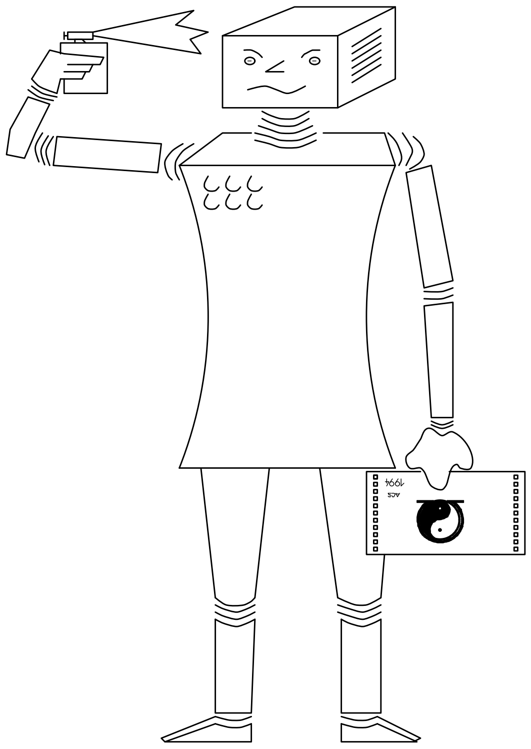

Unification in a Sorted λ-Calculus with Term Declarations and Function Sorts

Michael Kohlhase

Published as: In Proceedings of the 18.th Annual German Conference on Artificial Intelligence, September 18.-23. 1994, Saarbrücken, Germany, *LNAI* Springer Verlag

Unification in a Sorted λ-Calculus with Term Declarations and Function Sorts

Michael Kohlhase[*]

FB Informatik, Universität des Saarlandes, 66041 Saarbrücken, Germany
+49-681-301-4627 kohlhase@cs.uni-sb.de
http://js-sfbsun.cs.uni-sb.de/pub/www

Abstract. The introduction of sorts to first-order automated deduction has brought greater conciseness of representation and a considerable gain in efficiency by reducing search spaces. This suggests that sort information can be employed in higher-order theorem proving with similar results. This paper develops a sorted λ-calculus suitable for automatic theorem proving applications. It extends the simply typed λ-calculus by a higher-order sort concept that includes term declarations and functional base sorts. The term declaration mechanism studied here is powerful enough to subsume subsorting as a derived notion and therefore gives a justification for the special form of subsort inference. We present a set of transformations for sorted (pre-) unification and prove the nondeterministic completeness of the algorithm induced by these transformations.

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 restricting 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 to a more concise representation of problems and proofs than the unsorted variants. The exploitation of this information during proof search can dramatically reduce the search space associated with theorem-proving and make the resulting sorted calculi much more efficient. In the context of first-order logic sort information has been successfully employed by C. Walther [21], M. Schmidt-Schauß [19], A. Cohn [6], C. Weidenbach [22] 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 [2] are rather weak on the first-order fragment, which is in part due to the fact that many of the advances of first-order deduction (like sorted calculi) have not yet been transported to higher-order logic. Thus the question about the behavior of higher-order logic under the constraints of a full 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

[*] This work was supported by the Deutsche Forschungsgemeinschaft in SFB 314 (D2)

study of a sorted version of higher-order logic in an appendix to [7]. The unification problem in extensions of this system have since been studied by Nipkow and Qian [16] and by Pfenning and the author [14]. 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 [18, 4, 3, 17].

Here we present a λ-calculus $\Sigma\mathcal{T}$ that differs from the abovementioned in that we do not consider function restriction as a "built-in" of the system, since we take the mathematical intuition that functions have uniquely specified domains seriously. Consequently our subsort relation is not covariant in the domain sort (this principle semantically corresponds to implicit function restriction). Furthermore the term declaration mechanism is much more powerful than the declaration schemas proposed in those logical systems. This paper is an extension of the results presented in [8, 9]. This subsystem of $\Sigma\mathcal{T}$ only allows signatures consisting of constant declarations and thus treats the interaction of functional base sorts and extensionality in isolation. In contrast to this subsystem the powerful mechanism of term declarations in $\Sigma\mathcal{T}$ allows a straightforward specification of many mathematical concepts (cf. Example 2.8). This paper also corrects an earlier attempt [11] to solve the problem. We have corrected the relevant definitions of [11] and with these were able to prove all the results claimed there. For details and proofs we refer the reader to [12].

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 class of objects is through the use of unary predicates, such as the predicate $\mathbb{N}_{\iota\to o}$ in the formulae $(\mathbb{N}2_\iota)$, *i.e.* "2 is a natural number", or $\neg(\mathbb{N}\mathrm{Peter}_\iota)$, *i.e.* "Peter is not a natural number". This leads to a multitude of unit clauses $(\mathbb{S}_{\iota\to o}\mathbf{A})$ 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 $\forall X_\iota.(\mathbb{N}X_\iota) \Rightarrow (\geq_{\iota\to\iota\to o} X_\iota 0_\iota)$. This approach is unsatisfactory because inter alia the derivation of the nonsensical formula $(\mathbb{N}\mathrm{Peter}) \Rightarrow (\geq \mathrm{Peter}\ 0)$ is permitted, even though $(\geq \mathrm{Peter}\ 0)$ can never be derived because of $\neg(\mathbb{N}\mathrm{Peter})$. Sorted logics remedy this situation by assigning sorts to constants and variables and by restricting quantification to sorts. Furthermore formulae have to meet certain (sort) restrictions to denote meaningful objects.

In typed λ-calculi the idea of declaring sort information is very natural, as all objects are already typed, which amounts to a – very coarse – division of the universe into classes. The type system is merely refined by considering the sorts as additional base types. For example the last formula above would read $\forall X_{\mathbb{N}}.(\geq X0)$, where $\geq$ is a binary relation on $\mathbb{N}$ and 0 is of sort $\mathbb{N}$ in the signature. Sorting the universe of individuals gives rise to new classes of functions, namely functions, where domains and codomains 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 that we represent by base sorts of functional type. As an example for the sort restrictions on formulae consider the application $(\mathbf{AB})$. Here, there must be sorts $\mathbb{A}$ and $\mathbb{B}$, such that $\mathbf{A}$ is of sort $\mathbb{A}$, $\mathbf{B}$ is of sort $\mathbb{B}$ and $\mathbb{B}$ is a subsort of the domain sort of $\mathbb{A}$. The sort of the application $(\mathbf{AB})$ is defined to be the codomain sort of $\mathbb{A}$.

In $\Sigma\mathcal{T}$ we relax the implicit condition that only the sorts of constants and variables can be declared, and allow declarations of the form $[\forall\Gamma\mathbf{.A}::\mathbb{A}]$ called term declarations, where $\mathbf{A}$ can be an arbitrary formula of appropriate type and Γ is a variable context. 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 for instance the addition function, which we (semantically) would like to have the sort $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ where $\mathbb{N}$ is the sort of natural numbers. If we also have a sort for the even numbers $\mathbb{E}$, then we might want to specify that the expression $[+aa]$ is an even number, even if a is not. This information can be formalized by declaring the term $[+X_{\mathbb{N}}X_{\mathbb{N}}]$ to be of sort $\mathbb{E}$ using a term declaration. We might also want to give the addition function the sort $\mathbb{E}\times\mathbb{E} \to \mathbb{E}$, however since we insist that terms have unique domain sorts, we cannot declare this directly in the signature. Closer inspection of the semantics behind our example reveals that it is consistent with our program to declare the restriction of the addition function to the even numbers has codomain in the evens, which we can legally do with a term declaration $[\forall[X::\mathbb{E}],[Y::\mathbb{E}]\mathbf{.}+XY::\mathbb{E}]$.

In this expressive system term declarations of the form $[\forall[X::\mathbb{A}]\mathbf{.}X::\mathbb{B}]$ entail that $\mathbb{A}$ is a subsort of $\mathbb{B}$ and induce the intended subsort ordering on the set of sorts.

2 Sorted λ-Calculus

We assume the reader to be familiar with the syntax and semantics of simply typed λ-calculus $(\Lambda^{\to})$ [5, 1]. The set $\mathcal{T}$ of *types* $(\alpha, \beta, \gamma \ldots)$ is built up from a set $\mathcal{BT}$ of *base types* by closure under $\to$. We assume a set of typed constants $\Sigma := \bigcup_{\alpha\in\mathcal{T}} \Sigma_\alpha$ and a countably infinite set $\mathcal{V}_\alpha$ of variables for each type $\alpha \in \mathcal{T}$. Well-formed formulae are built up from variables and constants as *applications* and λ-*abstractions* in the usual way.

Another mathematical notion which will play a great role in this paper is that of a *partial function* $\Phi\colon A \longrightarrow B$. With this we will mean a relation $\Phi \subseteq A \times B$, such that for all pairs (a, b) and (c, d) in Φ we have $a \neq c$. For partial functions that can be presented by a finite set of pairs (e.g. substitutions or variable contexts), we will often use a notation like $\Phi := [b^1/a^1], \ldots, [b^n/a^n]$ if $\Phi = \{(a^1, b^1), \ldots, (a^n, b^n)\}$. Furthermore, if Ψ is that partial function, such that $\Psi(a) = b$ but $\Psi(c) = \Phi(c)$ for all $c \neq a$, then we will denote Ψ by $\Phi, [b/a]$. If the restrictions of Φ and Ψ to $\mathbf{Dom}(\Phi) \cap \mathbf{Dom}(\Psi)$ are identical, then we say that they *agree* $(\Phi||\Psi)$. In this case $\Phi \cup \Psi$ is again a partial function.

Definition 2.1 (Sort System) A *sort system* is a quintuple $(\mathcal{S}, \mathcal{BS}, \mathfrak{r}, \mathfrak{d}, \tau)$, where $\mathcal{BS}$ is a finite set of symbols, called *base sorts* and the *set of sorts* $\mathcal{S}$ is the closure of $\mathcal{BS}$ under $\to$. We will denote sorts with symbols like $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, $\mathbb{D}$ and have $\mathbb{B} \to \mathbb{A} \in \mathcal{S}$, whenever $\mathbb{A}, \mathbb{B} \in \mathcal{S}$. The functions $\mathfrak{r}, \mathfrak{d}$ and τ specify the sorts of the *codomain*, *domain* and the *type* of a sort and we require that $\tau(\mathbb{A}) = \tau(\mathfrak{d}(\mathbb{A})) \to \tau(\mathfrak{r}(\mathbb{A}))$ for all sorts $\mathbb{A} \in \mathcal{S}$. The type $\tau(\mathbb{A}) \in \mathcal{T}$ is called the *type of the sort* $\mathbb{A}$. We will denote the set of sorts of type α with

$\mathcal{S}_\alpha$, call a sort $\mathbb{A} \in \mathcal{S}_{\alpha\to\beta}$ a *functional sort* and denote the set of functional sorts by $\mathcal{S}^f$ (non-functional sorts by $\mathcal{S}^{nf}$). Note that the sets $\mathcal{BS}$ and $\mathcal{S}^{nf}$ are in general distinct (see example 2.8). Furthermore let $\mathbf{ln}(\mathbb{A}) := 0$, iff $\mathbb{A} \in \mathcal{BS}$ and $\mathbf{ln}(\mathbb{A} \to \mathbb{B}) := 1 + \mathbf{ln}(\mathbb{B})$. We will use the shorthands $\mathfrak{r}^i(\mathbb{A})$ and $\mathfrak{r}^i(\mathbb{A})$ defined by

$$\mathfrak{r}^0(\mathbb{A}) = \mathbb{A} \quad \mathfrak{r}^{i+1}(\mathbb{A}) = \mathfrak{r}(\mathfrak{r}^i(\mathbb{A})) \qquad \mathfrak{d}^0(\mathbb{A}) = \mathbb{A} \quad \mathfrak{d}^{i+1}(\mathbb{A}) = \mathfrak{d}(\mathfrak{r}^i(\mathbb{A}))$$

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 $\mathbb{A}$ and $\mathbb{B}$ of $\mathbf{A}$, the identity $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B})$ holds. The proof that signatures indeed satisfy this property depends on the consistency conditions for valid signatures, given in terms of the equivalence relation **Rdom** , where $\mathbb{A}$ **Rdom** $\mathbb{B}$, iff $\mathfrak{d}^i(\mathbb{A}) = \mathfrak{d}^i(\mathbb{B})$ for all $i \leq k$, such that $\mathfrak{r}^k(\mathbb{A})$ and $\mathfrak{r}^k(\mathbb{B})$ are of the same base type.

Next, we will introduce the concept of well-sortedness for formulae. A term $\mathbf{A}$ will be called *well-sorted* with respect to a signature Σ and a context Γ, if the judgment $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ 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 given by term schemata (the 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 the judgment $\vdash_{sig} \Sigma$ (Σ is a a valid signature) and that for well-sortedness ($\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$) into one large system $\Sigma\mathcal{T}$. Another difficulty is that we also have to treat a sorted $\beta\eta$-conversion judgment $\Gamma \vdash_\Sigma \mathbf{A}{=_{\beta\eta}}\mathbf{B}$ in $\Sigma\mathcal{T}$, since we want $\beta\eta$-conversion to be sort preserving.

Definition 2.2 (Variable Context) Let X_α be a variable and $\mathbb{A}$ a sort, then we call a pair $[X::\mathbb{A}]$ a *variable declaration* for X, iff $\tau(\mathbb{A}) = \alpha$. We call a finite set of variable declarations a *(variable) context* ($\vdash_{ctx} \Gamma$), if it is a partial function *i.e.* $\Gamma \subseteq \mathcal{V}_\alpha \times \mathcal{S}_\alpha$. Note that with our convention for partial functions we have $\Gamma(X) = \mathbb{A}$ for $\Gamma := \Gamma', [X::\mathbb{A}]$, even if $\Gamma'(X) = \mathbb{B}$.

Definition 2.3 (Well-Sorted Formulae and Valid Signatures) For a fixed signature Σ and a context Γ we say that a formula $\mathbf{A}$ *is of sort* $\mathbb{A}$, iff the judgment $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ is derivable in the following inference system.

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

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

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B} \quad \Gamma \vdash_\Sigma \mathbf{A}{=_{\beta\eta}}\mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}}$$

The following inference rules define the the judgment $\vdash_{sig} \Sigma$ by specifying that it is legal to add term declarations to valid signatures, if either they are the first declarations for new constants, or if the formula **A** is well-sorted and the new sort $\mathbb{A}$ respects function domains.

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

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

Finally let $\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ be the congruence judgment induced by the reduction judgment.

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}{\Gamma \vdash_\Sigma (\lambda X_{\mathfrak{d}(\mathbb{A})}\mathbf{.A}X) \rightarrow_\eta \mathbf{A}} \qquad \frac{\Gamma, [X{::}\mathbb{B}] \vdash_\Sigma \mathbf{A}{::}\mathbb{A} \quad \Delta \vdash_\Sigma \mathbf{B}{::}\mathbb{B} \quad \Gamma||\Delta}{\Gamma \cup \Delta \vdash_\Sigma (\lambda X_{\mathbb{B}}\mathbf{.A})\mathbf{B} \rightarrow_\beta [\mathbf{B}/X]\mathbf{A}}$$

In the definition of sorted η-reduction we have taken care to identify the *supporting sort* $\mathfrak{d}(\mathbb{A})$ of **A** (which will turn out to be unique in theorem 2.4), since the formula $(\lambda X_{\mathbb{B}}\mathbf{.A}X)$ denotes the restriction of the function **A** to sort $\mathbb{B}$, if $\mathbb{B}$ is a subsort of $\mathfrak{d}(\mathbb{A})$ and can therefore not be equal to **A**.

It is easy to see that the judgments defined above respect well-typedness, *i.e.* that the information described by $\Sigma\mathcal{T}$ merely refines the type information. In particular sorted $\beta\eta$-conversion is a sub-relation of typed conversion. As a direct consequence sorted $\beta\eta$-reduction is terminating. The confluence result depends on the following theorem

Theorem 2.4 *If* $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ *and* $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{B}$, *then* $\mathbb{A}\,\mathbf{Rdom}\,\mathbb{B}$.

In fact the formal system $\Sigma\mathcal{T}$ is designed to capture informal mathematical practice, where functions have unique domains associated with them.

If we only have one base sort per base type, then the set of well-sorted formulae is isomorphic to the set of well-typed formulae, therefore $\Sigma\mathcal{T}$ is a generalization of $\Lambda^{\rightarrow}$. 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.

Definition 2.5 Let Γ and Δ be variable contexts, then we call a substitution σ a Σ-*substitution* ($\sigma \in \mathbf{wsSub}(\Sigma, \Delta \rightarrow \Gamma)$), iff the judgment $\Gamma \vdash_\Sigma \sigma{::}\Delta$ is derivable in the following inference system.

$$\frac{}{\emptyset \vdash_\Sigma \emptyset{::}\emptyset} \qquad \frac{\Gamma \vdash_\Sigma \sigma{::}\Delta \quad \Gamma' \vdash_\Sigma \mathbf{A}{::}\mathbb{A} \quad \Gamma||\Gamma' \quad X \notin \mathbf{Dom}(\Gamma)}{\Gamma \vdash_\Sigma \sigma, [\mathbf{A}/X]{::}\Delta, [X{::}\mathbb{A}]}$$

Let $\Gamma \vdash_\Sigma \sigma{::}\Delta$. We can show that if $\Xi \cup \Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, then $\Xi \cup \Gamma \vdash_\Sigma \sigma(\mathbf{A}){::}\mathbb{A}$ and furthermore $\mathbf{Dom}(\Delta) = \mathbf{Dom}(\sigma)$ and $\mathbf{Dom}(\Delta) \cap \mathbf{Dom}(\Gamma) = \emptyset$. Thus Σ-substitutions are idempotent and their application conserves sets of sorts. As a consequence we can show that if $\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$, then **A** and **B** have the same set of sorts. Thus the fundamental operations of sorted higher-order deduction systems do not allow the formation of ill-sorted terms from well-sorted ones. This will ensure that such systems never have to handle ill-sorted terms, even intermediately.

Let $\Gamma \vdash_\Sigma X{::}\mathbb{B}$ but $\Gamma(X) = \mathbb{A}$ (we abbreviate this by $\Gamma \vdash_\Sigma \mathbb{A} \leq_\Sigma \mathbb{B}$), then for all formulae $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ we also have $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{B}$, since $\Gamma \vdash_\Sigma [\mathbf{A}/X]X{::}\mathbb{B}$. This is just the situation that is captured with the notion of sort inclusion in traditional sorted logics, where the subsort relation is the smallest partial ordering that contains a set of subsort declarations. The subsort relation plays such a central role in these systems that they are collectively called "order-sorted". Since subsorting is a derived relation in $\Sigma\mathcal{T}$ (cf. theorem 2.7), we do not have to treat it in our meta-logical development. On the object level (and for computation) however it is a useful notion to employ, since it allows to specify taxonomic hierarchies of sorts, which play a great role in intuitive mathematics.

In contrast to the first-order systems the subsort relation is not finite, even with a finite set of base sorts. Thus the relation cannot be pre-computed in advance. On the other hand it is not clear, whether the sort-checking problem is decidable (in fact this problem can be seen to be equivalent to the higher-order matching problem, where decidability is known only for restricted classes of formulae), which is another reason for limited practical usefulness of the full subsorting relation. One way out of this situation is to approximate the subsort relation by a sub-relation computed from a finite set of subsort declarations with certain induction principles.

Definition 2.6 (Sort Inclusion) Let $\mathcal{R}$ be a binary relation on sorts, such that $[X{::}\mathbb{A}] \vdash_\Sigma X{::}\mathbb{B}$, whenever $\mathcal{R}(\mathbb{A},\mathbb{B})$, then we call $\mathcal{R}$ an *approximation of the subsort relation in* Σ. We will call term declarations of the form $[\forall X_\mathbb{A}.X{::}\mathbb{B}]$ *subsort declarations* and abbreviate them with $[\mathbb{A} \leq \mathbb{B}]$. The following inference system is called the $\Sigma\mathcal{T}$ *subsort inference system for* $\mathcal{R}$

$$\frac{\mathcal{R}(\mathbb{A},\mathbb{B}) \quad \vdash_{sig} \Sigma}{\Sigma \vdash \mathbb{A} \leq_\mathcal{R} \mathbb{B}} \qquad \frac{\vdash_{sig} \Sigma}{\Sigma \vdash \mathbb{A} \leq_\mathcal{R} \mathbb{A}} \qquad \frac{\vdash_{sig} \Sigma}{\Sigma \vdash \mathbb{A} \leq_\mathcal{R} \mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})}$$

$$\frac{\Sigma \vdash \mathbb{A} \leq_\mathcal{R} \mathbb{B} \quad \Sigma \vdash \mathbb{B} \leq_\mathcal{R} \mathbb{C}}{\Sigma \vdash \mathbb{A} \leq_\mathcal{R} \mathbb{C}} \qquad \frac{\Sigma \vdash \mathbb{A} \leq_\mathcal{R} \mathbb{B}}{\Sigma \vdash \mathbb{C} \to \mathbb{A} \leq_\mathcal{R} \mathbb{C} \to \mathbb{B}}$$

We will call the relation $\mathcal{R}^\leq$ defined by $\mathcal{R}^\leq(\mathbb{A},\mathbb{B})$, iff $\Sigma \vdash \mathbb{A} \leq_\mathcal{R} \mathbb{B}$ is the *ordering relation for* $\mathcal{R}$. For a given, valid signature Σ we will denote subsorting judgment for the subsort declarations simply with $\Sigma \vdash \mathbb{A} \leq \mathbb{B}$.

Theorem 2.7 *If $\mathcal{R}$ is an approximation of the subsort relation of Σ, then the relation $\mathcal{R}^\leq$ is also an approximation.*

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

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A} \quad \Sigma \vdash \mathbb{A} \leq_\mathcal{R} \mathbb{B}}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{B}}$$

As a consequence of Theorem 2.7 we can see that if $\mathcal{R}$ is an approximation of the subsort relation in Σ, then the weakening rule is admissible in $\Sigma\mathcal{T}$. Furthermore we have $\mathbb{A}$ **Rdom** $\mathbb{B}$ whenever $\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}$. In particular $\tau(\mathbb{A}) = \tau(\mathbb{B})$ in this situation and therefore, the sets $\{\mathbb{A} \in \mathcal{S} \mid \tau(\mathbb{A}) = \alpha\}$ are mutually incomparable.

Example 2.8 Let $\mathcal{BS} := \{\mathbb{R}, \mathbb{C}, \mathbb{D}, \mathbb{P}\}$ where the intended meaning of $\mathbb{R}$ is the set of real numbers, that of $\mathbb{C}$ and $\mathbb{D}$ the sets of continuous and differentiable functions and finally that of $\mathbb{P}$ the set of polynomials. Therefore the types have to be $\tau(\mathbb{R}) = \iota$, $\tau(\mathbb{C}) = \tau(\mathbb{D}) = \tau(\mathbb{P}) = \iota \to \iota$ and $\mathfrak{r}(\mathbb{C}) = \mathfrak{d}(\mathbb{C}) = \mathbb{R}, \ldots$ In this example we want to model a taxonomy for elementary calculus, so let Σ be the set containing the subsort declarations $[\mathbb{P} \leq \mathbb{D}], [\mathbb{D} \leq \mathbb{C}]$, and the term declarations

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

for polynomials and furthermore $[\partial{::}\mathbb{D} \to \mathbb{C}], [\partial{::}\mathbb{P} \to \mathbb{P}]$ for the differentiation operator ∂, then it is easy to check that Σ is a valid signature. We can see that we have coded a great deal of information about polynomials and differentiation into the term declarations of Σ. Note that up to (elementary arithmetic) any polynomial is indeed of sort $\mathbb{P}$. The practical advantage of this formalization of elementary algebra is that this can be used in the unification during proof search in refutation calculi and thus considerably reduce the search for proofs.

3 Structure Theorem and General Bindings

The key tool for the investigation of well-sorted formulae will be the structure theorem which we are about to prove. The principal difficulty of $\Sigma\mathcal{T}$ is that the property of well-sortedness is highly non-structural, which makes the classical deduction methods, such as unification that analyze the structure of formulae difficult. The structure theorem recovers structural properties of well-sorted formulae by linking the sort information (the existence of certain term declarations) with structural information about normal forms.

Theorem 3.1 (Structure Theorem) *Let* $\mathbf{A}$ *be a well-sorted formula with long head normal form* $[\lambda \overline{X^k}.h\overline{\mathbf{U}^n}]$ *and* $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$*. Furthermore let* Ξ^j *be the variable context* $[X^1{::}\mathfrak{d}(\mathbb{A})], \ldots, [X^j{::}\mathfrak{d}^j(\mathbb{A})]$ *and* $l = \mathbf{ln}(\mathbb{A})$*, then*

1. *h is a variable with* $\Gamma, \Xi^k(h) = \mathbb{B}$*, such that* $\mathfrak{r}^n(\mathbb{B}) = \mathfrak{r}^k(\mathbb{A})$, $\Gamma, \Xi^k \vdash_\Sigma \mathbf{U}^i{::}\mathfrak{d}^i(\mathbb{B})$ *for* $1 \leq i \leq n$*.*
2. *there is a term declaration* $[\forall\Delta.\mathbf{B}{::}\mathbb{B}] \in \Sigma$*, a* Σ*-substitution* θ *and well-sorted formulae* $\mathbf{D}^i$*, such that*
   (a) $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} (\lambda \overline{X^l_{\mathfrak{d}^l(\mathbb{A})}}.\theta(\mathbf{B})\overline{\mathbf{D}^m})$*, where* $m := l + \mathbf{ln}(\tau(\mathbb{B})) - \mathbf{ln}(\tau(\mathbb{A})) \geq 0$*.*
   (b) $\Gamma, \Xi^l \vdash_\Sigma \theta{::}\Delta$, $\Gamma, \Xi^l \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{B})$ *for all* $i \leq m$ *and* $\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^l(\mathbb{A})$*.*
   (c) *If h is a constant, then* $\mathbf{head}(\mathbf{B}) = Z \in \mathbf{Dom}(\theta)$ *and* $\mathbf{head}(\theta(Z)) = h$ *or else* $\mathbf{head}(\mathbf{B}) = h$*.*
   (d) $\mathbf{dp}(\mathbf{D}^i) < \mathbf{dp}(h\overline{\mathbf{U}^n})$ *and* $\mathbf{dp}(\theta) < \mathbf{dp}(h\overline{\mathbf{U}^n})$*.*

Here the depth of a substitution θ is the maximum of the $\mathbf{dp}(\theta(X))$ *for all* $X \in \mathbf{Dom}(\theta)$ *and the depth of a formula is the depth of the corresponding tree.*

One of the key steps in sort computation and unification is solving the following problem: given a sort $\mathbb{A}$ and an atom $\mathbf{C}$, find the most general well-sorted formula of sort $\mathbb{A}$ that has head $\mathbf{C}$. Such formulae are called general bindings of sort $\mathbb{A}$ for the head $\mathbf{C}$. In $\Sigma\mathcal{T}$, this problem requires a more careful investigation than in $\Lambda^{\rightarrow}$. For instance consider a context Γ, such that $\Gamma(Z) = \mathbb{B} \rightarrow \mathbb{B}$, $\Gamma(X) = \Gamma(Y) = \mathbb{B}$ and $\Gamma(W) = \mathbb{A}$ and Σ consists of the following term declarations $[\mathbb{A} \leq \mathbb{B}]$, $[a{::}\mathbb{A}]$, $[b{::}\mathbb{B}]$, $[f{::}(\mathbb{B} \rightarrow \mathbb{B} \rightarrow \mathbb{B})]$, $[\forall[X{::}\mathbb{B}].(faX){::}\mathbb{A}]$, $[\forall[X{::}\mathbb{B}].(fXb){::}\mathbb{A}]$ then the most general formulae with the head f and sort $\mathbb{B}$ is fXY, of sort $\mathbb{A}$ are faX and fXb and finally of sort $(\mathbb{B} \rightarrow \mathbb{A})$ are $\lambda X_{\mathbb{B}}.fa(ZX)$ and $\lambda X_{\mathbb{B}}.f(ZX)b$. In $\Lambda^{\rightarrow}$ these general bindings are unique and consist only of the head and of variables. In order-sorted type-theory each term declaration, that has the appropriate head and meets certain conditions will contribute a general binding.

Definition 3.2 (General Binding) For the definitions of general bindings we have two possibilities, corresponding to the two cases of the structure theorem. The first (classical) one obtains the sort information from the head variable, whereas the second one obtains the sort information from a term declaration. Let Γ be a context and $\mathbb{A}$ and $\mathbb{B}$ be sorts, such that

1. $l = \mathbf{ln}(\mathbb{A})$ and $m = l + \mathbf{ln}(\tau(\mathbb{A})) - \mathbf{ln}(\tau(\mathbb{B})) \geq 0$
2. $\mathfrak{r}^m(\mathbb{B}) = \mathfrak{r}^l(\mathbb{A})$
3. $\mathbf{V}^i = (H^i X^1 \ldots X^l)$, where H^i are variables not in $\mathbf{Dom}(\Gamma)$
4. $\mathcal{H} := [H^1{::}\overline{\mathfrak{d}^l(\mathbb{A})} \rightarrow \mathfrak{d}^1(\mathbb{B})], \ldots, [H^m{::}\overline{\mathfrak{d}^l(\mathbb{A})} \rightarrow \mathfrak{d}^m(\mathbb{B})]$,

then the formula $G := (\lambda X^1_{\mathfrak{d}^1(\mathbb{A})} \ldots X^l_{\mathfrak{d}^l(\mathbb{A})}.h\mathbf{V}^1 \ldots \mathbf{V}^m)$ is called a *general binding of sort* $\mathbb{A}$ *and head* h if $h = X^j$ or $h \in \mathbf{Dom}(\Gamma)$ and $\Gamma(h) = \mathbb{B}$. We call $\mathcal{H}$ the *context of variables introduced for* $\mathbf{G}$.

Let $[\forall\overline{[Y^t{::}\mathbb{C}^n]}.\mathbf{B}{::}\mathbb{B}] \in \Sigma$ and $\mathbb{A}$, $\mathbb{B}$ and $\mathbf{V}^i$ as above and furthermore

5. $\mathbf{W}^i := (K^i X^1 \ldots X^l)$, where K^i are variables not in $\Gamma \cup \mathcal{H}$
6. $\mathcal{K} := \underline{[K^1{::}\overline{\mathfrak{d}^l(\mathbb{A})} \rightarrow \mathbb{C}^1], \ldots, [K^t{::}\overline{\mathfrak{d}^l(\mathbb{A})} \rightarrow \mathbb{C}^t]}$
7. $\mathbf{B}' = \overline{[\mathbf{W}^t/Y^n]}\mathbf{B}$ and $h := \mathbf{head}(\mathbf{B}')$

then the formula $G := (\lambda X^1_{\mathfrak{d}^1(\mathbb{A})} \ldots X^l_{\mathfrak{d}^l(\mathbb{A})}.\mathbf{B}'\mathbf{V}^1 \ldots \mathbf{V}^m)$ is called a *general binding of sort* $\mathbb{A}$ *and head* h. In this case the context of variables introduced for $\mathbf{G}$ is $\mathcal{H} \cup \mathcal{K}$. Now we define the set $\mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$ to be the set of all general bindings of sort $\mathbb{A}$ and head h and introduced context $\mathcal{C}$. If the head of $\mathbf{G}$ is bound, then we call $\mathbf{G}$ a *projection binding*, if h is a variable in $\mathbf{Dom}(\Gamma)$ or a constant *imitation binding* and if $h \in \mathbf{Dom}(\mathcal{K})$ ($\mathbf{G}$ is induced by a flexible term declaration in this case), then we call $\mathbf{G}$ a *general weakening binding of sort* $\mathbb{A}$. We will denote the set of all such bindings with $\mathcal{W}_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$.

It is easily verified that $\Gamma, \mathcal{C} \vdash_{\Sigma} \mathbf{G}{::}\mathbb{A}$ and $\mathbf{head}(\mathbf{G}) = h$, provided that $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$, which explains the naming in the definition above. The following theorem is a consequence of the structure theorem and the basis of the unification transformations given below.

Theorem 3.3 (General Binding Theorem) *Let* $\mathbf{A} = \lambda\overline{X^k}.h\overline{\mathbf{U}^n}$ *be a long* $\beta\eta$*-normal form with* $\Gamma \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A}$*, then there exists a general binding* $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C}) \cup \mathcal{W}_{\mathbb{A}}(\Sigma, \Gamma, \mathcal{C})$ *and a* Σ*-substitution* $\rho \in \mathbf{wsSub}(\mathcal{C} \rightarrow \Gamma, \Sigma)$*, such that* $\mathbf{dp}(\rho) < \mathbf{dp}(\mathbf{A})$ *and* $\mathcal{C}, \Gamma \vdash_{\Sigma} \rho(\mathbf{G}) =_{\beta\eta} \mathbf{A}$.

4 Unification

Building upon the notion of general bindings we give a set of transformations for (pre-)unification, which we will prove correct and complete with the methods of [20].

Definition 4.1 (Unification Problem) A *unification problem* is a formula in the language $\mathcal{E} ::= \mathbf{A} \doteq \mathbf{B} \mid \exists[X :: \mathbb{A}]\mathcal{E} \mid \forall[X :: \mathbb{A}]\mathcal{E} \mid \mathcal{E}_1 \wedge \mathcal{E}_2 \mid \top$. In order to simplify the presentation of the algorithm, we assume that all unification formulae are in $\exists\forall$-form $\mathcal{E} := \exists\Gamma\forall\Delta.\mathcal{E}'$. Each formula is equivalent to one in this form by raising [15]. We call a Σ-substitution θ, such that $\Gamma \vdash_\Sigma \theta :: \Delta$ and $\Gamma \vdash_\Sigma \theta(\mathbf{A}) {=_{\beta\eta}} \theta(\mathbf{B})$ for all pairs $\mathbf{A} \doteq \mathbf{B}$ in $\mathcal{E}'$ a Σ-unifier for $\mathcal{E}$ and we will denote the set of Σ-unifiers of $\mathcal{E}$ with $\mathbf{wsU}(\Sigma, \mathcal{E})$. We call a subset $\Psi \subseteq \mathbf{wsU}(\Sigma, \mathcal{E})$ a *complete set of* Σ*-unifiers of* $\mathcal{E}$, iff for all $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$ there is a $\sigma \in \Psi$ that is more general than θ, *i.e.* there is a Σ-substitution ρ, such that $\Gamma \vdash_\Sigma \sigma(X) {=_{\beta\eta}} \rho(\theta(X))$ for all $X \in \mathbf{Dom}(\Delta) = \mathbf{Dom}(\sigma)$. If the singleton set $\{\sigma\}$ is a complete set of unifiers of $\mathcal{E}$, then we call σ a *most general unifier* for $\mathcal{E}$.

Note that Σ-unifiability does not entail that both formulae of a pair have identical sets of sorts, since these sets may grow as more term declarations become applicable with instantiation. Nevertheless Σ-unifiable pairs must have the same types and furthermore the sorts must obey the **Rdom** relation.

Definition 4.2 (Σ-Solved Form) A unification problem $\mathcal{E} := \exists\Gamma\forall\Delta.\mathcal{E}'$ is in Σ-*solved form* if all of its pairs are in solved form, *i.e.* of the form $X \doteq \mathbf{A}$, such that $\Gamma(X) = \mathbb{A}$, $\Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}$, neither X nor any $Y \in \mathbf{Dom}(\Delta)$ is free in $\mathbf{A}$, and X is not free elsewhere in $\mathcal{E}$. These conditions are sufficient to ensure that $\sigma_\mathcal{E} = [\mathbf{A}^1/X_1, \ldots, \mathbf{A}^n/X^n]$ is a most general unifier for $\mathcal{E}$ provided that $\mathcal{E}$ is in solved form with matrix $X^1 \doteq \mathbf{A}^1 \wedge \ldots \wedge X^n \doteq \mathbf{A}^n$

Definition 4.3 ($\mathcal{SIM}$: Simplification of Σ-Unification Problems)

$$\frac{\exists\Delta.\forall\Upsilon.(\lambda X_{\mathbb{A}}.\mathbf{A}) \doteq (\lambda Y_{\mathbb{A}}.\mathbf{B})}{\exists\Delta.\forall\Upsilon, [X :: \mathbb{A}].\mathbf{A} \doteq [X/Y]\mathbf{B}} \qquad \frac{\exists\Delta.\forall\Upsilon.(\lambda X_{\mathbb{A}}.\mathbf{A}) \doteq \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} :: \mathbb{B} \quad \eth(\mathbb{B}) = \mathbb{A}}{\exists\Delta.\forall\Upsilon, [X :: \mathbb{A}].\mathbf{A} \doteq (\mathbf{B}X)}$$

We apply these rules with the understanding that the operators $\wedge$ and $\doteq$ are commutative and associative, that trivial pairs may be dropped and that vacuous quantifications can be eliminated from the prefix. It is easy to see that these simplifications conserve the sets of Σ-unifiers.

Definition 4.4 ($\Sigma\mathcal{UT}$: Transformations for Σ-Unification)
Let $\Sigma\mathcal{UT}$ be the system $\mathcal{SIM}$ augmented by the following inference rules

$$\frac{\exists\Delta.\forall\Upsilon.h\overline{\mathbf{U}^n} \doteq h\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in \overline{\Sigma} \cup \mathbf{Dom}(\Delta) \cup \mathbf{Dom}(\Upsilon)}{\exists\Delta.\forall\Upsilon.\mathbf{U}^1 \doteq \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n \doteq \mathbf{V}^n \wedge \mathcal{E}}$$

together with the following rules where $\mathbf{G}$ is a general binding of sort $\mathbb{A}$ in $\mathcal{G}^h(\Sigma, \Delta, \mathcal{C}) \cup \mathcal{G}^j(\Sigma, \Delta, \mathcal{C}) \cup \mathcal{G}^w(\Sigma, \Delta, \mathcal{C})$

$$\frac{\exists \Delta.\forall \Upsilon.F\overline{\mathbf{U}} \doteq h\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Delta(F) = \mathbb{A}}{\exists \Delta \cup \mathcal{C}.\forall \Upsilon.F \doteq \mathbf{G} \wedge [\mathbf{G}/f](F\overline{\mathbf{U}} \doteq h\overline{\mathbf{V}} \wedge \mathcal{E})} *$$

$$\frac{\exists \Delta.\forall \Upsilon.F\overline{\mathbf{U}} \doteq h\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Delta(F) = \mathbb{A} \quad \Delta(h) = \mathbb{B}}{\exists \Delta \cup \mathcal{C}.\forall \Upsilon.F \doteq \mathbf{G} \wedge [\mathbf{G}/F](F\overline{\mathbf{U}} \doteq h\overline{\mathbf{V}} \wedge \mathcal{E})} **$$

Just as in $\mathcal{SIM}$ leave the associativity and commutativity of $\wedge$ and $\doteq$ implicit. Note that the concept of a weakening transformation for unification is new to $\Sigma\Lambda^{\rightarrow}$, where we use term declarations to model subsorting. We have combined it with the classical imitation ($\mathbf{G}$ has head h) and projection ($\mathbf{G}$ is a projection binding) transformations (see [20]) into $*$. This set of rules is used with the convention that all formulae are eagerly reduced to β-normal form.

Since we have captured the relevant features of $\Sigma\mathcal{T}$ in the structure and general binding theorems (both of which are nearly trivial in $\Lambda^{\rightarrow}$), we can now use the standard techniques (cf. [20, 9]) to soundness and completeness.

Theorem 4.5 (Completeness Theorem for $\Sigma\mathcal{UT}$) *For any well-sorted unification problem $\mathcal{E}$ and any $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$, there is a sequence of transformations in $\Sigma\mathcal{UT}$, such that $\mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{E}'$, where $\mathcal{E}'$ is in Σ-solved form and $\sigma_{\mathcal{E}'} \preceq_{\beta\eta} \theta[\mathbf{Free}(\mathcal{E})]$.*

As for unification in $\Lambda^{\rightarrow}$, the rule $**$ gives rise to a serious explosion of the search space for unifiers. Huet's solution to this problem was to redefine the higher-order unification problem to a form sufficient for refutation purposes: For the pre-unification problem flex-flex pairs are considered already solved, since they can always be trivially solved by binding the head variables to special constant functions that identify the formulae by absorbing their arguments.

However in $\Sigma\mathcal{T}$ the solution to the flex-flex problem is not as simple as in the unsorted case, since the heads of flex-flex pairs can be variables of functional base sorts $\mathbb{A}$. In this case flex-flex-pairs are not solvable independently of their arguments, since in general the constant functions needed for absorbing the arguments are not of sort $\mathbb{A}$. Our solution to this problem is to modify the definition of pre-solved pairs and to keep the guess rule, but restrict its application to the problematic flex-flex cases. Furthermore Σ-pre-unification only makes sense for regular signatures, where formulae have a unique least sort (with respect to the full subsort relation). Consider the non-regular signature given by $\mathcal{S} := \{\mathbb{A}, \mathbb{B}\}$, $\tau(\mathbb{A}) = \tau(\mathbb{B}) = \alpha$, $\overline{\Sigma} := \{c_\alpha\}$ and $\Sigma := \{[c{::}\mathbb{A}], [c{::}\mathbb{B}]\}$. The Σ-substitution $[c/X], [c/Y]$ is a Σ-unifier of the pair $\exists [X{::}\mathbb{A}], [Y{::}\mathbb{B}].X \doteq Y$, but it can only be found by applying some kind of $**$ transformation. Therefore we will only consider regular signatures for pre-unification.

Definition 4.6 Let $\mathbf{A}$ be a flexible formula with $\beta\eta$-normal form $\lambda\overline{X}.F\overline{\mathbf{U}^n}$ and $\Gamma(F) = \overline{\mathbb{A}^n} \rightarrow \mathbb{B}$, then we call $\mathbf{A}$ *functionally flexible with target sort* $\mathbb{B}$. Let $=^p$ be the least congruence relation on well-sorted formulae that contains $=_{\beta\eta}$ and all functional flexible pairs. Let $\mathcal{E} := \exists\Gamma\forall\Delta.\mathcal{E}'$ be an equational system, then a

Σ-substitution σ is called a Σ-pre-unifier of the pair $\mathbf{A} \doteq \mathbf{B} \in \mathcal{E}'$, iff $\Gamma \vdash_{\Sigma} \sigma{::}\Delta$ and $\Gamma \vdash_{\Sigma} \sigma(\mathbf{A}) =^{p} \sigma(\mathbf{B})$. We denote the set of Σ-pre-unifiers by $\mathbf{wsPU}(\Sigma, \mathcal{E})$.

Definition 4.7 (Pre-Solved Form) Let $\mathcal{E} := \exists\Gamma\forall\Delta.\mathcal{E}'$ be an equational system, then we call a formula $F\overline{\mathbf{U}^{k}Y}$ *functionally flexible in* $\mathcal{E}$ *with target sort* $\mathbb{B}$, iff $\Gamma(F) = \overline{\mathbb{A}^{k}} \to \mathbb{B}$ and $Y^{i} \in \mathbf{Dom}(\Delta)$. A pair in $\mathcal{E}'$ is in Σ-*pre-solved form* in $\mathcal{E}$, iff it is in solved form, or if it is a pair of functionally flexible formulae with identical target sorts.

This definition is tailored to guarantee that Σ-pre-unifiers can always be extended to Σ-unifiers by finding trivial unifiers for the flexible pairs and that equational problems in Σ-pre-solved form always have most general unifiers. Therefore an equational system $\mathcal{E}$ is Σ-pre-unifiable, iff it is Σ-unifiable.

Definition 4.8 ($\Sigma\mathcal{PT}$:Transformations for Σ-Pre-Unification) We define the set $\Sigma\mathcal{PT}$ of *transformations for well-sorted pre-unification* by modifying the $\Sigma\mathcal{UT}$ rules for decomposition and the rule $*$ by requiring that they may not be performed on a pair $\mathbf{A} \doteq \mathbf{B}$, if $\mathbf{head}(\mathbf{A}) \in \mathbf{Free}(\mathbf{A})$, and restricting $**$ to the case, where the variable it acts on is the head of a flexible pair that is not functionally flexible.

With these definitions we obtain a completeness result for $\Sigma\mathcal{PT}$ similar to 4.5 with the same methods, since most of the technical difficulties are encapsulated in the general binding theorem. In fact these methods can also be extended to yield a Σ-unification algorithm for higher-order patterns (cf. [12]).

5 Conclusion and Further Work

We have presented a sorted version $\Sigma\mathcal{T}$ of $\Lambda^{\to}$ that incorporates the notions of functional base sorts and term declarations, which is a a good basis for the development of higher-order automated theorem provers, since it greatly enhances the practical expressive power of $\Lambda^{\to}$ as a logic system. We have studied the subtle interactions of functional base sorts, function restriction and extensionality, and of term declarations with sorted β-conversion. We have presented correct and complete sets of transformations for unification and pre-unification in $\Sigma\mathcal{T}$, which form the basis of a sorted higher-order resolution calculus described in [13].

In first-order predicate logic, the introduction of term declarations has been a major step to the development of dynamic sorted logics [22], where variables are restricted to sorts, but where the sorts can also be treated as unary predicates in the logic; thus the signature is no longer fixed across the deduction, as sort information can appear in the deduction process. Extensions of these ideas have been utilized to formalize and mechanize a general first-order Kleene logic for partial functions [10]. In both systems 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 the methods in [22, 10] may yield very powerful calculi for mechanizing mathematics and in particular analysis, which was the original motivation for the research reported in this paper.

References

1. Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Academic Press, 1986.
2. Peter B. Andrews, Eve Longini-Cohen, Dale Miller, and Frank Pfenning. Automating higher order logics. *Contemp. Math*, 29:169–192, 1984.
3. Kim B. Bruce and Giuseppe Longo. A modest model of records, inheritance and bounded quantification. *Information and Computation*, 87:196–240, 1990.
4. Luca Cardelli. A semantics of multiple inheritance. In G. Kahn and G. Plotkin D.G. MacQueen, editors. *Semantics of Data Types*, volume 173 of LNCS. Springer Verlag, 1984.
5. Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.
6. A. G. Cohn. Taxonomic reasoning with many-sorted logics. *Artificial Intelligence Review*, 3:89–128, 1989.
7. Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic.* PhD thesis, Case Western Reserve University, 1972.
8. 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.
9. Patricia Johann and Michael Kohlhase. Unification in an extensional lambda calculus with ordered function sorts and constant overloading. In Proc. CADE'94, *LNCS*,Springer Verlag, 1994.
10. Manfred Kerber and Michael Kohlhase. A mechanization of strong Kleene logic for partial functions. In Proc. CADE'94, *LNCS* Springer Verlag, 1994.
11. Michael Kohlhase. Unification in order-sorted type theory. In Proc. LPAR'92, pages 421–432, volume 624 of *LNAI*. Springer Verlag, 1992.
12. Michael Kohlhase. *Automated Deduction in Order-Sorted Type Theory.* PhD thesis, Universität des Saarlandes, 1994. to appear.
13. Michael Kohlhase. Higher-order order-sorted resolution. Seki Report SR-94-01, FB Informatik, Universität des Saarlandes, 1994.
14. Michael Kohlhase and Frank Pfenning. Unification in a λ-calculus with intersection types. In Proc. ILPS'93, pages 488–505. MIT Press, 1993.
15. Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, pages 321–358, 1992.
16. Tobias Nipkow and Zhenyu Qian. Reduction and unification in lambda calculi with subtypes. In Proc. CADE'92 volume 607 of *LNCS*, pages 66–78, 1992. Springer Verlag.
17. Benjamin C. Pierce. *Programming with Intersection Types and Bounded Polymorphism.* PhD thesis, Carnegie Mellon University, 1991.
18. Zhenyu Qian. *Extensions of Order-Sorted Algebraic Specifications: Parameterization, Higher-Functions and Polymorphism.* PhD thesis, Universität Bremen, 1991.
19. Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *LNAI*. Springer Verlag, 1989.
20. Wayne Snyder. *A Proof Theory for General Unification.* Birkhäuser, 1991.
21. Christoph Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Pitman, London. Morgan Kaufman Publishers, Inc, 1987.
22. C. Weidenbach. Unification in sort theories and its applications. MPI-Report MPI-I-93-211, MPI Informatik, Saarbrücken, 1993.