UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY

Higher-Order Order-Sorted Resolution

Michael Kohlhase

SEKI Report SR-94-01

SEKI-Report

Higher-Order Order-Sorted Resolution

Michael Kohlhase*
Fachbereich Informatik, Universität des Saarlandes
66041 Saarbrücken, Germany
+49-681-301-4627
{kohlhase}@cs.uni-sb.de

January 1994

Abstract

The introduction of sorts to first-order automated deduction has brought greater conciseness of representation and a considerable gain in efficiency by reducing the search space. It is therefore promising to treat sorts in higher order theorem proving as well.

In this paper we present a generalization of Huet's Constrained Resolution to an order-sorted type theory $\Sigma\mathcal{T}$ with term declarations. This system builds certain taxonomic axioms into the unification and conducts reasoning with them in a controlled way. We make this notion precise by giving a relativization operator that totally and faithfully encodes $\Sigma\mathcal{T}$ into simple type theory.

*This work was supported by the "Sonderforschungsbereich 314, Künstliche Intelligenz" of the Deutsche Forschungsgemeinschaft (DFG) and the "Studienstiftung des deutschen Volkes".

1 Introduction

In the quest for calculi best suited for automating logic on computers, the introduction of sorts has been one of the most important contributions. Sort techniques consist in syntactically distinguishing between objects of different classes and then assigning sorts (specifying the membership in some class) to objects and restrict the range of variables to particular sorts. Since a good part of the set membership and subset information can be coded into the sorted signature, sorted logics lead a more concise representation of problems and proofs than the un-sorted variants. The exploitation of this information during proof search can dramatically reduce the search space associated with theorem-proving and hence the resulting sorted calculi are much more efficient for deduction purposes. In the context of first-order logic sort information has been successfully employed by A. Oberschelp [19], C. Walther [26], M. Schmidt-Schauß [22], A. G. Cohn [8], and others.

On the other hand there is an increasing interest in deduction systems for higher-order logic, since many problems in mathematics are inherently higher order. Current automated deduction systems for higher-order logic like TPS [4] are rather weak on the first-order fragment.

The question about the behavior of higher-order logic under the constraints of a full order-sorted type structure is a natural one to ask, in particular since calculi in this system promise the development of more powerful deduction systems for real mathematics. G. Huet proposed the study of a simple version of order-sorted type theory[1] in an appendix to [9]. The unification problem in extensions of this system have since been studied by Nipkow and Qian [18] and Pfenning and the author [16]. Furthermore typed λ-calculi with order-sorted type structures have been of interest in the programming language community as a theoretical basis for object-oriented programming and for more expressive formalisms for higher-order algebraic specifications [21, 6, 5, 20]. We will use the term "type theory" for the simply typed λ-calculus with logical axioms just as Church did in his original paper [7].

In [14] we have proposed a rich system $\Sigma\mathcal{T}$ of order-sorted type theory with term declarations and order-sorted function universes, for which we presented unification and pre-unification algorithms. In this paper we present a corrected system and complete the enterprise of building order-sorted higher-order deduction systems by presenting a resolution calculus for $\Sigma\mathcal{T}$ that uses the unification and pre-unification algorithms from [14] as a basis. For details and proofs we refer the reader to [15] and [10], where we treat a subsystem.

In the following we will shortly motivate the primary features of $\Sigma\mathcal{T}$. In unsorted logics the only way to express the knowledge that an object is a member of a certain

[1] However Huet only allows order-sorting the universe of individuals

class of objects is through the use of unary predicates, such as the predicate $\mathfrak{N}_{\iota\to o}$ in the formulae $(\mathfrak{N}2_\iota)$, *i.e.* "2 is a natural number", or $\neg(\mathfrak{N}\text{Peter}_\iota)$, *i.e.* "Peter is not a natural number". This leads to a multitude of unit clauses $(\mathfrak{S}_{\iota\to o}\mathbf{A})$ in the deduction that only carry the sort information for $\mathbf{A}$. Since quantification is unrestricted in unsorted logics, the restricted quantification has to be simulated by formulae like $\forall X_\iota(\mathfrak{N}X_\iota) \Rightarrow (\geq_{\iota\to\iota\to o} X_\iota 0_\iota)$. This approach is unsatisfactory because inter alia the derivation of the non-sensical formula $(\mathfrak{N}\text{Peter}) \Rightarrow (\geq \text{Peter } 0)$ is permitted, even though $(\geq \text{Peter } 0)$ can never be derived because of $\neg(\mathfrak{N}\text{Peter})$.

In type theory the idea of associating sort information is all the more natural, as all objects are already typed, which amounts to a – very coarse – division of the universe into classes of the same type. The simple type system is merely refined by considering the sort symbols as additional base type symbols, so that, for example, the last formula above would read $\forall X_{\mathfrak{n}}.(\geq_{\mathfrak{n}\to\mathfrak{n}\to o} X_{\mathfrak{n}} 0_{\mathfrak{n}})$. Sorting the universe of individuals gives rise to new classes of functions, namely functions the domains and codomains of which are just the sorts. In addition to this essentially first-order way of sorting the function universes, the classes of functions defined by domains and codomains can be further divided into subclasses, since functions are explicit objects of type theory. This possibilities modeled by sort symbols of functional type, *i.e.* base sort symbols that denote classes of functions.

In $\Sigma\mathcal{T}$ we relax the implicit condition that only the sorts of constants and variables can be declared, and allow declaration of the form $[\forall[X^1 :: \mathfrak{A}^1], \ldots, [X^n :: \mathfrak{A}^n]\mathbf{A} :: \mathfrak{A}]$ called term declarations, where $\mathbf{A}$ can be an arbitrary formula of appropriate type. Thus we obtain a more expressive system, in which term declarations of the form $[\forall(X :: \mathfrak{A}).X :: \mathfrak{B}]$ entail that $\mathfrak{A}$ is a subsort of $\mathfrak{B}$, thus giving rise to the name "order-sorted type theory", since the subsort relation induced by these declarations turns out to be a partial ordering relation. These subsort declarations induce the intended partial ordering on the set of sort symbols by covariance in the codomain sort and natural inclusion of the base sorts. The term declarations restrict the class of models for the logic, and so the well-formed formulae have to meet certain restrictions to denote meaningful objects, that is, formulae have to be well-sorted. For instance, for any application $(\mathbf{AB})$ in type theory, there must be sort symbols $\mathfrak{A}$ and $\mathfrak{B}$, such that $\mathbf{A}$ is of sort $\mathfrak{A}$, $\mathbf{B}$ is of sort $\mathfrak{B}$ and $\mathfrak{B}$ is a subsort of the domain sort of $\mathfrak{A}$. The sort of the application $(\mathbf{AB})$ is defined to be the codomain sort of $\mathfrak{A}$.

2 Order-Sorted Type Theory

We assume the reader to be familiar with the syntax and semantics of simple type theory, as presented e.g. in [7, 2]. We will proceed by giving the relevant definitions of [14] for the order-sorted type theory $\Sigma\mathcal{T}$, which we have motivated in the

introduction.

Definition 2.1 (Sort System) A *sort system* is a quintuple $(\mathcal{S}, \mathcal{S}_0, \mathfrak{r}, \mathfrak{d}, \tau)$, where $\mathcal{S}_0$ is a finite set of symbols, called *base sort symbols*, which we assume to contain the symbol $\mathfrak{o}$ for the sort of truth values but not type symbols o and ι. like $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$, $\mathfrak{D}$,....We will denote sort symbols by uppercase Gothic letters. The *set of sort symbols* $\mathcal{S}$ is the closure $\mathcal{S}_0$ under the function construction that is, $\mathfrak{B} \to \mathfrak{A} \in \mathcal{S}$, whenever $\mathfrak{A}, \mathfrak{B} \in \mathcal{S}$. The functions $\mathfrak{r}, \mathfrak{d}$ and τ specify the sorts of the codomain, domain and the type of a sort symbol. Furthermore we will often use the shorthands $\mathfrak{d}^k(\mathfrak{A})$ and $\mathfrak{r}^k(\mathfrak{A})$ for the k^{th} *domain sort* and the *k-fold codomain sort of* $\mathfrak{A}$, which we define by

$$\begin{array}{lcllcl} \mathfrak{r}^0(\mathfrak{A}) & = & \mathfrak{A} & \mathfrak{r}^{i+1}(\mathfrak{A}) & = & \mathfrak{r}(\mathfrak{r}^i(\mathfrak{A})) \\ \mathfrak{d}^0(\mathfrak{A}) & = & \mathfrak{A} & \mathfrak{d}^{i+1}(\mathfrak{A}) & = & \mathfrak{d}(\mathfrak{r}^i(\mathfrak{A})) \end{array}$$

It will be important that the signatures over which our well-sorted terms are built "respect function domains," *i.e.*, that for any term $\mathbf{A}$ and any sorts $\mathfrak{A}$ and $\mathfrak{B}$ of $\mathbf{A}$, the identity $\mathfrak{d}(\mathfrak{A}) = \mathfrak{d}(\mathfrak{B})$ holds. The proof that signatures indeed satisfy this property depends on the consistency conditions for valid signatures, given in terms of the equivalence relation $\mathbf{Rdom}$, where $\mathfrak{A}\mathbf{Rdom}\mathfrak{B}$ if either $\mathfrak{A}, \mathfrak{B} \in \mathcal{S}^{nf}$ and $\tau(\mathfrak{A}) = \tau(\mathfrak{B})$ or $\mathfrak{r}(\mathfrak{A})\mathbf{Rdom}\mathfrak{r}(\mathfrak{B})$ and $\mathfrak{d}(\mathfrak{A}) \doteq \mathfrak{d}(\mathfrak{B})$.

Next, we will introduce the concept of well-sortedness for well-formed formulae. A term $\mathbf{A}$ will be well-sorted with respect to a signature Σ and a context Γ, if the judgement $\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A}$ is derivable in the inference system $\Sigma\mathcal{T}$. Here the context gives local sort information for the variables, whereas the signature contains sort information for term schemata (term declarations). One of the difficulties in devising a formal system with term declarations is that the signature needed for defining well-sortedness in itself contains terms that have to be well sorted. Therefore we need to combine the inference systems for valid signatures $\vdash_{\text{sig}} \Sigma$ and that for well-sortedness into one large system $\Sigma\mathcal{T}$. Another difficulty is that we also have to treat $\beta\eta$-conversion $\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ in $\Sigma\mathcal{T}$, since we want $\beta\eta$-conversion to be sort preserving, because otherwise conversion might increase the sort of a subterm and thereby possibly convert a well-sorted term to an ill sorted one.

Definition 2.2 Let X_α be a variable and $\mathfrak{A}$ a sort symbol, then we call a pair $[X :: \mathfrak{A}]$ a *variable declaration* for X, iff $\tau(\mathfrak{A}) = \alpha$. We call a sequence of variable declarations a *context* and write $\Gamma(X) := \mathfrak{A}$, iff $[X :: \mathfrak{A}]$ is the rightmost declaration for X in Γ, we will often indicate this situation by writing the sort $\mathfrak{A}$ in the index of $X_{\mathfrak{A}}$.

Definition 2.3 (Valid Signatures and Well-Sorted Formulae) We first present the inference system for valid signatures; for this we assume the existence of typed collection $\overline{\Sigma}_\mathcal{T}$ of constant symbols. Now a signature is a set of term declarations (triples

of the form $[\forall \Gamma.\mathbf{A} :: \mathfrak{A}]$) that obey certain restrictions which are verified by the inference system $\Sigma\mathcal{T}$ given below. The idea of term declarations is that there can be sort information within the structure of a term, if the term matches a certain schematic term (a term declaration). Consider e.g. the addition function $+ :: \mathfrak{N} \to \mathfrak{N} \to \mathfrak{N}$ and $+ : \mathfrak{E} \to \mathfrak{E} \to \mathfrak{E}$ on the natural numbers $\mathfrak{N}$ and the evens $\mathfrak{E}$ respectively, then the expression $(+aa)$ is an even, even if a is not. This information can be formalized by declaring the term $(+X_{\mathfrak{N}}X_{\mathfrak{N}})$, to be of sort $\mathfrak{E}$ (an even number) using a term declaration $[\forall[X :: \mathfrak{N}].+XX :: \mathfrak{E}]$.

$$\frac{\dfrac{}{\vdash_{\mathrm{sig}} \emptyset}}{\dfrac{\vdash_{\mathrm{sig}} \Sigma \quad c \notin \Sigma \quad c \in \overline{\Sigma}_\alpha \qquad \mathfrak{A} \in \mathcal{S} \quad \tau(\mathfrak{A}) = \alpha}{\vdash_{\mathrm{sig}} \Sigma, [c :: \mathfrak{A}]}}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A} \qquad \Sigma \vdash \mathfrak{A}\mathbf{Rdom}\mathfrak{B}}{\vdash_{\mathrm{sig}} \Sigma, [\forall \Gamma.\mathbf{A} :: \mathfrak{B}]}$$

Theses inference rules allow to add term declarations to valid signatures, if either they are the first declarations for new constants, or if the formula $\mathbf{A}$ is well-sorted and the new sort $\mathfrak{A}$ respects function domains. The last inference rule needs the judgement of well-sorted formulae, which we will define next set of inference rules:

$$\frac{\Gamma(X) = \mathfrak{A}}{\Gamma \vdash_\Sigma X :: \mathfrak{A}} \qquad \frac{[\forall \Delta.\mathbf{A} :: \mathfrak{A}] \in \Sigma \qquad \vdash_{\mathrm{sig}} \Sigma \quad \Delta \subseteq \Gamma}{\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A}}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A} \qquad \Delta \vdash_\Sigma \mathbf{B} :: \mathfrak{d}(\mathfrak{A})}{\Delta \cup \Gamma \vdash_\Sigma (\mathbf{AB}) :: \mathfrak{r}(\mathfrak{A})} \qquad \frac{\Gamma, X :: \mathfrak{B} \vdash_\Sigma \mathbf{A} :: \mathfrak{A}}{\Gamma \vdash_\Sigma (\lambda X :: \mathfrak{B}.\mathbf{A}) :: \mathfrak{B} \to \mathfrak{A}}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A} \qquad \Delta \vdash_\Sigma \mathbf{B} :: \mathfrak{B} \quad \Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B} \quad \Delta \subseteq \Gamma}{\Gamma \vdash_\Sigma \mathbf{B} :: \mathfrak{A}}$$

For a fixed signature Σ and a context Γ we say that a formula $\mathbf{A}$ *is of sort* $\mathfrak{A}$, iff $\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A}$ and we denote the set of well-sorted formulae of sort $\mathfrak{A}$ by $wsf_{\mathfrak{A}}(\Sigma, \Gamma)$.

Note that the rules for variables, application and abstraction are the obvious generalizations of the corresponding rules for simple type theory. In the setting with term declarations we do not need a separate rule for constants, since all constants have to be declared in term declarations.

Now we only have to define the judgements for order-sorted $\beta\eta$ equality needed in the last inference rule. Let $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$ be the congruence judgement induced by the reduction judgement.

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A}}{\Gamma \vdash_\Sigma (\lambda X :: \mathfrak{d}(\mathfrak{A}).\mathbf{A}X) \longrightarrow^t_\eta \mathbf{A}} \qquad \frac{\Gamma, X :: \mathfrak{B} \vdash_\Sigma \mathbf{A} :: \mathfrak{A} \qquad \Gamma \vdash_\Sigma \mathbf{B} :: \mathfrak{B}}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{A})\mathbf{B} \longrightarrow^t_\beta [\mathbf{B}/X]\mathbf{A}}$$

In the definition of order-sorted η-reduction we have taken care to identify the (unique) supporting sort $\mathfrak{d}(\mathfrak{A})$ of $\mathbf{A}$, since the formula $\lambda X :: \mathfrak{B}\mathbf{A}X$ denotes the restriction of the function $\mathbf{A}$ to sort $\mathfrak{B}$, if $\mathfrak{B}$ is a subsort of $\mathfrak{d}(\mathfrak{A})$.

It is easy to see that the judgements defined above respect well-typedness, *i.e.* that the information described by $\Sigma\mathcal{T}$ merely refined the type information. In particular sorted $\beta\eta$-conversion is a sub-relation of well-typed conversion and well-sorted terms are well-typed. Furthermore, if we only have one base sort per base type, then $wsf(\Sigma, \Gamma)$ is isomorphic to the set of well-typed formulae, therefore $\Sigma\mathcal{T}$ is a generalization of $\mathcal{Q}$.

It is an important property of our system, that any valid signature is *subterm-closed*, that is each subterm of a well-sorted term is again well-sorted. This fact is natural, since it does not make sense to allow ill-formed subexpressions in well-formed expressions. Now we see why we had to require the term $\mathbf{B}$ to be well-sorted in the inference rule for well-sorted β-reduction, since otherwise a term $\mathbf{B} := (\lambda X :: \mathfrak{A}.c)\mathbf{D}$ would be well-sorted for arbitrary well-typed terms $\mathbf{D}$ whenever c is a well-sorted constant and then our system would not be subterm-closed any more.

Definition 2.4 (well-sorted substitution) A substitution σ is called a *well-sorted substitution (Σ-substitution) in context* Γ, iff the judgement $\Sigma \vdash \sigma :: \Gamma$ is derivable in the following inference system.

$$\frac{}{\Sigma \vdash \emptyset :: \Gamma} \qquad \frac{\Sigma \vdash \sigma :: \Gamma \qquad \Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A}}{\Sigma \vdash \sigma, [\mathbf{A}/X] :: \Gamma[X :: \mathfrak{A}],}$$

In other words, if $\sigma := [\mathbf{A}^i/X^i]$ and $\Gamma(X^i) = \mathfrak{A}^i$, then $\Gamma \vdash_\Sigma \mathbf{A}^i :: \mathfrak{A}^i$. The set of well-sorted substitutions is denoted by $\mathbf{wsSub}(\Gamma, \Sigma)$.

We can show that if $\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A}$ and $\Sigma \vdash \sigma :: \Gamma$, then $\Gamma \vdash_\Sigma \sigma(\mathbf{A}) :: \mathfrak{A}$. Thus application of well-sorted substitution conserves the property of well-sortedness. Thus this notion of well-sorted substitution is the appropriate for our purposes.

Remark 2.5 (Semantics) We adapt the usual set theoretic semantics of simple type theory (see e.g. [2]) by allowing universes of partial functions $(\overline{\mathcal{D}}_\alpha)_{\alpha \in \mathcal{T}}$ and fixing subsets $\mathcal{D}_\mathfrak{A} \subseteq \mathcal{D}_{\tau(\mathfrak{A})}$ for all sort symbols $\mathfrak{A}$, such that for all functional sorts $\mathfrak{B}$ and all $f \in \mathcal{D}_\mathfrak{B}$ is total on $\mathcal{D}_{\mathfrak{d}(\mathfrak{B})}$ with codomain in $\mathcal{D}_{\mathfrak{r}(\mathfrak{B})}$. We get the appropriate class of models $(\mathcal{D}, \mathcal{I})$ for $\Sigma\mathcal{T}$ by requiring that the homomorphic extension $\mathcal{I}_\varphi$ of the interpretation $\mathcal{I}$ maps a formula $\mathbf{A}$ into $\mathcal{D}_\mathfrak{A}$, whenever $[\mathbf{A} :: \mathfrak{A}]$ is a term declaration and that that $\mathcal{I}$ maps $=^\mathfrak{A}$ to the identity relation on $\mathcal{D}_\mathfrak{A}$.

Note that this only describes the intended semantics of $\Sigma\mathcal{T}$, since this semantics is extensional we will not be able to show completeness of our resolution calculus with

respect to this class of models. In fact our calculus fails to capture extensionality in exactly the same way as the unsorted version, therefore we can only hope for a relative completeness theorem like 3.10.

It is important to our program of developing an order-sorted calculus for higher-order automated theorem proving, that the fundamental operations of the calculus do not allow the formation of ill-sorted terms from well-sorted ones. This will ensure that our calculus never has to handle ill-sorted terms, even intermediately. In fact we can show that if $\Gamma \vdash_\Sigma \mathbf{A}=_{\beta\eta}\mathbf{B}$, then $\mathbf{A}$ and $\mathbf{B}$ have the same set of sorts. This fact can be used to to show that order-sorted reduction is strongly normalizing and all results carry over from the typed case. Termination is a consequence of the fact that order-sorted $\beta\eta$-reduction is a sub-relation of unsorted $\beta\eta$-reduction, which is known to be terminating for typed λ-calculi. For the confluence result we need that well-sorted functional formulae have unique domain sorts, which is a consequence of the fact that if a well-sorted formula $\mathbf{A}$ has sorts $\mathfrak{A}$ and $\mathfrak{B}$, then $\mathfrak{A}\mathbf{Rdom}\mathfrak{B}$. In fact the formal system $\Sigma\mathcal{T}$ is designed to capture informal mathematical practice, where functions have unique domains associated with them. This fact a posteriori justifies our definition of order-sorted η-conversion and the **Rdom**-restrictions in the inference rules for valid signatures.

We observe that if $\Gamma \vdash_\Sigma X :: \mathfrak{B}$ for a context Γ with $\Gamma(X) = :: \mathfrak{A}$, then $\mathcal{D}_\mathfrak{A} \subseteq \mathcal{D}_\mathfrak{B}$. This is just the situation that is traditionally captured with the notion of sort inclusion in sorted logics and written as $\Gamma \vdash_\Sigma \mathfrak{A} \leq_\Sigma \mathfrak{B}$, where $\leq_\Sigma$ is the smallest partial ordering that contains the subsort declarations, *i.e.* term declarations of the form $[\forall X :: \mathfrak{A}.X :: \mathfrak{B}]$. In our system $\Sigma\mathcal{T}$ we do not have to take sort inclusion as primitive, since this judgement is derivable in $\Sigma\mathcal{T}$.

Definition 2.6 (Sort Inclusion) Term declarations of the forms $[\forall[X :: \mathfrak{A}].X :: \mathfrak{B}]$ are called *subsort declarations* and are abbreviated by $[\mathfrak{A} \leq_\Sigma \mathfrak{B}]$.

$$\frac{[\forall X :: \mathfrak{A}.X :: \mathfrak{B}] \in \Sigma \qquad \vdash_{\mathrm{sig}} \Sigma}{\Sigma \vdash \mathfrak{B} \leq \mathfrak{A}} \qquad \frac{\vdash_{\mathrm{sig}} \Sigma}{\Sigma \vdash \mathfrak{A} \leq \mathfrak{A}}$$

$$\frac{\Sigma \vdash \mathfrak{A} \leq \mathfrak{B} \qquad \Sigma \vdash \mathfrak{B} \leq \mathfrak{C}}{\Sigma \vdash \mathfrak{A} \leq \mathfrak{C}}$$

$$\frac{\vdash_{\mathrm{sig}} \Sigma}{\Sigma \vdash \mathfrak{A} \leq \mathfrak{d}(\mathfrak{A}) \to \mathfrak{r}(\mathfrak{A})} \qquad \frac{\Sigma \vdash \mathfrak{A} \leq \mathfrak{B}}{\Sigma \vdash \mathfrak{C} \to \mathfrak{A} \leq \mathfrak{C} \to \mathfrak{B}}$$

The first set of inference rules extend the declarations to a partial ordering relation, whereas the last two rules propagate subsort information to function sorts by "natural inclusion of function sorts" and "covariance in the codomain sort". Since our signatures respect function domains the widely studied propagation schema [18, 16]

of "contravariance in the domain sort" cannot be a admissible in our system. We do not need this rule, which semantically corresponds to function restriction, since we can treat that explicitly as we have argued in the context of η-conversion.

The subsort judgement interacts with well-sorted formulae by the traditional *weakening rule*, which allows to weaken the sort information.

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{A} \qquad \Sigma \vdash \mathfrak{A} \leq \mathfrak{B}}{\Gamma \vdash_\Sigma \mathbf{A} :: \mathfrak{B}}$$

We can show that $[X :: \mathfrak{A}] \in \Gamma \vdash_\Sigma X :: \mathfrak{B}$, if $\Sigma \vdash \mathfrak{A} \leq \mathfrak{B}$, and therefore the above inference rules for the subsort judgement are admissible in $\Sigma\mathcal{T}$. Thus we have recovered the mechanism for subsorting from the mechanism of term declarations.

Example 2.7 Let Σ be a higher-order signature with $\mathcal{S}_0 := \{\mathfrak{R}, \mathfrak{C}, \mathfrak{D}, \mathfrak{P}\mathfrak{o}\}$ where the intended meaning $\mathfrak{R}$ is the set of real numbers, $\mathfrak{C}$: continuous functions, $\mathfrak{D}$: differentiable functions and $\mathfrak{P}$: polynomials. Therefore the types are $\tau(\mathfrak{R}) = \iota$, $\tau(\mathfrak{C}) = \tau(\mathfrak{D}) = \tau(\mathfrak{P}) = \iota \to \iota$ and $\mathfrak{r}(\mathfrak{C}) = \mathfrak{d}(\mathfrak{C}) = \mathfrak{R}, \ldots$. In this example we want to model a taxonomy for elementary calculus, so let Σ be the set containing the subsort declarations $[\forall[X :: \mathfrak{C}.X :: \mathfrak{D}], [\forall[X :: \mathfrak{D}].X :: \mathfrak{P}]$, and the term declarations

$$[\lambda X_{\mathfrak{R}}.X :: \mathfrak{P}],\ [\lambda X_{\mathfrak{R}}.Y_{\mathfrak{R}} :: \mathfrak{P}],\ [\lambda X_{\mathfrak{R}}. + (F_{\mathfrak{P}}X)(G_{\mathfrak{P}}X) :: \mathfrak{P}),\ [\lambda X_{\mathfrak{R}}. * (F_{\mathfrak{P}}X)(G_{\mathfrak{P}}X) :: \mathfrak{P}]$$

for polynomials and furthermore $[\partial :: \mathfrak{D} \to \mathfrak{C}], [\partial :: \mathfrak{P} \to \mathfrak{P}]$ for the differentiation operator ∂, then Σ is a valid signature. We can see that we have coded a great deal of of information about polynomials and differentiation into the term declarations of Σ, that can be used in the unification during proof search in our resolution calculus. For instance we can check (up to arithmetic) that any real polynomial is indeed of sort $\mathfrak{P}$.

3 A Resolution Calculus for $\Sigma\mathcal{T}$

In this section we will present a variant of Huet's *Constrained Resolution* calculus [9] and prove it correct and complete relative to a Hilbert-style calculus $\mathfrak{T}$.

General Assumption 3.1 (Sorted Logical Constants) For a Resolution calculus we have to assume declarations for the logical constants in the signature. Thus we always assume that our signatures contain the declarations for conjunction and negation $[\wedge :: \mathfrak{o} \to \mathfrak{o} \to \mathfrak{o}], \neg :: \to \mathfrak{o} \to \mathfrak{o}]$ and the quantification constants $[\Pi^{\mathfrak{A}} :: (\mathfrak{A} \to \mathfrak{o}) \to \mathfrak{o}]$ for all $\mathfrak{A} \in \mathcal{S}$. We assume all other logical constants to be defined from these in the usual way.

This is a problem at first glance, since this set of declarations is infinite, which is not warranted by our definitions. However we can always add the $\Pi^{\mathfrak{A}}$-declarations to valid signatures, since they are new constants. In any proof situation we only have finitely many quantors, so we can always chose the signature sufficiently large to contain all necessary declarations, so we can treat the infinite signature as an abbreviation of all needed subsets. We can proceed analogously for the skolem constants needed in the computation of the clause normal form.

For the semantics we assume that $\mathcal{I}(\wedge)$ and $\mathcal{I}(\neg)$ always are the conjunction and negation relations, whereas $\mathcal{I}(\Pi^{\mathfrak{A}})$ is a set that only contains the constant predicate in $\mathcal{D}_{\mathfrak{A}\to o}$ that is true on all members of $\mathcal{D}_{\mathfrak{A}}$. Note that this definition, together with the traditional convention that $\forall X :: \mathfrak{A}.\mathbf{A}$ stands for $\Pi^{\mathfrak{A}}(\lambda X :: \mathfrak{A}.\mathbf{A})$ recovers the usual quantification bounded by the sort $\mathfrak{A}$.

General Assumption 3.2 We furthermore assume that for each $\mathfrak{A} \in \mathcal{S}$, there is a ground formula $\mathbf{G}^{\mathfrak{A}} \in wsf_{\mathfrak{A}}(\Sigma, \Gamma)$. This will guarantee that sorts are not empty. Automated deduction systems based on unification usually work with this implicit assumption, since otherwise the calculus becomes unsound: if the sort $\mathfrak{A}$ is empty the formula $[pX_{\mathfrak{A}}] \wedge \neg[pX_{\mathfrak{A}}]$ is contradictory but not unsatisfiable.

Note, that it is sufficient to assume a ground formula $\mathbf{G}^{\mathfrak{A}}$ for all $\mathfrak{A} \in \mathcal{S}$, if we set $\mathbf{G}^{\mathfrak{A}\to\mathfrak{B}} := [\lambda X :: \mathfrak{A}\mathbf{G}^{\mathfrak{B}}]$. Therefore each well-sorted formula $\mathbf{A} \in wsf_{\mathfrak{A}}(\Sigma, \Gamma)$ has a well-sorted ground instance, that is there exists a Σ-substitution $\sigma \in \mathbf{wsSub}(\Gamma, \Sigma)$, such that $\sigma(\mathbf{A})$ is a ground term. In this case the contradiction above really is unsatisfiable.

Definition 3.3 (Equational System) An *equational system* is a finite set of pairs of well-sorted formulae of the form $\Xi = \{\langle \mathbf{A}^i, \mathbf{B}^i \rangle \mid i = 1, \ldots, n\}$. A substitution $\sigma \in \mathbf{wsSub}(\Gamma, \Sigma)$ is called a Σ-*unifier of a pair* $\langle \mathbf{A}, \mathbf{B} \rangle$, if $\sigma(\mathbf{A}) = \sigma(\mathbf{B})$ and a Σ-*unifier of* Ξ, iff σ is a Σ-unifier of all pairs $\langle \mathbf{A}^i, \mathbf{B}^i \rangle$. We denote the set of Σ-unifiers of Ξ by $\mathbf{wsU}(\Gamma, \Sigma, \Xi)$.

A pair $\langle \mathbf{A}, \mathbf{B} \rangle$ is in Σ-*solved form* in an equational system Ξ, iff $\mathbf{A}$ is the the variable $X_{\mathfrak{A}}$, which does not occur anywhere else in Ξ and $\Gamma \vdash_\Sigma \mathbf{B} :: \mathfrak{A}$, it is called Σ-*pre-solved*, iff $\mathbf{A}$ and $\mathbf{B}$ are flexible, *i.e.* the heads are free variables. Obviously a system $\Xi = \{\langle X^1, \mathbf{A}^1 \rangle, \ldots, \langle X^n, \mathbf{A}^n \rangle\}$ in Σ-solved form, *i.e.* all of its pairs are in Σ-solved form corresponds to a well-sorted substitution $\theta := [\mathbf{S}^1/X^1], \ldots, [\mathbf{A}^n/X^n]$. We write $\langle \theta \rangle := \Xi$ and $\sigma_\Xi := \theta$ and note that σ_Ξ is the most general Σ-unifier for Ξ.

Definition 3.4 (General Binding) Let $[\forall [Y^1 :: \mathfrak{C}_1], \ldots, [Y^n :: \mathfrak{C}_n].\mathbf{B} :: \mathfrak{B}]$ be a term declaration in Σ with $\mathbf{head}(\mathbf{B}) = h$ or let $\mathbf{B} = h$ be a variable of sort $\mathfrak{B}$, then

$$G := (\lambda X^1 \ldots X^l.\mathbf{B}'\mathbf{V}^1 \ldots \mathbf{V}^m)$$

is called a *general binding of sort* $\mathfrak{A}$ *and head* h, if

1. $l = \mathbf{length}(\mathfrak{A})$ and $m = l + \mathbf{length}(\tau(\mathfrak{A})) - \mathbf{length}(\tau(\mathfrak{B}))$
2. $\mathfrak{r}^m(\mathfrak{B}) \leq \mathfrak{r}^l(\mathfrak{A})$
3. $\mathbf{V}^i = (H^i X^1 \ldots X^l)$, where H^i are fresh variables of sort $\mathfrak{d}^1(\mathfrak{A}) \rightarrow \ldots \mathfrak{d}^l(\mathfrak{A}) \rightarrow \mathfrak{d}^i(\mathfrak{B})$.
4. $\mathbf{B}' = \mathbf{B}$ if $\mathbf{B}$ is a variable or $\mathbf{B}' \doteq [\mathbf{W}^i/Y^i]\mathbf{B}$ where $\mathbf{W}^i = (K^i X^1 \ldots X^l)$, where K^i are fresh variables of sort $\mathfrak{d}^1(\mathfrak{A}) \rightarrow \ldots \mathfrak{d}^l(\mathfrak{A}) \rightarrow \mathfrak{C}_i$.

Here $\mathbf{length}(\mathfrak{A})$ stands for the number of arrows in $\mathfrak{A}$ and the term "fresh variables" stands variables not in the current context Γ, but are added to it in the process.

The general bindings get their importance from the fact that if $\mathbf{G}$ is a general binding of sort $\mathfrak{A}$, then $\Gamma \vdash_\Sigma \mathbf{G} :: \mathfrak{A}$ and for any other formula $\mathbf{A}$ of sort $\mathfrak{A}$ there is a substitution ρ, such that $\rho(\mathbf{G}) =_{\beta\eta} \mathbf{A}$. This fact makes it possible to define a sound and complete higher-order unification and pre-unification algorithms $\Sigma\mathcal{UT}$ [15, 13] and $\Sigma\mathcal{PT}$ by substituting the well-sorted general bindings for the well-typed ones in Huet's unification algorithms [23].

Definition 3.5 (Constrained Clause) Let $\mathbf{A}$ be an application, constant or variable of sort $\mathfrak{o}$, such that $\mathbf{head}(\mathbf{A})$ is not one of the logical constants, then $\mathbf{A}$ is called *atomic*. Atoms and their negations are together called *literals*, and finite sets of literals are called *clauses*.

A pair $\mathcal{C} = C||\Xi$, where C is a clause and Ξ is an equational system is called a *constrained clause*, and C is called the *clause* of C and Ξ the *constraint* of $\mathcal{C}$.

Just as in first-order logic we have that each set Φ of sentences can effectively be transformed into a set of clauses $\mathbf{ICCNF}(\Phi)$ that are satisfiable, iff Φ is. The only difference to the simply typed case is that Skolem constants have to be well-sorted, so we have to add term declarations to the current signature. For a discussion of soundness of skolemization see 3.8.

Definition 3.6 (Primitive Substitution) We call a term $\mathbf{P}$ a *primitive instance of sort* $\mathfrak{A} := \mathfrak{A}_1 \rightarrow \ldots \rightarrow \mathfrak{A}_n \rightarrow \mathfrak{o}$, if $\mathbf{P}$ is of the forms

$$(\lambda\overline{X^k}.(H^1\overline{X}) \vee (H^2\overline{X}) \qquad (\lambda\overline{X^k}.\neg(H^1\overline{X})) \qquad (\lambda\overline{X^k}.\forall Y :: \mathfrak{B} H^3 \overline{X} Y)$$

where H^1 and H^2 are variables of sort $\mathfrak{A}$ and H^3 is a variable of sort $\mathfrak{A} := \mathfrak{B} \rightarrow \mathfrak{A}$, or if $\mathbf{P}$ is a j-projection binding of sort $\mathfrak{A}$. Let P be a variable of sort $\mathfrak{A}$, then we will call the substitution $[\mathbf{P}/P]$ a *primitive substitution*.

Definition 3.7 ($\mathcal{HR}$) The $\mathcal{HR}$ calculus is a variant of Huet's constrained resolution and has the following rules of inference. Let $\square$ stand for any constrained clause $\emptyset||\Xi$, where Ξ is an equational system in pre-solved form. A derivation of $\square$ from a set $\mathcal{C}$ of constrained clauses with the inference rules below is called a *refutation of* $\mathcal{C}$.

$$\frac{\{\mathbf{N}^1,\ldots,\mathbf{N}^n\}||\Xi \quad \{\neg\mathbf{M}^1,\ldots,\mathbf{M}^m\}||\Delta}{\{\mathbf{N}^2,\ldots,\mathbf{N}^n,\mathbf{M}^2,\ldots,\mathbf{M}^m\}||\Xi\cup\Delta*\langle\mathbf{N}^1,\mathbf{M}^1\rangle}\ \mathcal{HR}(Res)$$

$$\frac{\{\mathbf{N}^1,\ldots,\mathbf{N}^n\}||\Xi}{\{\mathbf{N}^2,\ldots,\mathbf{N}^n\}||\Xi*\langle\mathbf{N}^1,\mathbf{N}^2\rangle}\ \mathcal{HR}(Fac)$$

$$\frac{\{\neg\mathbf{M}^1,\ldots,\mathbf{M}^m\}||\Xi}{\{\mathbf{M}^1,\ldots,\mathbf{M}^m\}||\Xi\cup\langle P,\mathbf{P}\rangle}\ \mathcal{HR}(Prim)$$

$$\frac{\{\mathbf{M}^1,\ldots,\mathbf{M}^m\}||\Xi,\langle\sigma\rangle}{\{\mathbf{CNF}(\sigma(\mathbf{M}^1)\vee\ldots\vee\sigma(\mathbf{M}^m))\}||\Xi,\langle\sigma\rangle}\ \mathcal{HR}(Solv)$$

$$\frac{\mathbf{C}||\Xi}{\mathbf{C}||\Xi'}\ \mathcal{HR}(\Sigma\mathcal{PU})$$

For $\mathcal{HR}(Prim)$ we have the proviso that one of the $\mathbf{M}^i$ is a flexible literal of the form $P_\alpha\overline{\mathbf{U}^k}$ and $\mathbf{P}$ is a primitive instance of sort $\mathfrak{A}$ and in $\mathcal{HR}(\Sigma\mathcal{PU})$ the unification problem Ξ' is obtained from Ξ by a $\Sigma\mathcal{PT}$ transformation.

The rule $\mathcal{HR}(Solv)$ will propagate partial solutions from the constraints to the clause part and thus help detect clashes early. Since the substitution may well change the propositional structure of the clause by instantiating a predicate variable we have to renormalize the clause on the fly.

Note that in contrast to Huet's calculus we use Andrews' primitive substitutions [3] instead of splitting rules and that we are able to perform unification everywhere in the deduction in contrast to only at the end. The rule $\mathcal{HR}(Solv)$ will propagate partial solutions from the constraints to the clause part and thus help detect clashes early. Since the substitution may well change the propositional structure of the clause by instantiating a predicate variable we have to renormalize the clause on the fly.

We will call a set Φ of well-formed sentences $\mathcal{HR}$-refutable, iff $\square$ is derivable from the set of constrained clauses $\mathbf{CNF}(\Phi)||\emptyset$. Note that $\square$ denotes falsehood or contradiction. so by 3.8 a refutation of a set of sentences Φ proves the unsatisfiability of Φ.

Naturally a practical implementation of $\mathcal{HR}$ would split the set of unification transformations in $\mathcal{HR}(\Sigma\mathcal{PU})$ rule into sets *Simpl* (*trivial*, *decompose*, *eliminate-var*) and *Match* (*imitate*, *project*) and destructively apply the *Simpl* rules whenever one

of the nondeterministic *Match* rules has been applied or the equational system has been instantiated.

Theorem 3.8 (Soundness) *The $\mathcal{HR}$ calculus is sound.*

Proof sketch: It is well known that naive skolemization in higher-order logic is not sound. In fact it is possible to prove an instance of the axiom of choice, which is known to be independent, in a resolution system with naive skolemization. In his thesis [17] Dale Miller gives a sound version of skolemization in the context of expansion trees and higher-order matings. The idea is to restrict the unification algorithm, such that only formulae can be produced by substitution where the Skolem functions always have enough arguments. This method can also be utilized in the resolution context and yields a soundness theorem. □

Remark 3.9 For the relative completeness theorem we consider the Hilbert-style calculus $\mathfrak{T}$ with the usual propositional axioms $(p_o \vee p) \Rightarrow p$, $p_o \Rightarrow (q \Rightarrow p)$, $(p_o \vee q_o) \Rightarrow (q \vee p)$ and $(p_o \Rightarrow q_o) \Rightarrow .(r_o \vee p) \Rightarrow (r \vee q)$, the quantor axioms $\Pi_{o(o\alpha)}F_{o\alpha} \Rightarrow FX_\alpha$ and $\Pi(\lambda X_\alpha(Y_\alpha \vee F_{o\alpha}X)) \Rightarrow .Y \vee \Pi_{o(o\alpha)}F$. As rules of inference we have $\alpha\beta\eta$-conversion, the substitution rule, *i.e.* $\mathbf{A}X_{\mathfrak{A}} \vdash_{\mathfrak{T}} \mathbf{AB}$, for any $\mathbf{B} \in wsf_{\mathfrak{A}}(\Sigma, \Gamma)$, modus ponens and universal generalization. Note that this is a generalization of a subset of the calculus in [7], where the notion of well-formed formulae and substitution is simply changed to well-sorted.

Theorem 3.10 (Completeness) *$\mathcal{HR}$ is complete relative to $\mathfrak{T}$, in other words, $\mathcal{HR}$ can refute the negation of all $\mathfrak{T}$-theorems.*

Proof sketch: To prove completeness of $\mathcal{HR}$ we proceed along the lines of the proofs for completeness of Huet's unsorted resolution calculus in [9] and redo all arguments in the order-sorted setting. In particular we have to generalize the "Unifying Principle" for type theory from [1] to $\Sigma\mathcal{T}$. For this enterprise we suitably modify the definition of abstract consistency property by requiring well-sorted substitutions for the Π quantor and show that all sets Φ of sentences that have an abstract consistency property can be extended to higher-order order-sorted Hintikka sets, which are closely related to Andrews' semivaluation v. For these we can build a term model along the lines of the construction of v-complexes. In order to prove the unifying principle which states that is Φ is $\mathfrak{T}$-consistent it is now only necessary to verify that $\mathfrak{T}$ is sound with respect to v-complexes.

The completeness proof for $\mathcal{HR}$ concludes by verifying that the property of sentences not to be $\mathcal{HR}$-refutable is an abstract consistency property and therefore $\mathfrak{T}$-consistent. Since the negations of $\mathfrak{T}$-theorems are $\mathfrak{T}$-inconsistent, they must therefore

be $\mathcal{HR}$-refutable. We stress that the completeness of the pre-unification algorithm $\Sigma\mathcal{PT}$ is a key fact in the verification of the conditions for an abstract consistency property. □

4 Relativization

In first-order logic, passing to a sorted signature only enhances the expressive power of the system with respect to practical considerations, such as the conciseness of representation, the size of search spaces and the like. It does not enhance the expressive power in principle, as the sorted language and model theory can always be coded into an unsorted version by the technique of relativization (see the sort theorems in [25, 22]). However the system loses much of its deductive power in the sense of larger search spaces. Relativization in $\Sigma\mathcal{T}$ is even more awkward than in the first-order case, as order-sorted type theory captures a nontrivial fragment of a type theory with partial functions, which would have to be coded into relativization.

It is noteworthy that, even though we do not require description functions in $\Sigma\mathcal{T}$, we need them in our target system for coding partial functions. So in this section we consider a formulation of simple type theory with description functions as in [7]. This system is obtained by adding a logical constant $\iota_{\alpha(o\alpha)}$ for each $\alpha \in \mathcal{T}$ and the axiom

$$P_{o\alpha}X_\alpha \Rightarrow .\forall Y_\alpha(PY \Rightarrow X = Y) \Rightarrow P(\iota P)$$

to $\mathfrak{T}$. Furthermore in the definition of general models we have to specify that the value of ι is that function that maps singleton sets to their unique member. It is well known that in this system we can define conditionals like $\mathbf{w}_{\alpha\alpha o}$, such that $(\mathbf{w}\mathbf{P}_o\mathbf{A}_\alpha)$ is $\mathbf{A}$, iff $\mathbf{P}$ is true.

Definition 4.1 (Relativization) Let Σ be a higher-order order-sorted signature. We first extend $\overline{\Sigma}$ by a predicate symbol $\mathbf{P}_{\mathfrak{A}}$ of type $\tau(\mathfrak{A}) \to o$ for each base sort symbol $\mathfrak{A}$ and then define the *relativization operator* **Rel** on sort symbols by

- $\mathbf{Rel}(\mathfrak{A}) := \mathbf{P}_{\mathfrak{A}}$ if $\mathfrak{A}$ is a nonfunctional base sort symbol,
- $\mathbf{Rel}(\mathfrak{A}) := (\lambda X_{\tau(\mathfrak{A})}.(\mathbf{P}_{\mathfrak{A}}X) \wedge (\mathbf{Rel}(\mathfrak{d}(\mathfrak{A}) \to \mathfrak{r}(\mathfrak{A}))X))$, if $\mathfrak{A}$ is a functional base sort symbol, and
- $\mathbf{Rel}(\mathfrak{A} \to \mathfrak{B}) := (\lambda F_\alpha.\forall X_{\tau(\mathfrak{A})}.(\mathbf{Rel}(\mathfrak{A})X) \Rightarrow (\mathbf{Rel}(\mathfrak{B})(FX)))$ and $\alpha = \tau(\mathfrak{A}) \to \tau(\mathfrak{B})$.

Rel acts as the identity on well-sorted terms except that it changes sorts to their types on variables and abstractions of the form $(\lambda X_{\mathfrak{B}}.\mathbf{A})$ are relativized to the formula

$(\lambda X_{\tau(\mathfrak{B})}.\mathbf{w}(\mathbf{Rel}(\mathfrak{B})X)\mathbf{A})$. For term declarations we have

$$\mathbf{Rel}((\mathbf{A} :: \mathfrak{A})) := (\mathbf{Rel}(\mathfrak{A}_1)X^1_{\tau(\mathfrak{A}_1)}) \Rightarrow \ldots (\mathbf{Rel}(\mathfrak{A}_k)X^k_{\tau(\mathfrak{A}_k)}) \Rightarrow (\mathbf{Rel}(\mathfrak{A})\mathbf{Rel}(\mathbf{A}))\ ,$$

where $\{X^1_{\mathfrak{A}_1}, \ldots, X^k_{\mathfrak{A}_k}\}$ is the set of free variables of $\mathbf{A}$. Now $\mathbf{Rel}(\Sigma)$ is just the universal closure of the conjunction of all term declarations in Σ.

Example 4.2 Let $[\mathfrak{A} \leq \mathfrak{B}]$ be a subsort declaration in Σ, then

$$\mathbf{Rel}([\mathfrak{A} \leq \mathfrak{B}]) = [\forall[X :: \mathfrak{A}].X :: \mathfrak{B}] = \forall X_{\tau(\mathfrak{A})}.(\mathbf{Rel}(\mathfrak{A})X) \Rightarrow .\mathbf{Rel}(\mathfrak{B})X)\ ,$$

just as in first-order logic, if we keep in mind that $\mathbf{w}_{ooo}$ is just implication.

The following theorem shows that the **Rel** operator is a total and faithful encoding of order-sorted type theory into simple type theory.

Theorem 4.3 (Sort Theorem) *Let Φ be a set of sentences, then Φ has a Σ-model, iff* $\mathbf{Rel}(\Phi)$ *has a $\overline{\Sigma} \cup \mathbf{P}_{\mathcal{S}}$-model that entails* $\mathbf{Rel}(\Sigma)$.

Example 4.4 We will take another look at example 2.7 to compare deduction in $\Sigma\mathcal{T}$ to that in simple type theory from a practical point of view. If Σ is the sorted signature defined in 2.7, then $\mathbf{Rel}(\Sigma)$ is the following set of formulae. So in the light of the previous theorem this is the taxonomic information present in the sorted signature Σ.

$$\forall F.((\mathbf{P}_{\mathfrak{D}}F) \wedge \forall Y.(\mathbf{P}_{\mathfrak{R}}Y) \Rightarrow (\mathbf{P}_{\mathfrak{R}}(FY))) \Rightarrow .(\mathbf{P}_{\mathfrak{C}}(\partial F)) \wedge (\forall Y.(\mathbf{P}_{\mathfrak{R}}Y) \Rightarrow (\mathbf{P}_{\mathfrak{R}}(\partial F)Y))$$
$$\forall F.((\mathbf{P}_{\mathfrak{P}}F) \wedge \forall Y.(\mathbf{P}_{\mathfrak{R}}Y) \Rightarrow (\mathbf{P}_{\mathfrak{R}}(FY))) \Rightarrow .(\mathbf{P}_{\mathfrak{P}}(\partial F)) \wedge (\forall Y.(\mathbf{P}_{\mathfrak{R}}Y) \Rightarrow \mathbf{P}_{\mathfrak{R}}(\partial F)Y)$$
$$\mathbf{P}_{\mathfrak{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathfrak{R}}XX))\forall Y.(\mathbf{P}_{\mathfrak{R}}Y) \Rightarrow (\mathbf{P}_{\mathfrak{R}}(\mathbf{w}(\mathbf{P}_{\mathfrak{R}}X)X))$$
$$\forall Z.(\mathbf{P}_{\mathfrak{R}}Z \Rightarrow \mathbf{P}_{\mathfrak{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathfrak{R}}XZ))\forall Y.(\mathbf{P}_{\mathfrak{R}}Y)) \Rightarrow (\mathbf{P}_{\mathfrak{R}}(\mathbf{w}(\mathbf{P}_{\mathfrak{R}}X)Z)))$$
$$\forall F, G.(\mathbf{P}_{\mathfrak{P}}F) \Rightarrow .(\mathbf{P}_{\mathfrak{P}}G) \Rightarrow$$
$$\quad \mathbf{P}_{\mathfrak{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathfrak{R}}X. + (FX)(GX)))\forall Y.(\mathbf{P}_{\mathfrak{R}}Y) \Rightarrow (\mathbf{P}_{\mathfrak{R}}(\mathbf{w}(\mathbf{P}_{\mathfrak{R}}Y) + (FY)(GY)))$$
$$\forall F, G.(\mathbf{P}_{\mathfrak{P}}F) \Rightarrow .(\mathbf{P}_{\mathfrak{P}}G) \Rightarrow$$
$$\quad \mathbf{P}_{\mathfrak{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathfrak{R}}X. * (FX)(GX)))\forall Y.(\mathbf{P}_{\mathfrak{R}}Y) \Rightarrow (\mathbf{P}_{\mathfrak{R}}(\mathbf{w}(\mathbf{P}_{\mathfrak{R}}Y) * (FY)(GY)))$$

This set of axioms would have to be added to the relativization of any well-sorted theorem that we want to prove in the relativized form. A further effect that we have not illustrated for lack of space is that the unification in the order-sorted setting will find out conflicting taxonomic information for proof objects and prevent any inference that would yield ill-sorted objects. These objects arise naturally in the relativized context but due to $\mathbf{Rel}(\Sigma)$ can never contribute to any proof. Thus resolution in $\Sigma\mathcal{T}$ gives us the further advantage of cutting off redundant branches in the search space.

5 Conclusion and Further Work

We have presented a generalization of Huet's Constrained Resolution calculus for order-sorted type theory and a systematic procedure in which sorted type theory can be encoded into the simple type theory of [2]. This shows that in theory all theorems of $\Sigma\mathcal{T}$ can be proven by coding them into simple type theory and then proving them by simply typed constrained resolution. However as example 4.4 shows the search spaces involved are so much bigger that order-sorted resolution is of clearly practical advantage. In an a posteriori view we can view order-sorted resolution as the process of building certain classes of axioms, namely those that correspond to term declarations, into the unification. This takes axioms of the form $(\forall \overline{X^k}.p^1X^1 \Rightarrow \ldots \Rightarrow p^kX^k \Rightarrow (q\mathbf{A}))$, where the p^i and q are predicate constants and X^i are the free variables of $\mathbf{A}$, out of the search and treats them algorithmically in the unification process.

On the other hand it is clear that the basic operations like unification are more complicated than in the unsorted case, and it is conceivable that with the use of term declarations it is possible to encode so much information into the signature that sorted deduction systems may in some cases be less efficient than unsorted ones.

In first-order predicate logic the introduction of term declarations has been a major step to the development of dynamic order-sorted logics [27], where variables are restricted to sorts, but where the sorts can also be treated as unary predicates in the logic, allowing the specification of conditioned term declarations; thus the signature is no longer fixed across the deduction, as sort information can appear in the deduction process. The resolution rule always uses sorted unification with respect to the signature specified by the current state of the proof. Since predicates are primary objects of type theory, a generalization of that in [27] may yield very powerful calculi for mechanizing mathematics and in particular analysis. Recently Weidenbach's results have been applied by the author in collaboration with Kerber to obtain an efficient resolution calculus for first-order logic with partial functions. We believe that this result can be generalized to higher-order logic and lead to a very natural and powerful logic system for mechanizing informal mathematical practice.

References

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

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

[3] Peter B. Andrews. On connections and higher order logic. *Journal of Automated Reasoning*, 5:257–291, 1989.

[4] Peter B. Andrews, Eve Longini-Cohen, Dale Miller, and Frank Pfenning. Automating higher order logics. *Contemp. Math*, 29:169–192, 1984.

[5] Kim B. Bruce and Giuseppe Longo. A modest model of records, inheritance and bounded quantification. *Information and Computation*, 87:196–240, 1990.

[6] Luca Cardelli. A semantics of multiple inheritance. In *[11]*, 1984.

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

[8] A. G. Cohn. Taxonomic reasoning with many-sorted logics. *Artificial Intelligence Review*, 3:89–128, 1989.

[9] Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic.* PhD thesis, Case Western Reserve University, 1972.

[10] Patricia Johann and Michael Kohlhase. Unification in an extensional lambda calculus with ordered function sorts and constant overloading. SEKI-Report SR-93-14, Universität des Saarlandes, 1993. submitted to CADE'94.

[11] G. Kahn and G. Plotkin D.G. MacQueen, editors. *Semantics of Data Types*, number 173 in LNCS. Springer Verlag, 1984.

[12] D. Kapur, editor. *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, Saratoga Spings,NY, USA, 1992. Springer Verlag.

[13] Michael Kohlhase. Beweissysteme mit Logiken höherer Stufe. In Karl Hans Bläsius and Hans-Jürgen Bürckert, editors, *Deduktionssysteme, Automatisierung des Logischen Denkens*, chapter 6, pages 213–238. R. Oldenbourg Verlag, 2 edition, 1992.

[14] Michael Kohlhase. Unification in order-sorted type theory. In *[24]*, pages 421–432, 1992.

[15] Michael Kohlhase. *Automated Deduction in Order-Sorted Type Theory.* PhD thesis, Universität des Saarlandes, 1994. to appear.

[16] Michael Kohlhase and Frank Pfenning. Unification in a λ-calculus with intersection types. In Dale Miller, editor, *Proceedings of the International Logic Programming Sympsion. ILPS'93*, pages 488–505. MIT Press, 1993.

[17] Dale Miller. *Proofs in Higher-Order Logic.* PhD thesis, Carnegie-Mellon University, 1983.

[18] Tobias Nipkow and Zhenyu Qian. Reduction and unification in lambda calculi with subtypes. In *[12]*, pages 66–78, 1992.

[19] Arnold Overschelp. Untersuchungen zur mehrsortigen Quantorenlogik. *Mathematische Annalen*, 145:297–333, 1962.

[20] Benjamin C. Pierce. *Programming with Intersection Types and Bounded Polymorphism.* PhD thesis, School of Computer Science, Carnegie Mellon University, December 1991. Available as Technical Report CMU–CS–91–205.

[21] Zhenyu Qian. *Extensions of Order-Sorted Algebraic Specifications: Parameterization, Higher-Functions and Polymorphism.* PhD thesis, Universität Bremen, März 1991.

[22] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *LNAI.* Springer Verlag, 1989.

[23] Wayne Snyder and Jean Gallier. Higher-Order Unification Revisited: Complete Sets of Transformations. *J. Symbolic Computation*, 8:101–140, 1989.

[24] A. Voronkov, editor. *Proceedings of the International Conference on Logic Programming and Automated Reasoning*, volume 624 of *LNAI.* Springer Verlag, 1992.

[25] Christoph Walther. A many–sorted calculus based on resolution and paramodulation. In Alan Bundy, editor, *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 882–891, Los Altos, California, USA, August 1983. William Kaufmann.

[26] Christoph Walther. Many-sorted unification. *Journal of the Accociation for Computing Machinery*, 35(1):1–17, January 1988.

[27] Christoph Weidenbach. A sorted logic using dynamic sorts. Technical Report MPI-I-91-218, Max-Planck-Institut für Informatik, 1991.