R^n- and G^n-Logics

Claus Hintermeier
Hélene Kirchner
Peter D. Mosses

BRICS Report Series RS-96-51

ISSN 0909-0878 December 1996

R^n- and G^n-Logics

Claus Hintermeier, Hélène Kirchner[1] and Peter D. Mosses[2]

[1] CRIN-CNRS & INRIA-Lorraine,
BP 239,
F-54506 Vandœuvre-lès-Nancy Cedex, France
email: hkirchne@loria.fr
[2] BRICS, University of Aarhus,
Ny Munkegade, bldg. 540
DK-8000 Aarhus C, Danmark
email: pdmosses@brics.dk

Abstract. This paper proposes a simple, set-theoretic framework providing expressive typing, higher-order functions and initial models at the same time. Building upon Russell's ramified theory of types, we develop the theory of R^n-logics, which are axiomatisable by an order-sorted equational Horn logic with a membership predicate, and of G^n-logics, that provide in addition partial functions. The latter are therefore more adapted to the use in the program specification domain, while sharing interesting properties, like existence of an initial model, with R^n-logics. Operational semantics of R^n-/G^n-logics presentations is obtained through order-sorted conditional rewriting.

1 Motivations

The general goal of this work is to give a simple, set-theoretic framework providing expressive typing, higher-order functions and initial models at the same time. The decision to use set-theoretic interpretations is taken mainly because of their simplicity, and their intuitive appeal for formal software specification. Higher-order functions and highly expressive types including polymorphism, dependent and higher-order types are frequently used concepts that we want to handle in a uniform way. We also want to provide a concise and sufficiently simple deduction system, easily implemented by rewriting.

Algebraic specification techniques model types as sets and subtypes as subsets, called sorts and subsorts, respectively. However, in conventional algebraic frameworks, sort expressions are generally restricted to constants, functions are first-order and sort assertions are static and unconditional. From the algebraic approach, we want to keep the initial semantics which provides a unique model up to isomorphism for classes of models, and rewrite techniques for operational semantics.

Defining function graphs as sets of argument/value pairs for each function is a classical set-theoretic technique to give semantics to functions. This is the case in Russell's ramified theory of types from which we started our work. However, self-applicable functions, as in the untyped λ-calculus, are not possible in that theory; our approach will include references to sets (i.e. names) in order to cope with this problem.

2 Introduction to R^n-Logics and G^n-logics

An R^n-logic is an equational Horn logic with membership predicate $\in$. The parameter n, which is a natural number ≥ 0, gives a bound on the nesting depth of the sets used in interpretations. Analogous to Whitehead and Russell [37], we assign orders $i \in [0..n]$ to variables and terms, so that a term of order 0 is interpreted as an individual value and a term of order 1 or greater as a set. Moreover formulas are restricted to stratified ones, i.e. $t \in t'$ is a valid formula only if t is of one order lower than t' and $t = t'$ is an admissible equation only if all its instances are order-preserving, i.e. left and right hand side are of the same order. This prevents Russell's paradox. Furthermore, the syntax of terms does not include the empty set as a predefined constant, in order to avoid negation in the considered Horn clause fragment. Instead we have the restriction that all sets represented by terms are non-empty.

The difference from Russell's ramified theory of types [37] is the consideration of non-term-generated models. Our choice of models avoids Gödel's second theorem which proves the incompleteness of deduction systems like the one in Principia Mathematica [37]. We get a complete deduction system by using non-standard axioms of choice and extensionality. This goes along with an extension of the signature by choice functions, which are deterministic in our framework. The essential use of choices here is the possibility to express that there may be other objects than those represented by terms in a particular model. Hence, given a set of individuals, we define sets together with choices as follows: a choice of order 0 is a term representing a (possibly non-standard) individual. A set of order 1 is a set of choices of order 0 and individuals. A choice of order $k \in [1..n-1]$ is a term representing a set of order k. A set of order $k \in [2..n]$ is a set of choices and sets of order $k-1$.

Therefore, the underlying idea for the sort structure is to start with Russell's ramified theory of types up to order n, which is basically many-sorted. Assume $\{s_0, \ldots, s_n\}$ is the set of sorts. Then s_0 is the sort of individuals represented by terms and for $i \in [1..n]$, s_i is the sort of terms representing sets of sets of $\ldots$ (i times) of individuals. Therefore, sets in s_i are called sets of *order* i. Now, we add supersorts $s'_0, \ldots, s'_n$ for $s_0 \ldots, s_n$, respectively, such that s'_0 is the sort of all individuals not necessarily represented by a term, s'_i, $i \in [1..n]$, is the sort of all sets of sets of $\ldots$ (i times) of individuals, also not necessarily represented by a term. This allows us for example to reason about real numbers as individuals although it is impossible to represent all of them as terms. Choice functions are thus defined from s_i to s'_{i-1}, $i \in [1..n]$. Let us call this intermediate theory *simple R^n-logic.* The interpretation of the sort structure for R^n-logic is illustrated in Figure 1.

Simple G^n-logics are defined analogously, except that functions are not necessarily total. When f is declared as a function e.g. from individuals to individuals

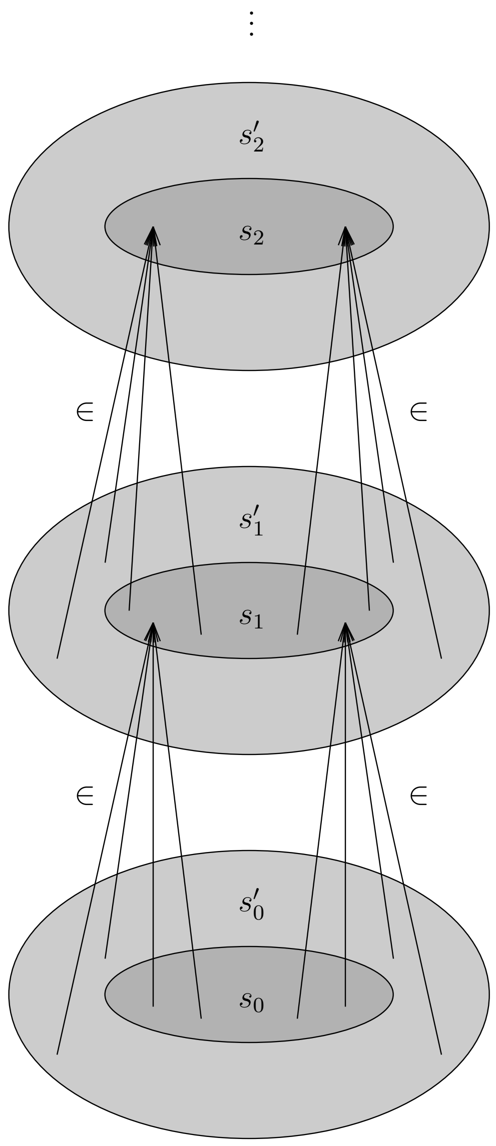

Fig. 1. Interpretation of sorts in R^n-logic

in G^n-logics, this does not imply that f is completely defined over all individuals; however, if $f(t)$ is defined for some individual term t, then $f(t)$ has to be an individual. G^n-logics seem to be more natural for program specifications than R^n-logics. However, they are also a bit more complex since we have to handle definedness of a term t using an additional predicate $Ex\ t$.

The difference between simple and full R^n-/G^n-logics is that the latter include *references to sets*, which are individuals. As the last step of the construction, we add sorts for references to objects (i.e. to individuals, sets of individuals, etc.) represented by terms in one of the sorts $s_0, \ldots, s_n$. The intuition for references is that they are names for the objects they refer to. References in our framework are mainly useful for the construction of function graphs: it is possible to define a graph of a higher-order function as a set of individual pairs of references to arguments and results.

The basic result in this paper is the existence of a sound and complete deduction system for full R^n-/G^n-logics. This is not in contradiction to the general incompleteness of higher-order predicate logics, since we use a particular non-standard model notion, similar to Henkin models [9]. Furthermore, this category of models together with standard order-sorted homomorphisms contains an initial object for each presentation, since we use a Horn clause fragment and all operations are deterministic.

3 Illustration of R^n-and G^n-logics

Let us give some examples of specifications in R^n- and G^n-logics, before proceeding to the formal definitions. To start with, the example of polymorphic ordered lists illustrates the expressiveness of R^n- and G^n-logics, as it involves types depending on functions.

Recall that s_0, s_1 and s_2 are the sorts of individuals, sets of individuals, and sets of such sets, respectively. Let the signature contain the operators *lists* (ordered lists), *nil*, *cons* (ordered list constructors), *insert* (element insertion), *orders* (orders over elements) and *pair* with the following declarations of domains and co-domains:

$$\begin{array}{llll} \mathit{lists} & : s_1, s_1 \to s_1 & \mathit{nil} & : s_1 \to s_0 \\ \mathit{cons} & : s_0, s_0 \to s_0 & \mathit{orders} & : s_1 \to s_2 \\ \mathit{insert} & : s_0, s_0 \to s_0 & \mathit{pair} & : s_0, s_0 \to s_0. \end{array}$$

Let additionally x, y, l be variables of sort s_0, and i, o be variables of sort s_1. Let ϕ be the conjunction $o \in \mathit{orders}(i) \wedge x \in i \wedge y \in i \wedge l \in \mathit{lists}(i, o)$ and ϕ' be $\phi \wedge \mathit{cons}(y, l) \in \mathit{lists}(i, o)$. The axioms are:

$$\begin{array}{r} o \in \mathit{orders}(i) \Rightarrow \mathit{nil}(o) \in \mathit{lists}(i, o) \\ o \in \mathit{orders}(i) \wedge x \in i \Rightarrow \mathit{cons}(x, \mathit{nil}(o)) \in \mathit{lists}(i, o) \\ \mathit{pair}(x, y) \in o \wedge \phi' \Rightarrow \mathit{cons}(x, \mathit{cons}(y, l)) \in \mathit{lists}(i, o) \\ \phi \Rightarrow \mathit{insert}(x, l) \in \mathit{lists}(i, o) \end{array}$$

$$pair(x, y) \in o \wedge \phi' \Rightarrow insert(x, cons(y, l)) = cons(x, cons(y, l))$$
$$pair(y, x) \in o \wedge \phi \Rightarrow insert(x, cons(y, l)) = cons(y, insert(x, l)).$$

The reader should not be alarmed by the amount of detail in the above axioms: here, for simplicity, we are using the bare R^n-logic specifications, without introducing any of the syntactic sugar that would be needed for large-scale use in practical applications.

Some simple consequences of the above specification in R^n-logic, taking the set of natural numbers N for i and assuming $leq = \{pair(m, n) \mid m \leq n\} \in orders(N)$, include:

$$cons(1, cons(3, nil(leq))) \in lists(N, leq)$$
$$insert(2, cons(1, cons(3, nil(leq)))) = cons(1, cons(2, cons(3, nil(leq))))$$

whereas $cons(3, cons(1, nil(leq))) \in lists(N, leq)$ is *not* a consequence.

In R^n-logics functions are total, so all well-sorted terms are required to have values. One may however restrict ones attention to those terms whose values belong to some particular sets. For example, above we may be interested only in those terms whose values belong to $lists(N, leq)$; we may regard $cons(3, cons(1, nil(leq)))$ as an error term.

In G^n-logics, however, functions may be partial, and the values of error terms may be left undefined. This provides a canonical way of distinguishing errors: we do not have to identify some particular sets of interest. Taking the specification above in G^n-logics, we get that $cons(3, cons(1, nil(leq)))$ is undefined (in the standard model of the specification, at least).

R^n-/G^n-logics offer the possibility to express and manipulate functions via their graphs. Suppose $f : s_i \to s_j$. Then we may specify the graph $g :\to s_1$ of f by:

$$pair(ref(x), ref(y)) \in g \iff f(x) = y$$

where $ref : s_k \to s'_0$ is the function that maps each value in s_k to the corresponding reference, and $pair : s'_0, s'_0 \to s_0$.

Our logics also allow us to specify self-applicable functions with set theoretic semantics. Assume we want to define domain restrictions $restrict(f, s)$ for functions, often written $f|_s$. Now, our type system does not prevent us writing $restrict(restrict, s)$, where s is a set of function graphs defined in the same logic, since the graph of restrict may be defined as a simple set of individuals as shown above.

Let us conclude these illustrations with another familiar example: *maplist*, an operator that takes an operator and a list as arguments, and applies the operator to each item in the list, making a list of the results. Here, having already specified ordered lists, we consider only the case where the applied operator belongs to *monotones*, the set of monotone increasing functions on the ordered items, so that the resulting list is also ordered. The signature for ordered lists is extended as follows:

$$maplist : s_1, s_0 \to s_0 \qquad monotones : s_1, s_1 \to s_2.$$

We use the same variables as in the specification of ordered lists above, together with m of sort s_1. The axioms are:

$$\begin{aligned}
&o \in \mathit{orders}(i) \wedge m \in \mathit{monotones}(i,o) \Rightarrow \mathit{maplist}(m, \mathit{nil}(o)) = \mathit{nil}(o)\\
&o \in \mathit{orders}(i) \wedge m \in \mathit{monotones}(i,o) \wedge x \in i \wedge l \in \mathit{lists}(i,o) \wedge\\
&\mathit{cons}(x,l) \in \mathit{lists}(i,o) \wedge \mathit{pair}(x,y) \in m \Rightarrow\\
&\qquad\qquad \mathit{maplist}(m, \mathit{cons}(x,l)) = \mathit{cons}(y, \mathit{maplist}(m,l))
\end{aligned}$$

Notice that the domain of m and the set of list items i have to be the same. When i is properly included in the domain of m, we may either increase i to match (thereby increasing the set of lists) or make use of $\mathit{maplist}(\mathit{restrict}(m,i),l)$.

4 R^n-Logics

Presentations in simple and full R^n-logics have a signature and Horn clause axioms, including conditional membership formulas and conditional equalities, which allow to express, among other things, polymorphic and dependent types. Let us give their formal definition, using a fragment of conventional first-order order-sorted logic [30]:

Definition 1. An *R^n-signature Σ* is an order-sorted signature $(\mathbf{S}, \leq_{\mathbf{S}}, \mathbf{F}, \mathbf{R})$, such that:

– $\mathbf{S}$ is a non-empty set of sorts, $\mathbf{S} = \{s_0, \ldots, s_n\} \cup \{s_{0r}, \ldots, s_{nr}\} \cup \{s'_0, \ldots, s'_n\}$,
– $\leq_{\mathbf{S}}$ is an ordering relation on $\mathbf{S}$ defined by: for all $i \in [0..n]$, $s_i \leq_{\mathbf{S}} s'_i$ and $s_{ir} \leq_{\mathbf{S}} s'_0$,
– $\mathbf{F}$ is a set of function symbols. Any f in $\mathbf{F}$ with an arity k has a set of ranks $f \ : \ s_1, \ldots, s_k \to s$ with $s_1, \ldots, s_k, s$ in $\mathbf{S}$. If $n > 0$, $\mathbf{F}$ contains the following functions:
  • *choose* with ranks $\{(\mathit{choose} \ : \ s_i \to s'_{i-1}) \mid i \in [1..n]\}$,
  • *ref* with ranks $\{(\mathit{ref} \ : \ s_i \to s_{ir}) \mid i \in [0..n]\}$,
  • *deref* with ranks $\{(\mathit{deref} \ : \ s_{ir} \to s_i) \mid i \in [0..n]\}$,
  All other functions f have ranks of the form $(f \ : \ s''_1, \ldots, s''_q \to s'')$, where the sorts s'', s''_i belong to $\{s_0, \ldots, s_n, s_{0r}, \ldots, s_{nr}\}$ for all $i \in [1..q]$, so that each well-sorted term has a unique least sort (Σ is called *regular* in this case).
– $\mathbf{R}$ is a set of relation symbols. Any p in $\mathbf{R}$ with an arity k has a set of ranks $p \ : \ s_1, \ldots, s_k$ with $s_1, \ldots, s_k$ in $\mathbf{S}$. If $n > 0$, $\mathbf{R}$ contains the relation $\in$ with ranks $\{(\in \ : \ s'_{i-1} \ s_i) \mid i \in [1..n]\}$.

A Σ-term (or literal) is ground if it does not contain any variable.

An *R^n-presentation* $\mathbf{P}$ is a set of Σ-Horn clauses, written $G \Rightarrow L$, where G is the premiss (or the body) and L is the conclusion (or the head). A clause $G \Rightarrow$ with an empty conclusion is called a goal clause by analogy with logic programming. All variables of sorts $s'_0, \ldots, s'_n$ in $\mathbf{P}$ occur only on the left of a membership relation '$\in$', '*choose*' only appears as top operator of a left argument of '$\in$' and all equalities occurring in clauses of $\mathbf{P}$ are sort preserving, which means that, for each instance, the least sort of the left-hand side is the same as the least sort of the right-hand side.

The sort structure of R^n-signatures is illustrated in Figure 2.

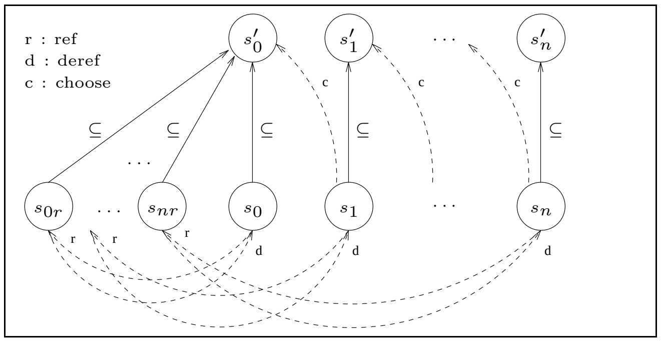

Fig. 2. The Sort Structure of R^n-signatures

First of all, remark the nature of the terms in the different sorts: the lower sorts s_m for m in $[0..n]$ should contain only terms without the symbol *choose*, the ones in s'_{m-1} those of s_m with one *choose* symbol on the top. In addition, s_{mr} contains only terms of s_m with *ref* additionally on its top, whenever $m \in [0..n]$. Regularity of the signature implies that for all $i, j \in [0..n]$, $i \neq j$, there is no ground term both of sort s_i and s_j.

The signature restrictions are relatively strong, because of the clear separation of sets of different order. In fact, the sort preservation of equalities results in static typing like in many-sorted logics. The choice functions are introduced in order to get a characteristic element for each set represented by a term. They also play a role in constructing the initial model.

Some difficulty arises with the treatment of these choice functions, which classically have a non-deterministic behaviour, but which we need to keep deterministic for technical reasons (Order-sorted equational Horn logic does not handle non-deterministic functions!). However, the definition of R^n-presentations does not allow us to define the result of a choice. Hence, the non-determinism has to move to the model level. Remark that these choice functions correspond with Hilbert's ϵ symbol [10, 17] and with the ϵ operator of HOL (cf. [8]).

References are particularly useful when we regard functions as sets, since their introduction allows for arbitrary arguments for functions in s'_0. From a set theoretic point of view, there may be objections to the use of references as elements different from sets. However, similar to individuals, we regard them as *a priori* given objects, just as terms are. The appropriate intuition is to think of references as purely syntactic, finite objects (just like terms), although the set they represent might be infinite.

R^n-models are a special case of first-order Σ-models defined for order-sorted equational Horn logic with non-overloaded semantics [34].

Definition 2. Let Σ be an R^n-signature, $\mathbf{X}$ a set of $\mathbf{S}$-sorted variables and $\mathbf{P}$ an R^n-presentation. An *R^n-model* $\mathbf{A}$ of $\mathbf{P}$ is a Σ-model of $\mathbf{P}$, where the carrier $C^{\mathbf{A}}$ contains a non-empty set of individuals ${s'_0}^{\mathbf{A}}$ in the sense of [37], such that:

- s_0 is interpreted as the set of individuals represented by a term,
- for $i \in [1..n]$, $s_i^{\mathbf{A}}$ contains only non-empty sets of order i represented by a term,
- for $i \in [0..n]$, $s_{ir}^{\mathbf{A}}$ is a set of symbolic, finite names contained in ${s'_0}^{\mathbf{A}}$,
- for $i \in [1..n]$, ${s'_i}^{\mathbf{A}}$ is the set of all sets of order i, plus all choices of order $i+1$ whenever $i < n$,
- $\mathit{choose}^{\mathbf{A}}$ gives a unique element from any set in $C^{\mathbf{A}}$ given as argument (choice function),
- *ref* is a bijective function over $s_i^{\mathbf{A}}$, $i \in [0..n]$, taking a set of order i and returning its unique name in $s_{ir}^{\mathbf{A}}$,
- *deref* is its inverse function on $s_{ir}^{\mathbf{A}}$, and
- $\in$ is interpreted as the membership relation on all sets in $C^{\mathbf{A}}$.

The notation $(\Sigma, \mathbf{P}, \mathbf{X}) \models_{R^n} \phi$ means that the formula ϕ is true in all R^n-models of $\mathbf{P}$.

Figure 3 now shows the set of deduction rules necessary to perform deduction in R^n-logics. Here, u^m, t^m stand for terms of sorts s_m, x^k, x^m, x^{kr} for variables of sorts s_k, s_m and s_{kr}, respectively. It is not difficult to prove their soundness:

Theorem 3. *Let* $\mathbf{P}$ *be an* R^n*-presentation. Assume that there exists at least one ground term of sort* s_m *for each* $m \in [0..n]$*. The deduction rules of Figure 3 are sound with respect to deduction in* R^n*-models.*

Figure 3 actually shows deduction rule schemes, which should be understood for all $m \in [1..n]$ and $k \in [0..n]$. Remark that **Choice** is the axiom of choice and **Ext** is a non-standard version of the axiom of extensionality, which can also be given as hereditary Harrop formula, although going out of the syntax of R^n-logics:

$$\textbf{ClassicExt}\ (\forall {x'}^{m-1}, {x'}^{m-1} \in y^m \Rightarrow {x'}^{m-1} \in z^m) \wedge \\ (\forall {x'}^{m-1}, {x'}^{m-1} \in z^m \Rightarrow {x'}^{m-1} \in y^m) \Rightarrow (y^m = z^m)$$

The intuition behind **Ext** is the freeness of *choose*, the choice function, which we keep without equational axioms by requiring that it does not occur in the conclusion of a clause in an R^n-presentation. Remark that **Ext** cannot be formulated as a Horn clause of the following form:

$$\textbf{HornExt}\quad \mathit{choose}(y^m) \in z^m \wedge \mathit{choose}(z^m) \in y^m \Rightarrow y^m = z^m.$$

This is simply not a valid clause, i.e. satisfied by all R^n-logic models, since it does not hold for any fixed interpretation of *choose*, as the following example shows:

1. R^n-Deduction Rules from Order Sorted Equational Horn Logics:

Reflexivity $\dfrac{}{x = x}$ if $x \in \mathbf{X}$

Axioms $\dfrac{}{G \Rightarrow L}$ if $G \Rightarrow L \in \mathbf{P}$

Substitutivity $\dfrac{G \Rightarrow L}{\sigma(G) \Rightarrow \sigma(L)}$ if $\sigma \in \mathsf{Subst}_{\Sigma}(\mathbf{X})$

Cut $\dfrac{G \wedge L' \Rightarrow L \quad G' \Rightarrow L'}{G \wedge G' \Rightarrow L}$

Paramodulation $\dfrac{G \Rightarrow L[s] \quad G' \Rightarrow (s = t)}{G \wedge G' \Rightarrow L[t]}$

2. Specific Deduction Rules of R^n-logics (for $k \in [0..n]$ and $m \in [1..n]$):

Ref $\dfrac{}{\mathit{ref}(\mathit{deref}(x^{kr})) = x^{kr}}$

Deref $\dfrac{}{\mathit{deref}(\mathit{ref}(x^{k})) = x^{k}}$

Choice $\dfrac{}{\mathit{choose}(x^{m}) \in x^{m}}$

Ext $\dfrac{\mathit{choose}(u^{m}) \in t^{m} \quad \mathit{choose}(t^{m}) \in u^{m}}{u^{m} = t^{m}}$

Fig. 3. R^n-deduction rules

Example 1. Let $n = 1$ and $\mathbf{P} = \{a \in A,\ b \in B,\ a \in C,\ b \in C\}$. Now, let $\mathbf{A}$ be the model interpreting A as $\{a\}$, B as $\{b\}$ and C as $\{a, b\}$. $\mathbf{A}$ trivially satisfies **ClassicExt**, since the premiss never gets true. However $\mathit{choose}^{\mathbf{A}}(C^{\mathbf{A}})$ has to coincide with either $\mathit{choose}^{\mathbf{A}}(A^{\mathbf{A}})$ or $\mathit{choose}^{\mathbf{A}}(B^{\mathbf{A}})$. But neither $A^{\mathbf{A}} = C^{\mathbf{A}}$ nor $B^{\mathbf{A}} = C^{\mathbf{A}}$. So **HornExt** is not valid in this R^n-model of $\mathbf{P}$. But, since *choose* is free, we may quantify over all possible interpretations, and therefore **Ext** is sound for deduction.

The fact that choice functions allow the formulation of non-standard extensionality as a deduction rule over Horn clauses without inductive conditions, is the key for achieving completeness of deduction for R^n-logics using the simple first-order, order-sorted, equational Horn clause calculus given in Figure 3.

The notation $(\Sigma, \mathbf{P}, \mathbf{X}) \vdash_{\mathcal{RNL}} \phi$ means that the formula ϕ is deducible from $\mathbf{P}$ using the R^n-deduction rules.

Theorem 4. *Let* $\mathbf{P}$ *be a* R^n*-presentation. Assume that there exists at least one ground term of sort* s_m *for each* $m \in [0..n]$*. The deduction rules of Figure 3 are complete: for any* $(\Sigma, \mathbf{X})$*-atom* L*, if* $(\Sigma, \mathbf{P}, \mathbf{X}) \models_{R^n} L$*, then* $(\Sigma, \mathbf{P}, \mathbf{X}) \vdash_{\mathcal{RNL}} L$*.*

The construction of the initial model $\mathbf{I}_R$ can be done inductively: for ground terms in s_0, i.e. representing individuals, and for non-standard ground terms in s'_i, $i \in [1..n]$, it is just the usual quotient construction. For terms t in s_i, $i \in [1..n]$, it is the set of all terms $u^{\mathbf{I}_R}$ such that $(\Sigma, \mathbf{P}, \mathbf{X}) \vdash_{\mathcal{RNL}} u \in t$. Using this construction, we get:

Theorem 5. *Let* $\mathbf{P}$ *be a* R^n*-presentation. Assume that there exists at least one ground term of sort* s_m *for each* $m \in [0..n]$*. Then* $\mathbf{I}_R$ *is an initial object in the category of* R^n*-models of* $\mathbf{P}$*.*

The proof relies on three lemmas stating the following facts:

- Equational replacement using sort-preserving equalities preserves the sort of the terms, i.e. their order.
- Deductive term models (as described above in the construction of the initial model) interpret "=" as the equality relation and "$\in$" as the membership relation.
- All terms occuring in a deductive term model are well-sorted.

Full details of the proofs can be found in [11].

5 G^n-Logics

In this section, we go over from R^n-logics to G^n-logics. The main difference between the two is that functions in G^n-logics are partial over the universe. This leads to complications when we define the corresponding models, since terms, that have to be interpreted in order-sorted equational Horn clause logic, might not necessarily be interpreted in G^n-logics.

G^n-logics were inspired by G-algebras [20], but can also be seen as closely related to Scott's logic of partial equality [33, 6] and to partial models in [32]. In what follows strictness is reflected by well-definedness axioms and leads to a unique notion of equality. G^n-signatures are just extensions of R^n-signatures.

Definition 6. A G^n*-signature* is an R^n-signature $\Sigma = (\mathbf{S}, \leq_{\mathbf{S}}, \mathbf{F}, \mathbf{R})$, such that $\mathbf{R}$ contains additionally the relation Ex with ranks $\{(Ex : s'_i) \mid i \in [0..n]\}$.
A G^n*-presentation* is an R^n-presentation, such that Σ is a G^n-signature.

The difference between R^n- and G^n-models relies on partiality of functions and on the existential predicate. Let us precisely state the additional requirements for G^n-models: in a G^n*-interpretation* $\mathbf{A}$,

- $\forall f \in \mathbf{F}$, if $(f : s_1, \ldots, s_q \to s)$, then $f^{\mathbf{A}}$ is a partial function from the Cartesian product $s_1^{\mathbf{A}} \times \ldots \times s_q^{\mathbf{A}}$ into $s^{\mathbf{A}}$.
- Moreover Ex is interpreted as the membership relation in $C^{\mathbf{A}}$.

The difference between R^n- and G^n-interpretations seems small at first glance, but is more fundamental. There is no more totality obligation for function symbols. However, a function f with rank $(f : s_1, \ldots, s_q \rightarrow s)$ must have $s^{\mathbf{A}}$ as codomain whenever it is defined for an element in the Cartesian product $s_1^{\mathbf{A}} \times \ldots \times s_q^{\mathbf{A}}$. So G^n-formulas have to include declarations for the domain of functions. The satisfaction relation in all G^n-interpretations is now denoted by $\models_{G^n}$.

The less restrictive definition of term interpretations prevents us to use order-sorted equational Horn clause deduction directly. In particular, **Reflexivity** is no more sound. Instead, we have to ask for well-definedness of the interpretation of a term before deducing reflexivity. Figure 4 shows the set of deduction rules for G^n-logics incorporating well-definedness formulas and therefore suitable for partial function handling.

1. G^n-Deduction Rules from Order Sorted Equational Horn Logics with Partial Functions:

   PartialReflex $\quad \dfrac{Ex\ t}{t = t}$

   Axioms $\quad \dfrac{}{G \Rightarrow L}$ if $G \Rightarrow L \in \mathbf{P}$

   SubstConform $\quad \dfrac{G \Rightarrow L}{\sigma(G) \Rightarrow \sigma(L)}$ if $\sigma \in \mathbf{P}\text{-}\mathtt{Subst}_{\Sigma}(\mathbf{X})$

   Cut $\quad \dfrac{G \wedge L' \Rightarrow L \quad G' \Rightarrow L'}{G \wedge G' \Rightarrow L}$

   Paramodulation $\quad \dfrac{G \Rightarrow L[s] \quad G' \Rightarrow (s = t)}{G \wedge G' \Rightarrow L[t]}$

   WellDef $\quad \dfrac{\Phi[t]}{Ex\ t}$ if $\Phi[t]$ *is a* $(\Sigma, \mathbf{X})$-atom containing t

2. Specific Deduction Rules of G^n-logics (for $k \in [0..n]$ and $m \in [1..n]$):

   Ref $\quad \dfrac{}{ref(deref(x^{kr})) = x^{kr}}$

   Deref $\quad \dfrac{}{deref(ref(x^{k})) = x^{k}}$

   Choice $\quad \dfrac{}{choose(x^{m}) \in x^{m}}$

   Ext $\quad \dfrac{choose(u^{m}) \in t^{m} \quad choose(t^{m}) \in u^{m}}{u^{m} = t^{m}}$

Fig. 4. G^n-deduction rules

As a consequence, the deduction system for G^n-logics is not a conservative extension of the R^n-logics system, since the introduction of partial functions makes it necessary to use a restricted form of reflexivity and substitution. A Σ-substitution is called **P**-conform, when $Ex\ \sigma(x)$ can be proved for all x in the domain of σ. $\mathbf{P}\text{-}\mathsf{Subst}_{\Sigma}(\mathbf{X})$ stands for the set of **P**-conform Σ-substitutions whose domain is a subset of **X**. Additionally, we need to add a well-definedness rule **WellDef**.

As in the case of R^n-logics, soundness and completeness of the calculus with respect to the corresponding class of models can be stated. The notation $(\Sigma, \mathbf{P}, \mathbf{X}) \vdash_{\mathcal{GNL}} \phi$ means that the formula ϕ is deducible from **P** using the G^n-deduction rules.

Theorem 7. (soundness, completeness of deduction, initial model)
Let **P** *be a* G^n*-logic presentation. Assume that there exists at least one ground term of sort* s_m *for each* $m \in [0..n]$*.*
The deduction rules of Figure 4 are sound w.r.t. deduction in G^n*-models.*
The deduction rules of Figure 4 are complete:
for any $(\Sigma, \mathbf{X})$*-atom* L*, if* $(\Sigma, \mathbf{P}, \mathbf{X}) \models_{G^n} L$*, then* $(\Sigma, \mathbf{P}, \mathbf{X}) \vdash_{\mathcal{GNL}} L$*.*
Furthermore, there exists an initial object $\mathbf{I}_G$ *in the category of* G^n*-models of* **P***.*

The proofs of these results can be found in [11]. The main difficulty here comes from partiality of functions. In contrast to R^n-logics, we do not need to interpret all terms in the set of all Σ-ground terms, just those terms that have to denote an element in all models.

6 Operationalisation by Rewriting Techniques

The main application for R^n-and G^n-logics is the software specification and verification domain. The major idea is to use set theoretic semantics for both types and higher-order features. We have therefore investigated operationalisation aspects for R^n-and G^n-presentations through first-order typed conditional term rewriting systems [11]. These techniques may be used for the design of a programming language in the style of OBJ-3, but using dynamic types and sort constraints with clear set theoretic semantics.

In order to transform a R^n-or G^n-presentation into a conditional term rewriting system, we can adapt a saturation procedure on equational Horn clauses, such as the ones described for instance in [2, 27]. The three main inference rules are superposition into conclusion, superposition into premisses and equality resolution. Application of these rules requires the existence of a well-founded reduction ordering [4] on terms and literals. Superposition is performed by unifying the maximum term in an equational conclusion of a clause with a subterm in another clause, then performing a paramodulation step on the instantiated clauses. In this process, relation symbols are considered as boolean functions. Other inference rules, such as subsumption by another clause and elimination of tautologies, are also added to eliminate redundant clauses. An ordered strategy is used for reducing the search space by using only maximal terms and literals with respect to the given ordering. A saturation process is a sequence of presentations $(P_0, P_1, \ldots)$, also called a derivation, where P_i is deduced from P_{i-1} by application of one inference rule. This derivation must be fair in the intuitive sense that no clause is forgotten in the process of generating consequences. P_0 is consistent if and only if the empty clause does not belong to any P_i. Moreover if P_∞ is the set of persisting clauses in this fair derivation and does not contain the empty clause, then one can construct from P_∞ a conditional term rewriting system which is terminating and confluent in the initial model of P_0. This indeed provides a way to compute in a finite and unambiguous way the normal form of any expression in P_0. The complete description of the process and its proof can be found in [11].

In R^n-logics, the saturation procedure defined for instance in [29], can be reused after replacing the unsorted unification algorithm by an order-sorted one. This is possible thanks to the sort preservation of R^n-presentations. If saturation terminates, then the set of ground instances of rules decreasing with respect to a given ordering on terms [5], forms a terminating and confluent term rewriting system on ground terms.

Concerning G^n-logics, we have to change the inference rules for saturation a little bit due to the partiality of functions, which results in a partial form of reflexivity (cf. **PartialReflex**). We omit details here (the interested reader may refer to [11]). Let us instead illustrate the saturation technique with the following example :

Example 2. Assume we want to define stacks as set St over elements of sort E. Let the R^n-presentation $\mathbf{P}$ be defined as follows:

$$\begin{aligned}
&\epsilon \in E\\
&empty \in St\\
x \in E \wedge y \in St \Rightarrow\ &pop(push(x,y)) \in St\\
x \in E \wedge y \in St \Rightarrow\ &top(push(x,y)) \in E\\
x \in E \wedge y \in St \Rightarrow\ &push(x,y) \in St\\
x \in E \wedge y \in St \Rightarrow\ &pop(push(x,y)) = y\\
x \in E \wedge y \in St \Rightarrow\ &top(push(x,y)) = x.
\end{aligned}$$

Let $pop \succ top \succ push \succ empty \succ \epsilon \succ E \succ St$ be the precedence for a lexicographic path ordering (LPO) on terms [14]. The multiset expression giving the complexity of membership formulas with respect to this LPO is defined in the same way as for equalities in [26], i.e. by ignoring the relation symbol.

Using the inference rules from [26] with an order-sorted equational constraint solver, we can then eliminate the membership formulas for $pop(push(x,y))$ and $top(push(x,y))$ (i.e. the third and fourth clauses in $\mathbf{P}$ above) by simplification with the two last equalities. The result is a saturated presentation.

Refuting $(top(pop(push(x,y))) = top(y) \Rightarrow)$ then gives $(top(y) = top(y) \Rightarrow)$, which has the identity substitution as solution.

It should be more natural to consider $\mathbf{P}$ as a presentation in G^n-logic, since the operations *pop* and *top* are partial functions not defined on the empty stack. Then the goal $(top(pop(push(x,y))) = top(y) \Rightarrow)$ has an infinite set of solutions

$\{y \mapsto \mathit{push}(\epsilon, \ldots \mathit{push}(\epsilon, \mathit{empty}) \ldots)\}$, but no more the identity substitution, since *top* is undefined for *empty*.

In order to get a better efficiency for membership proofs, we extend Horn clauses furthermore by assertions, which are a kind of cache mechanism for atoms derivable from the premiss of a clause using the current presentation. This is illustrated in the following example :

Example 3. Let $\mathbf{F} = \{id, 0, Nat\}$ and $\mathbf{P}$ the following set of clauses :

$$\begin{array}{ll} & 0 \in Nat \\ x \in Nat & \Rightarrow id(x) = x \\ id(0) \in Nat \Rightarrow & \end{array}$$

We start saturation (using an LPO with precedence $id \succ 0 \succ Nat$) by deducing from the first clause an assertion for the last one. The result is :

$$id(0) \in Nat \Rightarrow \quad [\![0 \in Nat]\!]$$

Here, the part added to the clause between the brackets $[\![$ and $]\!]$ is the assertion. The remainder is a usual clause. For better readability, we omit ordering conditions and constraints in our argumentation. Now, we can superpose the second clause into the previous one to get :

$$0 \in Nat \Rightarrow \quad [\![0 \in Nat]\!]$$

Now the goal $0 \in Nat$ is satisfied since the corresponding atom is already present in the assertion. Hence, we can deduce without further superposition the empty clause, which proves the inconsistency of $\mathbf{P}$. The gain of efficiency appears in bigger examples when an assertion is used several times.

A saturation calculus with assertions in developed in [11] and gives a semantics to saturation in a fragment of G^1-logics in the style of [12, 13]. Saturation is performed for order-sorted presentations with polymorphic, dynamic types and partial functions, using so-called decorated terms, in which set terms are added locally to term nodes to store typing information [11]. The possibility to mix this style of dynamic typing with the more efficient static typing, like in [1], is to be investigated.

7 Discussion

In this section, we discuss more precisely the connections of R^n-and G^n-logics with set theory and algebraic specifications, but also mention relations with higher-order logic and functional programming.

Set Theory: R^n-logics stem from naive set theory and may also be seen as a fragment of Z [35]. Let us examine which part of set theory can be easily specified.

Horn clauses are built with two logical connectives, implication and conjunction, which correspond directly with inclusion and intersection. It is therefore not surprising that R^n-and G^n-logics allow specifying set inclusion $\subseteq$ and intersection $\cap$, assuming x^m, y^m to be variables of sort s_m, z'^{m-1} be a variable of sort s'_{m-1}, for $m \in [1..n]$:

$$\begin{array}{ll}
\mathit{choose}(x^m) \in y^m & \Rightarrow x^m \subseteq y^m \\
z'^{m-1} \in x^m \wedge x^m \subseteq y^m & \Rightarrow z'^{m-1} \in y^m \\
z'^{m-1} \in x^m \wedge z'^{m-1} \in y^m & \Rightarrow z'^{m-1} \in x^m \cap y^m \\
z'^{m-1} \in x^m \cap y^m & \Rightarrow z'^{m-1} \in x^m \\
z'^{m-1} \in x^m \cap y^m & \Rightarrow z'^{m-1} \in y^m.
\end{array}$$

Depending on the logic, we have different behaviours. In R^n-logics, all terms are defined and therefore all intersections of non-empty sets (represented by a ground term) denote a non-empty set, since they are represented by a ground term. In G^n-logics, terms t are non-empty if they are defined, i.e. if we can derive *Ex t*. Hence, intersections like $\mathit{int} \cap \mathit{list}$, where *int* is the set of integers and *list* is the set of lists, can be specified as empty if we avoid to define their existence and use initial semantics.

One might try to define singletons $\mathit{sgl}(x^m)$ by:

$$\begin{array}{ll}
z'^m = x^m & \Rightarrow z'^m \in \mathit{sgl}(x^m) \\
z'^m \in \mathit{sgl}(x^m) & \Rightarrow z'^m = x^m.
\end{array}$$

But this last definition is not admissible due to the requirement of sort preservation for the equalities in clauses. Union would be:

$$\begin{array}{ll}
z'^k \in x^m & \Rightarrow z'^k \in x^m \cup y^m \\
z'^k \in y^m & \Rightarrow z'^k \in x^m \cup y^m \\
z'^k \in x^m \cup y^m & \Rightarrow z'^k \in x^m \vee z'^k \in y^m.
\end{array}$$

The last clause is obviously not a Horn clause. However, we may define a weak union that covers the exact one:

$$\begin{array}{c}
x^m \subseteq x^m \cup y^m \\
y^m \subseteq x^m \cup y^m \\
x^m \subseteq z^m \wedge y^m \subseteq z^m \Rightarrow x^m \cup y^m \subseteq z^m.
\end{array}$$

We may conclude from this short outline that the set theory that can be described by R^n-and G^n-logics is rather weak, due to the absence of negation. Negation must be avoided as long as we want initial models. A limited amount of negation can be used by admitting goal clauses, which are Horn clauses without conclusion. The existence of initial models is then guaranteed if the presentation is consistent. In [11], we extended the completeness results for consistency tests in form of saturation procedures to R^n-and G^n-logics, as illustrated in Section 6.

Order-sorted algebras, ETL and Unified Algebras: From an algebraic point of view, R^n-and G^n-logics compare best with many and order-sorted algebras [7], already implemented via rewriting for instance in OBJ-3 [15]. However simple G^n-logics provide arbitrary terms as sorts and thus achieve a greater expressivity. Polymorphic order-sorted algebras can be seen as a fragment of simple G^1 logic.

In order to compare R^n-logics with ETL and unified algebras, we tried to encode them in our framework. ETL [19] is in fact a fragment of R^n-logics. An ETL presentation is a triple $\langle \Omega, V, E \rangle$, such that Ω is a set of function symbols (with associated arity), V is a set of unsorted variables and E is a set of Ω-Horn clauses using only equality "$=$" and the typing relation ":" as binary operators. The typing relation satisfies the paramodulation axiom and therefore we may use it as a new relation symbol in R^n-logics. Remark that we cannot reuse "$\in$" for this purpose, since it has more properties than ":" in ETL. Now, it is possible to construct an R^n-presentation, such that an Ω-atom is true in R^n-logic if and only if it holds in ETL [11].

Concerning unified algebras [25], the main difference is the absence of the empty set, which should take the role of the bottom element in unified algebra. As mentioned above, we cannot allow for the empty set if we want to have initial models for all presentations. Extending our formula language by goals, it is actually possible to extend R^n-and G^n-logics by a predefined constant representing the empty set. However, *choose* has then to become partial. The problem with singletons, which are necessary for the relation ":" in unified algebras, cannot be solved as easily. A work-around is the use of quasi-singletons defined by the axioms for *sgl* given above, after replacing the variable z'^m of sort s'_m by one of sort s_m. Then, $sgl(x)$ is a set with exactly one standard element, namely x, but arbitrarily many non-standard elements. To cope with real singletons, we would need to extend our type theory in order to avoid that *choose* becomes deterministic, which is in conflict with the sort preservation property of R^n-and G^n-logic presentations and deduction rules. The other operators of unified algebras, lower bound $\leq$, join $|$ and meet &, can be realised by set inclusion, intersection and weak union, as defined above, so that we may come quite close to unified algebras at least. Simple R^n-logics have also strong similarities with power algebras, i.e. unified algebras with set interpretations [25].

Relation to Higher-Order Logic and Algebras: The framework of R^n- and G^n-logics provides some higher-order features, since function graphs are specified as set constants, which can be passed to other functions as higher-order arguments. This can be situated in the context of higher-order logic that provides variables for subsets, relations, functions on the universe, functions defined on functions and quantification over these. An introductory survey to the literature on higher-order logic, its relation to set theory, and reduction to first-order logic can be found in [36].

During the last years, a number of papers have dealt with the extension of first-order algebraic specifications to higher-order ones. Among them are [18, 32, 22, 24, 23, 21]. We share with these approaches the objectives of integrating higher-order function space with algebraic specifications, and keeping the existence of initial (and possibly terminal) models. But we differ from universal algebras with higher-order types as developed for instance in [21] in the fact that our models are set-theoretic rather than purely algebraic, and we so provide a uniform treatment of types and higher-order functions.

Relation to λ-calculi and higher-order rewriting: As in higher-order algebras, we do not allow λ-abstractions as terms, but rather consider that $\lambda x.t$ can always be replaced by a new constant symbol f together with the axiom $f(x) = t$. This has the advantage to avoid technical problems associated with binding mechanism and to minimize the functions in the initial model, which may be crucial for limiting the search space for automated theorem proving.

Polymorphically typed λ-calculi have shown that types and subtypes provide interesting features for functional programming and some connections can be drawn also from our work to the system $F_{\leq}$ of [3] and to dynamic types proposed in [1].

Concerning the operationalisation of deduction, we have a purely first-order mechanism and avoid higher-order rewriting or unification as developed in [28] as well as combinatory reduction systems [16]. In contrast, in the theorem proving domain, HOL [8] and Isabelle [31] are based on type theory but provide means to reason in set theory.

Relation to object-oriented languages: Through their use of sets, both for types and for higher-order functions, R^n-and G^n-logics are close to the semantics of imperative programming languages with subtyping, inheritance and polymorphism, like C++. In the latter language, two types declared to hold exactly the same objects have to be different. Furthermore, it is possible to compare function symbols. Such a test has to fail for two functions defined in exactly the same way but with different names. In analogy, we cannot derive $A = B$ from $\{a \in A, a \in B\}$ in R^n-and G^n-logics. Although R^n-and G^n-logics do not provide any built-in mechanism for features or records, extensions in this direction could be done.

Acknowledgements: We are grateful to anonymous referees for pointing out missing references in the original version of this paper. This work was partially supported by the Esprit Basic Research Working Group 6112 Compass.

References

1. M. Abadi, L. Cardelli, B. Pierce, and G. Plotkin. Dynamic typing in a statically typed language. *ACM Transactions on Programming Languages and Systems*, 13(2):237–268, Apr. 1991.
2. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. In *Proceedings 11th International Conference on Automated Deduction, Saratoga Springs (N.Y., USA)*, pages 462–476, 1992.

3. P.-L. Curien and G. Ghelli. Coherence of subsumption, minimum typing and type-checking in $F_{\leq}$. *Mathematical Structures in Computer Science*, 2(1):55–91, 1991.
4. N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
5. N. Dershowitz and M. Okada. A rationale for conditional equational programming. *Theoretical Computer Science*, 75:111–138, 1990.
6. M. Fourman and D. Scott. Sheaves and logic. In M. Fourman and C. Mulvey, editors, *Applications of Sheaves*, volume 753 of *Lecture Notes in Mathematics*, pages 302–401. Springer-Verlag, 1979.
7. J. A. Goguen and J. Meseguer. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 2(105):217–273, 1992.
8. M. J. C. Gordon and T. F. Melham. *Introduction to HOL: a theorem proving environment for higher order logic.* Cambridge University Press, 1993.
9. L. Henkin. Completeness in the theory of types. *The Journal of Symbolic Logic*, 15(2):81–91, June 1950.
10. D. Hilbert and P. Bernays. *Grundlagen der Mathematik*, volume 2. Berlin. Reprinted Ann Arbor, Mich., 1944, 1939.
11. C. Hintermeier. *Déduction avec sortes ordonnées et égalités.* Thèse de Doctorat d'Université, Université Henri Poincaré – Nancy 1, Oct. 1995.
12. C. Hintermeier, C. Kirchner, and H. Kirchner. Dynamically-typed computations for order-sorted equational presentations –extended abstract–. In S. Abiteboul and E. Shamir, editors, *Proc. 21st International Colloquium on Automata, Languages, and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 450–461. Springer-Verlag, 1994.
13. C. Hintermeier, C. Kirchner, and H. Kirchner. Sort inheritance for order-sorted equational presentations. In *Recent Trends in Data Types Specification*, volume 906 of *Lecture Notes in Computer Science*, pages 319–335. Springer-Verlag, 1995.
14. S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path ordering. Unpublished manuscript, 1980.
15. C. Kirchner, H. Kirchner, and J. Meseguer. Operational semantics of OBJ-3. In *Proceedings of 15th International Colloquium on Automata, Languages and Programming*, volume 317 of *Lecture Notes in Computer Science*, pages 287–301. Springer-Verlag, 1988.
16. J. W. Klop. *Combinatory Reduction Systems.* PhD thesis, CWI, 1980.
17. A. Leisenring. *Mathematical logic and Hilbert's ϵ-symbol.* University Mathematical Series, Bedford College, London, 1969.
18. T. Maibaum and C. Lucena. Higher order data types. *International Journal of Computer and Information Sciences*, 9:31–53, 1980.
19. V. Manca, A. Salibra, and G. Scollo. Equational type logic. *Theoretical Computer Science*, 77(1-2):131–159, 1990.
20. A. Mégrelis. *Algèbre galactique — Un procédé de calcul formel, relatif aux semi-fonctions, à l'inclusion et à l'égalité.* Thèse de Doctorat d'Université, Université Henri Poincaré – Nancy 1, 1990.
21. K. Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100:385–417, 1992.
22. B. Möller. Algebraic specifications with high-order operators. In L. Meertens, editor, *Proceedings IFIP TC2 Working Conf. on Program Specification and Transformation*, pages 367–392. IFIP, Elsevier Science Publishers B. V. (North-Holland), 1987.

23. B. Möller, A. Tarlecki, and M. Wirsing. Algebraic specification with built-in domain constructions. In M. Dauchet and M. Nivat, editors, *Proceedings of CAAP'88*, Lecture Notes in Computer Science, pages 132–148. Springer-Verlag, 1988.
24. B. Möller, A. Tarlecki, and M. Wirsing. Algebraic specifications or reachable higher-order algebras. In D. Sannella and A. Tarlecki, editors, *Recent Trends in Data Type Specification*, volume 332 of *Lecture Notes in Computer Science*, pages 154–169. Springer-Verlag, 1988.
25. P. D. Mosses. Unified algebras and institutions. In *Proceedings 4th IEEE Symposium on Logic in Computer Science, Pacific Grove*, pages 304–312, 1989.
26. R. Nieuwenhuis and A. Rubio. Basic superposition is complete. In B. Krieg-Brückner, editor, *Proceedings of ESOP'92*, volume 582 of *Lecture Notes in Computer Science*, pages 371–389. Springer-Verlag, 1992.
27. R. Nieuwenhuis and A. Rubio. Theorem proving with ordering constrained clauses. In D. Kapur, editor, *Proceedings 11th International Conference on Automated Deduction, Saratoga Springs (N.Y., USA)*, volume 607 of *Lecture Notes in Computer Science*, pages 477–491. Springer-Verlag, 1992.
28. T. Nipkow. Higher-order critical pairs. In *Proc. 6th IEEE symposium on Logic in Computer Science (LICS)*, pages 342–349. IEEE Computer Society Press, Los Alamitos, July 1991.
29. P. Nivela and R. Nieuwenhuis. Saturation of first-order (constrained) clauses with the *saturate* system. In C. Kirchner, editor, *Proceedings 5th Conference on Rewriting Techniques and Applications, Montreal (Canada)*, volume 690 of *Lecture Notes in Computer Science*, pages 436–440. Springer-Verlag, 1993.
30. A. Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik. *Math. Annalen*, 145(1):297–333, 1962.
31. L. C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5:363–397, 1989.
32. A. Poigné. Partial algebras, subsorting and dependent types – prerequisites of error handling in algebraic specifications. In *Proceedings of Workshop on Abstract Data Types*, volume 332 of *Lecture Notes in Computer Science*. Springer-Verlag, 1988.
33. D. Scott. Data types as lattices. *SIAM Journal of Computing*, 5(3):522–587, 1976.
34. G. Smolka, W. Nutt, J. A. Goguen, and J. Meseguer. Order-sorted equational computation. In H. Aït-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures, Volume 2: Rewriting Techniques*, pages 297–367. Academic Press inc., 1989.
35. J. M. Spivey. *Understanding Z: a specification language and its formal semantics*. Cambridge University Press, 1988.
36. J. Van Benthem and K. Doets. Higher-Order Logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 1, pages 275–329. Reidel Publishing Company, 1983.
37. A. N. Whitehead and B. Russell. *Principia Mathematica*, volume 1. Cambridge University Press, Cambridge, MA, 1925.

Addendum: This paper also appears in *Proceedings of HOA'95, 2nd Int. Workshop on Higher-Order Algebra, Logic, and Term-Rewriting, Paderborn, Germany, 1995*, volume 1074 of *Lecture Notes in Computer Science*, pages 90–108. Springer-Verlag, 1996.