A. Lifschitz (1971). *Specialization of the Form of Deduction in the Predicate Calculus with Equality and Function Symbols.* In: Orevkov (1971), pp. 1-24.

Vladimir A. Lifschitz (1989). *What Is the Inverse Method?.* J. Automated Reasoning **5**, pp. 1–23, Kluwer (Springer).

Sergey Yu. Maslov (1971). *The Inverse Method for Establishing Deducibility for Logic Calculi.* In: Orevkov (1971), pp. 25-96.

Dale A. Miller (1992). *Unification under a Mixed Prefix.* J. Symbolic Computation **14**, pp. 321–358, Academic Press (Elsevier).

Andreas Nonnengart (1996). *Strong Skolemization.* MPI–I–96–2–010, Max Planck Inst. für Informatik, Saarbrücken.

Vladimir P. Orevkov (1971). *The Calculi of Symbolic Logic.I.* American Mathematical Society, Providence, Rhode Island.

Peter Padawitz (1996). *Inductive Theorem Proving for Design Specifications.* J. Symbolic Computation **21**, pp. 41–99, Academic Press (Elsevier).

Peter Padawitz (1998). EXPANDER. *A System for Testing and Verifying Functional Logic Programs.* `http://\wwwpadawitzpubl{ExpaTex\psgz} (Sept. 14, 1999).`

Michaël S. Paterson, Mark N. Wegman (1978). *Linear Unification.* J. Computer and System Sci. **16**, pp. 158–167, Academic Press (Elsevier).

Padawitz (2000). *Swinging types = functions + relations + transition systems.* Theoretical Computer Sci. **243**, pp. 93–165, Elsevier.

Dag Prawitz (1960). *An Improved Proof Procedure.* In: Siekmann & Wrightson (1983), Vol. 1, pp. 159-199.

Dag Prawitz (1965). *Natural Deduction.* Almquist & Wiksells, Uppsala.

Martin Protzen (1994). *Lazy Generation of Induction Hypotheses.* 12th CADE 1994, LNAI 814, pp. 42–56, Springer. Long version in: Protzen (1995).

Martin Protzen (1995). *Lazy Generation of Induction Hypotheses and Patching Faulty Conjectures.* Ph.D. thesis, Infix, Akademische Verlagsgesellschaft Aka GmbH, Sankt Augustin, Berlin.

Horst Reichel (1987). *Initial Computability, Algebraic Specifications, and Partial Algebras.* Clarendon Press.

Grigore Roşu, Joseph Goguen (1998). *Hidden Congruent Deduction.* In: Caferra & Salzer (2000), pp. 251–266. `http://www-cse\netdotucsd\netdotedu\netslashusers\netslashgoguen\netslashps\netslashcong\netdotps\netdotgz (Oct. 20, 1999).`

Herman Rubin, Jean E. Rubin (1985). *Equivalents of the Axiom of Choice.* Elsevier.

Jörg Siekmann, Graham Wrightson (eds.) (1983). *Automation of Reasoning.* Springer.

Raymond M. Smullyan (1968). *First-Order Logic.* Springer.

Wayne Snyder, Jean Gallier (1989). *Higher-Order Unification Revisited: Complete Sets of Transformations.* J. Symbolic Computation **8**, pp. 101–140, Academic Press (Elsevier).

Christoph Walther (1992). *Computing Induction Axioms.* 3rd LPAR 1992, LNAI 624, pp. 381–392, Springer.

Christoph Walther (1994). *Mathematical Induction.* In: Gabbay &al. (1993 ff.), Vol. 2, pp. 127-228.

Claus-Peter Wirth (1997). *Positive/Negative-Conditional Equations: A Constructor-Based Framework for Specification and Inductive Theorem Proving.* Ph.D. thesis, Verlag Dr. Kovač, Hamburg.

Claus-Peter Wirth (1998). *Full First-Order Sequent and Tableau Calculi With Preservation of Solutions and the Liberalized δ-Rule but Without Skolemization.* Report 698/1998, FB Informatik, Univ. Dortmund. Short version in: Gernot Salzer, Ricardo Caferra (eds.). Proc. 2nd Int. Workshop on First-Order Theorem Proving (FTP'98), pp. 244–255, Tech. Univ. Vienna, 1998. Short version also in: Caferra & Salzer (2000), pp. 283–298. `http://\wwwcppubladdressftp98\netslashwelcome\netdothtml` (May 09, 2000).

Claus-Peter Wirth (1999). *Full First-Order Free Variable Sequents and Tableaux in Implicit Induction.* 8th TABLEAUX 1999, LNAI 1617, pp. 293–307, Springer. `http://\wwwcppubladdresstab99\netslashwelcome\netdothtml` (May 09, 2000).

Claus-Peter Wirth, Klaus Becker (1995). *Abstract Notions and Inference Systems for Proofs by Mathematical Induction.* 4th CTRS 1994, LNCS 968, pp. 353–373, Springer. `http://\wwwcppubladdressctrs94\netslashwelcome\netdothtml` (May 09, 2000).

Claus-Peter Wirth, Bernhard Gramlich (1994a). *A Constructor-Based Approach for Positive/Negative-Conditional Equational Specifications.* J. Symbolic Computation **17**, pp. 51–90, Academic Press (Elsevier). `http://\wwwcppubladdressjsc94\netslashwelcome\netdothtml` (May 09, 2000).

Claus-Peter Wirth, Bernhard Gramlich (1994b). *On Notions of Inductive Validity for First-Order Equational Clauses.* 12th CADE 1994, LNAI 814, pp. 162–176, Springer. `http://\wwwcppubladdresscade94\netslashwelcome\netdothtml` (May 09, 2000).

Ludwig Wittgenstein (1922). *Tractatus Logico-Philosophicus.* C. K. Ogden (ed.), Routledge and Kegan Paul, London. `http://www\netdotkfs\netdotorg\netslash\~{}jonathan\netslashwitt\netslashtlph\netdothtml` (Feb. 09, 2000).

Ludwig Wittgenstein (1939). *Lectures on the Foundations of Mathematics, Cambridge.* Cora Diamond (ed.), from the notes of R. G. Bosanquet, Norman Malcolm, Rush Rhees, and Yorick Smythies, Univ. of Chicago Press, 1989.