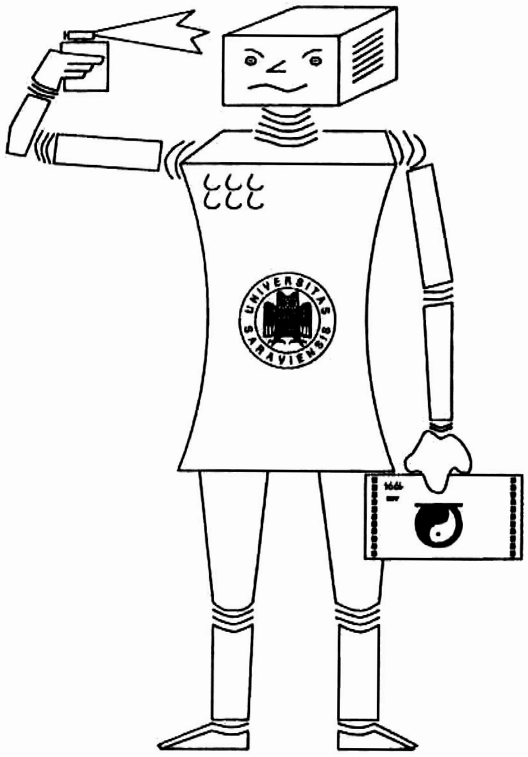
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://jswww.cs.uni-sb.de/pub/www/



SEKI Report

A Colored Version of the λ-Calculus

Dieter Hutter, Michael Kohlhase

SEKI Report SR-95-05

A Colored Version of the λ-Calculus

Dieter Hutter
German Research Center for
Artificial Intelligence
D-66123 Saarbrücken

Michael Kohlhase*
Universität des Saarlandes
D-66141 Saarbrücken

Abstract

Coloring terms (rippling) is a technique developed for inductive theorem proving which uses syntactic differences of terms to guide the proof search. Annotations (colors) to terms are used to maintain this information. This technique has several advantages, e.g. it is highly goal oriented and involves little search. In this paper we give a general formalization of coloring terms in a higher-order setting. We introduce a simply-typed lambda calculus with color annotations and present an appropriate (pre-)unification algorithm. Our work is a formal basis to the implementation of rippling in a higher-order setting which is required e.g. in case of middle-out reasoning.

Another application is in the construction of natural language semantics, where the color annotations rule out linguistically invalid readings that are possible using standard higher-order unification.

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

Contents

1 Introduction

In the field of inductive theorem proving syntactical differences between the induction hypothesis and induction conclusion are used in order to guide the proof [Bun88, BSvH+93], or [Hut90, Hut91]. This method to guide induction proofs is called rippling / coloring terms. Annotations or colors to each occurrence of a symbol are used to mark the syntactical differences between induction hypothesis and induction conclusion. Specific colors denote the *skeleton*, the common parts of both formulas while the other parts belong to the *wave-fronts*. Analogously, syntactical differences between both sides of equations or implications given in the database are colored. These formulas are classified depending on the locations of the wave-fronts inside the skeleton (e.g. wave-fronts on both sides, wave-fronts only on the right-hand side, or wave-fronts only on the left-hand side). Using these annotated (or colored) equations we are able to move, insert, or delete wave-fronts within the conclusion. This *rippling* of wave-fronts allows to reduce the differences between conclusion and hypothesis in a goal directed way.

This paper extends the coloring method to higher-order logic and presents unification and pre-unification algorithms that we prove correct and complete. Thus our work provides a formal basis to the implementation of rippling in a higher-order setting which is required e.g. in case of middle-out reasoning [Hes91] or generalization of theorems using proof critics [IB96]. In the latter the unknown generalized version of a formula is described by a pattern containing parts of the original formula and higher-order variables denoting the unknown syntactical extensions of it. Simulating the induction proof the higher-order variables will be instantiated step by step by the unification with appropriate wave-rules resulting in a possible (hopefully provable) generalization of the original formula.

But the set of possible applications of our method is not limited to automated deduction. From an abstract point of view, the coloring technique allows adding annotations to symbol occurrences in λ-terms. Thus in contrast to other semantic annotation techniques like sorts, it is possible to encode syntactic information and use that to guide inferencing processes. In section 2.3 we will briefly sketch an application [GK96] of our technique in computational linguistics and natural language semantics, where the use of higher-order unification is used to construct the values of elliptical references [DSP91] in the context of Montegovian semantics.

1.1 Informal Exposition

Since the formal development of the theory involves quite a lot of technical machinery, we will begin with a rather informal account first. The references in brackets indicate, where the reader can find a fully formal development of the respective informal arguments.

The colored λ-calculus is a variant of the simply typed λ-calculus [Chu40], where symbol occurrences can be annotated with so-called *colors* (**color constants** $\mathcal{C} = \{\mathtt{a}, \mathtt{b}, \ldots\}$ and **color variables** $\mathcal{X} = \{\mathtt{A}, \mathtt{B}, \ldots\}$; whenever colors are irrelevant, we simply omit them; colors are indicated by subscripts labeling symbol occurrences).

The set $w\!f\!f_\alpha$ of **well-formed formulae of type**[1] α (3.6) consists of

- (colored) constants $c^\alpha, c^\alpha_{\mathtt{b}}, f^\alpha_{\mathtt{a}}, f^\alpha_{\mathtt{A}}, \ldots,$
- (colored or uncolored) variables $X^\alpha_{\mathtt{b}}, Y^\alpha_{\mathtt{a}}, F^{\alpha\to\beta}\ldots$ (of which we assume an infinite supply for each type and color) of type α,

[1]Since for the purposes of this informal introduction types only play a theoretical role (they for instance make $\beta\eta$-reduction terminating and therefore $\beta\eta$-equality decidable), they are often omitted from the examples.

- (function) **applications** of the form $\mathbf{M}^{\beta\to\alpha}\mathbf{N}^{\beta}$ and
- **λ-abstractions** of the form $\lambda X^{\beta}.\mathbf{M}^{\gamma}$, if $\alpha = (\beta \to \gamma)$. Note that only variables without colors can be abstracted over.

We call a formula **M** **c-monochrome** (3.8), if all symbols in **M** are annotated by a single color c.

Clearly the colored λ-calculus is a generalization of the simply typed λ-calculus, since we can always restrict the supply of colors to a single color constant. Therefore we will use various elementary concepts of the λ-calculus, such as **free** and **bound** occurrences of variables or substitutions without defining them explicitly here.We will denote the substitution of a term **N** for all free occurrences of X in **M** with $[\mathbf{N}/X]\mathbf{M}$. Furthermore we will follow the λ-calculus tradition and consider formulae that only differ in the names of the bound variables as identical and we will require that the substitution process renames all bound variables in **M** in order to avoid variable capture.

It is crucial for our system that colors annotate symbol occurrences (i.e. colors are not sorts!), in particular, it is intended that different occurrences of symbols carry different colors (e.g. $f(X_{\mathtt{a}}, X_{\mathtt{b}})$) and that symbols that carry different colors are treated differently. This observation leads to the notion of colored substitutions (3.13), a notion of substitution that takes the color information of formulae into account. In contrast to traditional (uncolored) substitutions, a colored substitution σ is a pair $\langle\sigma^t, \sigma^c\rangle$, where the **term substitution** σ^t maps colored variables (i.e. the pair X_c of a variable X and the color c) to formulae of appropriate types and the **color substitution** σ^c maps color variables to colors. In order to be a legal **$\mathcal{C}$-substitution** such a mapping σ must obey the following constraints:

- If A and B are different colors, then $|\sigma(X_{\mathtt{A}})| = |\sigma(X_{\mathtt{B}})|$, where $|\mathbf{M}|$ is the **color erasure** of **M**, i.e. the formula obtained from **M** by erasing all color annotations in **M** (3.12).
- If $\mathtt{c} \in \mathcal{C}$ is a color constant, then $\sigma(X_{\mathtt{c}})$ is c-monochrome.

The first condition ensures that the color erasure of a $\mathcal{C}$-substitution is a classical substitution of the simply typed λ-calculus. The second condition formalizes the fact that free variables with constant colors stand for monochrome subformulae, whereas variable colors do not constrain the substitutions.

We say that **M** can be β-reduced (3.10) to **N**, iff **N** is obtained from **M** by applying the following replacement rule

$$(\lambda X.\mathbf{P})\mathbf{Q} \longrightarrow_{\beta} [\mathbf{Q}/X]\mathbf{P}$$

to a well-formed subformula of **M**. This rule relates the evaluation of functions (defined using λ-abstraction) to the process of substituting the argument **Q** for all formal parameters X in **P**. We obtain the notion of η-reduction (3.10) with the rule

$$\lambda X.\mathbf{P}X \longrightarrow_{\eta} \mathbf{P}$$

where the variable X is not free in **P**. If formulae **M** and **N** are convertible using a sequence of these reduction rules and their inverses ($\mathbf{M} =_{\beta\eta} \mathbf{M}$), they denote the same object or function. Thus $\beta\eta$-equality is considered to be built into the λ-calculus. In particular we will solve all equations used in our analysis with respect to $\beta\eta$-equality.

Note that since the bound variables do not carry color information, $\beta\eta$-reduction in the colored λ-calculus is just the classical notion. Thus we can lift all the known theoretical results to the colored calculus, such as the fact that $\beta\eta$-reduction always terminates producing unique normal forms and that $\beta\eta$-equality can be tested by reducing to normal form and comparing for syntactic equality. This gives us a decidable test for *validity* of an equation.

In contrast to this, higher-order unification (4.2) tests for *satisfiability* by finding a substitution σ that makes a given equation $\mathbf{M} = \mathbf{N}$ valid ($\sigma(\mathbf{M}) =_{\beta\eta} \sigma(\mathbf{N})$), even if the original equation is not ($\mathbf{M} \neq_{\beta\eta} \mathbf{N}$). In the colored λ-calculus the space of (semantic) solutions is further constrained by requiring the solutions to be $\mathcal{C}$-substitutions. Such a substitution is called a $\mathcal{C}$-unifier (4.2) of $\mathbf{M}$ and $\mathbf{N}$. In particular, $\mathcal{C}$-unification will only succeed if comparable formulae have unifiable colors. For instance, $f_{\mathtt{a}}(p_{\mathtt{a}}, j_{\mathtt{b}}, X_{\mathtt{a}})$ unifies with $f_{\mathtt{a}}(Y_{\mathtt{a}}, j_{\mathtt{A}}, s_{\mathtt{a}})$ but not with $f_{\mathtt{a}}(p_{\mathtt{a}}, j_{\mathtt{a}}, s_{\mathtt{a}})$ because of the color clash on j.

Even with the color restriction, the set of $\mathcal{C}$-unifiers of a given equation is enormous. Furthermore, most of these solutions introduce un-necessary instantiations; thus one is not interested in the set of *all* $\mathcal{C}$-unifiers, but rather in a subset that generates this set by instantiation. A substitution σ is called **more general** (4.2) than τ, iff there is a substitution ρ, such that $\tau =_{\beta\eta} \rho \circ \sigma$, i.e. τ can be reconstructed from σ by instantiation with ρ.

It is well-known, that in first-order logic (and in certain related forms of feature structures) there is always a most general unifier for any equation that is solvable at all. This is not the case for higher-order (colored) unification, where variables can range over functions, instead of only individuals. In fact there can even be solvable equations that have infinite chains of unifiers that become more and more general. In other words most general unifiers need not to exist in general.

1.2 Calculating Colored Unifiers

Just as in the case of unification for first-order terms, the higher-order unification algorithm is a process of recursive decomposition and variable elimination that transform sets of equations into solved forms. Since $\mathcal{C}$-substitutions have two parts, a term- and a color part, we need two kinds of equations ($\mathbf{M} =^t \mathbf{N}$ for term equations and $c =^c d$ for color equations). Sets $\mathcal{E}$ of equations in solved form (i.e. where all equations are of the form $X = \mathbf{M}$ such that the variable X does not occur anywhere else in $\mathbf{M}_c$ or $\mathcal{E}$) have a unique most general $\mathcal{C}$-unifier $\sigma_{\mathcal{E}}$ that also $\mathcal{C}$-unifies the initial equation (4.3).

There are several rules (4.9) that decompose the syntactic structure of formulae, we will only present two of them. The rule for abstractions transforms equations of the form $\lambda X.A =^t \lambda Y.B$ to $[Z/X]A =^t [Z/Y]B$, where Z is a new constant, while the rule for applications decomposes $h_{\mathtt{a}}(s^1, \ldots, s^n) =^t h_{\mathtt{b}}(t^1, \ldots, t^n)$ to the set $\{a =^c b, s^1 =^t t^1, \ldots, s^n =^t t^n\}$, provided that h is a constant. Furthermore equations are kept in $\beta\eta$-normal form. Note that this decomposition process also eliminates trivial equations, where both sides are $\beta\eta$-equal.

The variable elimination process for color variables is very simple, it allows to transform a set $\mathcal{E} \cup \{\mathtt{A} =^c \mathtt{d}\}$ of equations to $[\mathtt{d}/\mathtt{A}]\mathcal{E} \cup \{\mathtt{A} =^c \mathtt{d}\}$, making the equation $\{\mathtt{A} =^c \mathtt{d}\}$ solved in the result. In case of formula equations, elimination is not that simple, since we have to ensure that $|\sigma(X_{\mathtt{A}})| = |\sigma(X_{\mathtt{B}})|$ to obtain a $\mathcal{C}$-substitution σ. Thus we cannot simply transform a set $\mathcal{E} \cup \{X_{\mathtt{d}} =^t \mathbf{M}\}$ into $[\mathbf{M}/X_{\mathtt{d}}]\mathcal{E} \cup \{X_{\mathtt{d}} =^t \mathbf{M}\}$, since this would (incorrectly) solve the equations $\{X_{\mathtt{c}} = f_{\mathtt{c}}, X_{\mathtt{d}} = g_{\mathtt{d}}\}$. The correct variable elimination rule transforms $\mathcal{E} \cup \{X_{\mathtt{d}} =^t \mathbf{M}\}$ into $\sigma(\mathcal{E}) \cup \{X_{\mathtt{d}} =^t \mathbf{M}, X_{\mathtt{c}_1} = \mathbf{M}^1, \ldots, X_{\mathtt{c}_n} =^t \mathbf{M}^n\}$, where $\mathtt{c}_i$ are all colors of the variable X occurring in $\mathbf{M}$ and $\mathcal{E}$, the $\mathbf{M}^i$ are appropriately colored variants (same color erasure) of $\mathbf{M}$, and σ is the $\mathcal{C}$-substitution that eliminates all occurrences of X from $\mathcal{E}$.

It would be convenient, if the transformations described so far, were sufficient for transforming all unifiable sets of equations into solved form and thus finding all unifiers. But, due to the presence of function variables, systematic application can terminate with equations of the form $X_{\mathsf{c}}(s^1,\ldots,s^n) =^t h_{\mathsf{d}}(t^1,\ldots,t^m)$. Such equations can neither be further decomposed by the rules above, since this would loose unifiers, nor can the right hand side be substituted for X as in a variable elimination rule, since the types would clash. Let us consider the uncolored equation $X(a) =^t a$ which has the solutions $(\lambda Z.a)$ and $(\lambda Z.Z)$ for X.

The standard solution (4.14) for finding a complete set of solutions in this so-called **flex/rigid** situation is to substitute a term for X that will enable decomposition to be applicable afterwards. It turns out that for finding all $\mathcal{C}$-unifiers it is sufficient to bind X to terms of the same type as X (otherwise the unifier would be ill-typed) and compatible color (otherwise the unifier would not be a $\mathcal{C}$-substitution) that either

- have the same head as the right hand side; the so-called **imitation** solution ($\lambda Z.a$ in our example) or
- where the head is a bound variable that enables the head of one of the arguments of X to become head; the so-called **projection** binding $(\lambda Z.Z)$.

In order to get a better understanding of the situation let us reconsider our example using colors. $X_{\mathsf{d}}(a_{\mathsf{c}}) = a_{\mathsf{d}}$. For the imitation solution $(\lambda Z.a_{\mathsf{d}})$ we "imitate" the right hand side, so the color on a must be d. For the projection solution we instantiate $(\lambda Z.Z)$ for X and obtain $(\lambda Z.Z)a_{\mathsf{c}}$, which β-reduces to a_{c}. We see that this "lifts" the constant a_{c} from the argument position to the top. Incidentally, the projection is only a $\mathcal{C}$-unifier of our colored example, if c and d are identical.

Fortunately, the choice of instantiations can be further restricted to the most general terms in the categories above. If X_{c} has type $\overline{\beta_n} \to \alpha$ and h_{d} has type then these so-called **general bindings** (4.11) have the following form:

$$\mathcal{G}^h_{\mathsf{d}} = \lambda Z^{\alpha_1} \ldots Z^{\alpha_n}.@(H^1_{\mathsf{e}_1}\overline{Z})\ldots(H^m_{\mathsf{e}_m}\overline{Z})$$

where

- $@ = Z^{\alpha_j}$ and $\alpha_j = \overline{\gamma_m} \to \alpha$ for some $j \leq n$, or $@ = \mathsf{S}_{\mathsf{d}}$ for some constant or free variable S of type $\overline{\gamma_m} \to \alpha$,
- the H^i are new variables of type $\overline{\beta_n} \to \gamma_i$,
- the e_i are either distinct color variables (if $\mathsf{c} \in \mathcal{X}$) or $\mathsf{e}_i = \mathsf{d} = \mathsf{c}$ (if $\mathsf{c} \in \mathcal{C}$).

If @ is one of the bound variables Z^{α_i}, then $\mathcal{G}^h_{\mathsf{d}}$ is called a **projection binding**, and else, (@ is a colored constant or free variable), an **imitation binding**. Note that while imitation bindings are unique up to the names of the new free variables H^i, there can be up to n projection bindings, depending on the types involved.

The general rule for flex/rigid equations (4.14) transforms equations of the form

$$\mathcal{E} \wedge X_{\mathsf{c}}(s^1,\ldots,s^n) =^t h_{\mathsf{d}}(t^1,\ldots,t^m)$$

into

$$\mathcal{E} \wedge X_{\mathsf{c}}(s^1,\ldots,s^n) =^t h_{\mathsf{d}}(t^1,\ldots,t^m) \wedge X_{\mathsf{c}} = \mathcal{G}^h_{\mathsf{c}}$$

which in essence only fixes a particular binding for the head variable X_{c}. It turns out (cf 4.2) that these general bindings suffice to solve all flex/rigid situations, possibly at the cost of creating new flex/rigid situations after elimination of the variable X_{c} and decomposition of the changed equations (the elimination of X changes $X_{\mathsf{c}}(s^1,\ldots,s^n)$ to $\mathcal{G}^h_{\mathsf{c}}(s^1,\ldots,s^n)$ which has head h).

Finally the only remaining case, where the heads of both sides of the equation are free variables the so-called **flex/flex** case. The solution of this case is either to project as in the flex/rigid case or to "guess" (computationally: to search for) the right head for the equation and bind the head variable to the appropriate imitation binding. Clearly this need for guessing the right head leads to a serious explosion of the search space, which makes higher-order colored unification computationally infeasible. Fortunately, most applications do not need full higher-order unification:

- For theorem proving purposes it is only important to know about the existence of any unifier. In the case of classical higher-order unification it is therefore sufficient to consider flex/flex pairs as solved, since they are guaranteed to have unifiers (cf. section 4.3). In the colored case, this is no longer the case, i.e. there are flex/flex unification problems that do not have unifiers. We identify a necessary and sufficient condition (the absence of so-called flexible chains (4.25)) and specialise the unification algorithm accordingly.

- The linguistic applications sketched in section 2.3 formulae belong to very restricted syntactic subclasses, for which much better results are known (for classical higher-order unification). In particular, the fact that free variables only occur on the left hand side of our equations reduces the problem of finding solutions to higher-order matching, of which decidability has been proven for the subclass of third-order formulae [Dow92]. This class, (intuitively allowing only nesting functions as arguments up to depth three) covers all examples studied so far.

To fortify our intuition on calculating higher-order colored unifiers let us consider the following example. Let G be a function variable of type $\iota \to \iota$, i an individual constant and f a binary function constant, then we compute the unifiers of the equation

$$G_{\mathtt{b}}(i_{\mathtt{a}}) = f_{\mathtt{b}}(i_{\mathtt{a}}, i_{\mathtt{A}})$$

Since G is a variable, we are in a flex/rigid situation and have the possibilities of projection and imitation. The projection binding for $G_{\mathtt{b}}$ would be $\lambda X.X$, since it has to be of type $\iota \to \iota$; which would lead us to the equation $i_{\mathtt{a}} = f_{\mathtt{b}}(i_{\mathtt{a}}, i_{\mathtt{A}})$, which is obviously unsolvable, since the head constants $i_{\mathtt{a}}$ and $f_{\mathtt{b}}$ clash. So we can only bind $G_{\mathtt{b}}$ to the imitation binding $(\lambda X.f_{\mathtt{b}}(H^1_{\mathtt{b}}X)(H^2_{\mathtt{b}}X))$, where the new variables H^i have type $\iota \to \iota$. This leaves us with the set

$$G_{\mathtt{b}}(i_{\mathtt{a}}) = f_{\mathtt{b}}(i_{\mathtt{a}}, i_{\mathtt{A}}) \wedge G_{\mathtt{b}} = (\lambda X.f_{\mathtt{b}}(H^1_{\mathtt{b}}X)(H^2_{\mathtt{b}}X))$$

from which we can directly eliminate the variable $G_{\mathtt{b}}$, since there are no other variants. The resulting equation

$$f_{\mathtt{b}}(H^1_{\mathtt{b}}(i_{\mathtt{a}}), H^2_{\mathtt{b}}(i_{\mathtt{a}})) = f_{\mathtt{b}}(i_{\mathtt{a}}, i_{\mathtt{A}})$$

can be decomposed to

$$(H^1_{\mathtt{b}} i_{\mathtt{a}}) = i_{\mathtt{a}} \wedge (H^2_{\mathtt{b}} i_{\mathtt{a}}) = i_{\mathtt{A}}$$

Let us first look at the first equation; in this flex/rigid situation, only the projection binding $(\lambda Z.Z)$ can be applied, since the imitation binding $(\lambda Z.i_{\mathtt{a}})$ is not b-monochrome. Thus we are left with the second equation, since $(\lambda Z.Z)i_{\mathtt{a}}$ β-reduces to $i_{\mathtt{a}}$, giving the trivial equation $i_{\mathtt{a}} = i_{\mathtt{a}}$ which can be deleted by the decomposition rules. In the remaining equation

$$(H^2_{\mathtt{b}} i_{\mathtt{a}}) = i_{\mathtt{A}}$$

both imitation and projection bindings yield a legal solution:

- The imitation binding for $H^2_{\mathtt{b}}$ is $(\lambda Z.i_{\mathtt{b}})$, and not $(\lambda Z.i_{\mathtt{A}})$, as one is tempted to believe, since it has to be b-monochrome. Thus we are left with $i_{\mathtt{b}} = i_{\mathtt{A}}$, which can (uniquely) be solved by the color substitution $[\mathtt{b}/\mathtt{A}]$.

- If we bind $H_{\mathtt{b}}^2$ to $(\lambda Z.Z)$, then we are left with $i_{\mathtt{a}} = i_{\mathtt{A}}$, which can (uniquely) be solved by the color substitution $[\mathtt{a}/\mathtt{A}]$.

If we collect the instantiations, we arrive at two possible solutions for $G_{\mathtt{b}}$ in the original equations:

$$\begin{aligned} G_{\mathtt{b}} &= (\lambda X.f_{\mathtt{b}}(X, i_{\mathtt{b}})) \\ G_{\mathtt{b}} &= (\lambda X.f_{\mathtt{b}}(X, X)) \end{aligned}$$

Obviously both of them solve the equation and furthermore, they are both most general solutions, since $i_{\mathtt{b}}$ cannot be inserted for the variable X in the second unifier (which would make it more general than the first), since X is bound.

2 Applications

2.1 Inductive Proofs

Rippling was developed for proving theorems by induction and has been applied to a large number of practical examples from this domain [Bun88, BSvH+93, Hut91]. It is based on an observation that one can iteratively unfold recursive functions in the induction conclusion, preserving the structure of the induction hypothesis while unfolding. We use colors in order to indicate the structure of the hypothesis within the conclusion. Symbols belonging to this joined structure are annotated with the color "white" while differences between both formulas are colored "grey". Also left- and right-hand sides of given equations are difference unified: the common structure of both terms of a given equation is annotated by color variables while differences are colored grey. Rippling then applies just these annotated equations which move the difference out of the way leaving behind the skeleton. In their simplest form, these equations to be used are of the form $\alpha(\beta(\gamma)) = \beta(\alpha(\gamma))$.[2] By design, the skeleton $\alpha(\gamma)$ remains unaltered by their application. If rippling succeeds then the induction conclusion $P(s(n))$ is rewritten using wave-rules into some function of the induction hypothesis, $P(n)$; that is, into $f(P(n))$ (f may be the identity). At this point we can call upon the induction hypothesis to simplify the result.

To illustrate rippling and motivate our work on colored higher order unification let us consider the following simple theorem that can be proven by inductive theorem provers using rippling/colouring techniques.

$$\sum_{i=1}^{n} f(i) + \sum_{i=1}^{n} g(i) = \sum_{i=1}^{n} [f+g](i)$$

where f, g are functions from natural numbers to naturals and we have overloaded the function $+$ also to act on such functions. This example illustrates the properties of rippling and introduces also some higher-order colored unification problems.

We formalise summation by a binary function *sum* that takes a function (that is summed over) and a upper bound as arguments. Furthermore, we will the following definition of *sum* (let f, g, H be of type $nat \to nat$ and N, n be of type nat)[3]:

$$\begin{aligned} \forall H : sum(H, 0) &= 0 && (1)\\ \forall H, N : sum(H, s(N)) &= sum(H, N) + H(s(N)) && (2) \end{aligned}$$

Then our theorem takes the form

$$\forall f, g, n : sum(f, n) + sum(g, n) = sum(\lambda x. f(x) + g(x), n)$$

In order to prove this, simple heuristics employed by most inductive provers suggest induction on n which results in the following step case[4]:

$$\begin{aligned} &sum(f, n) + sum(g, n) = sum(\lambda x. f(x) + g(x), n)\\ &\to sum(f, s(n)) + sum(g, s(n)) = sum(\lambda x. f(x) + g(x), s(n)) \end{aligned}$$

To simplify the step case using rippling, the differences between the induction conclusion and the induction hypothesis are shaded as follows:

$$\begin{aligned} &sum(f, n) + sum(g, n) = sum(\lambda x. f(x) + g(x), n) && (3)\\ &\to sum(f, s(n)) + sum(g, s(n)) = sum(\lambda x. f(x) + g(x), s(n)) \end{aligned}$$

[2]For sake of simplicity we use a shading for symbols which are annotated by the color grey while non-shaded areas are annotated either by white or color variables. But in case the distinction between color variables and white is necessary we shall annotate the colors explicitly.

[3]We employ the Prolog convention of using capital letters to indicate metavariables.

[4]The proof of the base case can be directly obtained by applying (1), so it is omitted here.

If we can move the shaded areas, so-called contexts or wave-fronts, out of the way, then we will be able to simplify the induction conclusion by appealing to the induction hypothesis.

Rippling moves wave-fronts using annotated equations based on axioms, recursive definitions and previously proven lemmas that preserve the skeleton of the term being rewritten. Corresponding to the recursive definitions for *sum* we have the following annotated equation of (2)

$$sum(H, s(N)) = sum(H, N) + H(s(N)) \tag{4}$$

When rippling, the annotations on the left-hand side of the wave-rule must match those in the term being rewritten. As a consequence, there is very little search during rewriting. In order to simplify the conclusion of (3) by rippling we apply (4) on both sides yielding the modified conclusion:

$$\begin{aligned}&(sum(f,n) + f(s(n))) + (sum(g,n) + g(s(n)))\\ &= sum(\lambda x.f(x) + g(x), n) + (f(s(n)) + g(s(n)))\end{aligned}$$

Applying associativity and commutativity law of + results in

$$\begin{aligned}&((sum(f,n) + sum(g,n)) + f(s(n)) + g(s(n)))\\ &= (sum(\lambda x.f(x) + g(x), n) + (f(s(n)) + g(s(n))))\end{aligned}$$

which allows weak fertilisation[5] on either side which completes the proof.

2.2 Lemma Speculation

The rippling process — as illustrated in the example above — relies on the existence of appropriate annotated equations in order to ripple out (or ripple inside) the occurring wave-fronts. In cases appropriate equations are missing, Ireland & Bundy [IB96] presented a technique to speculate Lemmata which push the rippling process further and which are treated as subtasks to be proven separately. Their approach is based on some kind of higher order rippling.

In order to illustrate this application of our calculus, consider the following example involving list manipulations

$$\forall x, y : list \quad rev(app(rev(x), y)) = app(rev(y), x)$$

Here *rev* and *app* stand for the operations of reversing and concatenating lists. Using induction on x we obtain the following formula as an induction conclusion:

$$rev(app(rev(cons(h, x)), y)) = app(rev(y), cons(h, x))$$

The rippling process gets blocked[6] after unfolding the definition of *rev* on the left-hand side:

$$rev(app(app(rev(x), cons(h, nil)), y)) = app(rev(y), cons(h, x))$$

In order to push the rippling process further, Ireland & Bundy speculate appropriate Lemmata which are then considered as subtasks of the proof. In this example they calculate a schematic form of an appropriate annotated equation

$$app(X, cons(Y, Z)) = app(F_1(X, Y, Z), Z) \tag{5}$$

[5]This standard technique from inductive theorem proving allows to use the inductive hypothesis to rewrite the inductive conclusion

[6]There are no applicable annotated equations in the data base.

which can be used to move the blocked wave-front on the right-hand side towards the sink y. While the left-hand side of the speculated lemma is just a generalization of the subterm to be modified, the higher-order variable F_1 represents the unknown wave-front on the right-hand side which has still to be constrained by the further rippling process. Applying this equation on the right-hand side yields:

$$rev(app(app(rev(x), cons(h, nil)), y)) = app(F_1(rev(y), h, x), x) \tag{6}$$

In order to enable the use of the induction hypothesis in this example the context has to be moved in front of the universally quantified variable y which operates as a so-called *sink*. Thus, we use the annotated equation

$$app(rev(Y), cons(X, nil)) = rev(cons(X, Y)) \tag{7}$$

in order to ripple the wave-front on the right-hand side towards y. In order for (7) to be applicable to (6), we must unify[7] $F_1(rev(y), h, x)$ and $app(rev(Y), cons(X, nil))$. Higher-order colored unification (see example 4.15 for a trace of the computation) results in a solution $[\lambda U, V, W.app(U, cons(V, nil))/F_1, y/Y]$ Applying the instance of (7) to the right-hand side of (6) the wave-front is moved towards the sink y:

$$rev(app(app(rev(x), cons(h, nil)), y)) = app(rev(cons(h, y)), x) \tag{8}$$

The unifier used to perform this step now also refines the scheme of the speculated annotated equation (5) we used previously to unblock the rippling process, to

$$app(X, cons(Y, Z)) = app(app(X, cons(Y, nil)), Z) \tag{9}$$

Using this speculated equation (9) also on the left-hand side finally yields:

$$rev(app(rev(x), cons(h, y))) = app(rev(cons(h, y)), x)$$

which enables the use of the induction hypothesis and completes this particular proof. Proving also the speculated lemma (9) by induction finishes the overall proof.

2.3 Higher–Order Unification and NL semantics

In this section we will present a different kind of application of higher-order colored unification. In [GK96] the colored lambda calculus is used as a tool to specify the interface between the classical semantic construction process (using higher-order unification) and other sources of linguistic information (which are coded into color information). We will now briefly sketch the underlying ideas for the case of verb-phrase ellipsis (the phenomenon that parts of natural language sentences can be replaced by utterances like "does too"), for a thorough treatment of cases like focus constructions, second-occurrence expressions, and adverbial quantification, see [GK96].

The basic idea [DSP91] underlying the use of higher-order unification for natural language semantics is very simple: Following [Mon74], the typed λ–calculus is used as a semantic representation language while semantically underspecified elements (e.g. anaphoric references or ellipses) are represented by free variables whose value

[7]To ease readability we have slightly simplified the method of [IB96] since in practice the overall method is a bit more elaborate: In order to allow the speculation of more complex wave-fronts the occurrence of the meta-variable $F_1(rev(y), h, x)$ is replaced by a nested term $F_2(F_1(rev(y), h, x), h, x)$. Thus, in general F_2 allows one to create additional wave-fronts in the later rippling process but in this example it is of no use and will only be instantiated to the projection $\lambda X, Y, Z.X$.

is determined by solving higher-order equations. For instance, the discourse *Dan likes golf. Peter does too.* has the semantic representation

$$like(dan, golf) \wedge R(peter)$$

where the value of the predicate variable R is determined by equation

$$like(dan, golf) = R(dan)$$

however, only the first of the solutions

$$\sigma_1 = [\lambda X.like(X, golf)/R] \quad \text{and} \quad \sigma_2 = [\lambda X.like(dan, golf)]$$

leads to the linguistically desired solution $like(dan, golf) \wedge like(peter, golf)$, while the other one leads to $like(dan, golf) \wedge like(dan, golf)$, which is clearly not the desired reading of the discourse.

To remedy this shortcoming, Dalrymple, Shieber and Pereira, who have pioneered this analysis in [DSP91] propose an informal restriction, the **primary occurrence restriction**, which from the set of linguistically valid solutions, deletes any solution which contains a pre-determined so-called *primary occurrence* (in our case *dan*).

In the colored λ-calculus, the primary occurrence restriction can directly be modelled as follows: Primary occurrences are p-colored while free variables are s-colored (all other non-bound symbols are colored by distinct color variables, which we will not show in our examples). Given the restriction for $\mathcal{C}$-substitutions, such a coloring ensures that any solution containing a primary occurrence is ruled out. Hence no substitution will ever contain a primary occurrence (i.e. a p-colored symbol) as was required by the primary occurrence restriction. For instance, the colored representation of our little discourse above is $like(dan_{\mathrm{p}}, golf) \wedge R_{\mathrm{s}}(peter)$ together with the colored unification problem $like(dan_{\mathrm{p}}, golf) = R_{\mathrm{s}}(dan_{\mathrm{p}})$, which only has the $\mathcal{C}$-unifier $[\lambda X.like(x, golf)/R_{\mathrm{s}}]$. The color erasure of the equation above has another unifier $[\lambda X.like(dan_{\mathrm{p}}, golf)/R_{\mathrm{s}}]$ which is not well-colored.

Even though we have only sketched the relevant ideas, should be clear, that higher-order colored unification provides a general framework for specifying the linguistic information for the construction process that avoids over-generalization (i.e. the construction of linguistically undesired readings of discourses).

3 Colored λ-Calculus

In this section we extend the simply typed λ-Calculus with a concept of color annotations for constant and variable occurrences.

Definition 3.1 (Types) Let $\mathcal{BT}$ be a set of **base types**, then the set $\mathcal{T}$ of **types** is inductively defined to be the set $\mathcal{BT}$ together with all expressions $\alpha \to \beta$, where α and β are types. The **functional** type $\alpha \to \beta$ denotes the type of functions with domain α and codomain β.

We use the convention of association to the right for omitting parentheses in functional types, thus $\alpha \to \beta \to \gamma$ is an abbreviation for $(\alpha \to (\beta \to \gamma))$. This way the type $\gamma := \beta_1 \to \ldots \to \beta_n \to \alpha$ denotes the type of n-ary functions, that take n arguments of the types $\beta_1, \ldots, \beta_n$ and have values of type α. To conserve even more space we use a kind of vector notation and abbreviate γ by $\overline{\beta_n} \to \alpha$.

Definition 3.2 (Typed Collection) A collection $\mathcal{D} := \mathcal{D}_\mathcal{T} := \{\mathcal{D}_\alpha | \alpha \in \mathcal{T}\}$ of sets $\mathcal{D}_\alpha$, indexed by the set $\mathcal{T}$ of types, is called a **typed collection (of sets)**. A typed collection $\mathcal{I} := \{\mathcal{I}^\alpha \in \mathcal{F}_p(\mathcal{D}_\alpha; \mathcal{E}_\alpha) | \alpha \in \mathcal{T}\}$ of functions is called a **typed function** $\mathcal{I}: \mathcal{D}_\mathcal{T} \to \mathcal{E}_\mathcal{T}$.

We will write finite functions like substitutions or variable assignments as sets of pairs $\varphi := [a^1/X^1], \ldots, [a^n/X^n]$ with the intended meaning that $\varphi(X^i) = a^i$. Furthermore we use the convention that $\psi := \varphi, [a/X]$ assigns a to X and coincides with φ everywhere else. We say that (partial) functions Γ and Δ **agree** $(\Gamma || \Delta)$, iff for each $X \in \mathbf{Dom}(\Gamma) \cap \mathbf{Dom}(\Delta)$ we have $\Gamma(X) = \Delta(X)$. In this case the set-theoretic union $\Gamma \cup \Delta$ (taking partial functions to be right-unique relations) is again a partial function.

Definition 3.3 (Basic Material) For the definition of well-formed formulae we fix a **signature**, i.e. a typed collection $\Sigma_\mathcal{T}$ of symbols and countably infinite (untyped) sets $\mathcal{V}$ of **variables**, $\mathcal{C}$ of **colors** and $\mathcal{X}$ of **color variables**.

Definition 3.4 (Colored Symbols) Atomic occurrences in well-formed formulae have a color annotation, therefore we fix the notation $\Sigma_\mathcal{Z}$ for the set $\{c_\mathbf{a} | c \in \Sigma, \mathbf{a} \in \mathcal{Z}\}$ for some subset $\mathcal{Z} \subset \mathcal{C} \cup \mathcal{X}$, and analogously for $\mathcal{V}_\mathcal{Z}$.

While the **constants** in Σ are a priori typed by definition ($\tau(c) = \alpha$, iff $c \in \Sigma_\alpha$) the variables are more volatile objects and obtain their type by a **variable context** Γ.

Definition 3.5 (Variable Context) A **variable context** $\Gamma_\mathcal{Z}$ is a pair consisting of a partial function Γ, that assigns types to variables and a set $\mathcal{Z} \subseteq \mathcal{C} \cup \mathcal{X}$ of colours. We will use $\Gamma_\mathcal{Z}$ as a partial function assigns types to coloured variables in $\mathbf{Dom}(\Gamma_\mathcal{Z}) := \mathbf{Dom}(\Gamma)_\mathcal{Z}$.

We will say $\Gamma_\mathcal{Z} || \Delta_\mathcal{W}$, iff $\Gamma || \Delta$ and use $\Gamma_\mathcal{Z} \cup \Delta_\mathcal{W}$ as an abbreviation for $(\Gamma \cup \Delta)_{\mathcal{Z} \cup \mathcal{W}}$. If $\mathcal{Z}$ is clear from the context or irrelevant, then we will often drop the reference and only write Γ for $\Gamma_\mathcal{Z}$.

Definition 3.6 (Well-Formed Formulae) Let $\Gamma_\mathcal{Z}$ be a variable context, then for each $\alpha \in \mathcal{T}$ we inductively define the set $\mathit{wff}_\alpha(\Sigma; \Gamma_\mathcal{Z})$ of **well-formed formulae of type** α. For this we also need a function $\mathcal{C}_\in$, where $\mathcal{C}_\in(\mathbf{S}, \mathbf{B})$ is the set of the color annotations of all occurrences of the symbol $\mathbf{S}$ in the formula $\mathbf{B}$.

1. If $\mathbf{S}_\mathbf{a} \in \Sigma^\alpha_\mathcal{Z} \cup \mathcal{V}^\alpha_\mathcal{Z}$, then $\mathbf{S}_\mathbf{a} \in \mathit{wff}_\alpha(\Sigma; \Gamma_\mathcal{Z})$ and $\mathcal{C}_\in(\mathbf{S}_\mathbf{a}, \mathbf{S}_\mathbf{a}) = \{\mathbf{a}\}$
2. If $\mathbf{A} \in \mathit{wff}_{\beta \to \alpha}(\Sigma; \Gamma_\mathcal{Z})$ and $\mathbf{B} \in \mathit{wff}_\beta(\Sigma; \Delta, \mathcal{W})$, and furthermore $\Gamma_\mathcal{Z} || \Delta_\mathcal{W}$, then $\mathbf{AB} \in \mathit{wff}_\alpha(\Sigma; \Gamma_\mathcal{Z} \cup \Delta_\mathcal{W})$ and $\mathcal{C}_\in(\mathbf{S}, \mathbf{AB}) = \mathcal{C}_\in(\mathbf{S}, \mathbf{A}) \cup \mathcal{C}_\in(\mathbf{S}, \mathbf{B})$.

3. If $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma; \Gamma, [X:\beta]_\mathcal{Z})$, and $\mathcal{C}_\in(X, \mathbf{A}) = \emptyset$ then $(\lambda X_\beta . \mathbf{A}) \in \mathit{wff}_{\beta\to\alpha}(\Sigma; \Gamma_\mathcal{Z})$ and $\mathcal{C}_\in(\mathbf{S}, \lambda X.\mathbf{A}) = \mathcal{C}_\in(\mathbf{S}, \mathbf{A})$.

We call formulae of the form **AB applications**, and formulae of the form $\lambda X_\alpha.\mathbf{A}$ **abstractions.**

Note that this definition gives a dynamic account of variable typing, only requiring to specify the free variables of a formula in the context. For instance the variable X is discharged from the context Γ, when it is bound by the abstraction.

We adopt the usual definition of **free** and **bound** (all occurrences of the variable X in $\lambda X_\alpha.\mathbf{A}$ are called bound), variables and call a formula **closed,** iff it does not contain free variables. We will write **free(A)** for the set of (color and term) variables in **A**. As in first-order logic the names of bound variables have no meaning at all, thus we consider alphabetic variants as identical and use a notion of substitution that systematically renames bound variables in order to avoid variable capture.

Remark 3.7 Note that for a well-formed formula $\mathbf{A} \in \mathit{wff}(\Sigma; \Gamma_\mathcal{Z})$, we can restrict $\Gamma_\mathcal{Z}$, such that $\mathbf{Dom}(\Gamma)$ is just the set of variables and $\mathcal{Z}$ that of colors occurring in **A**. Furthermore, since there can only be finitely many occurrences of variables and constants in a well-formed formula, we can restrict ourselves to the case, where Γ and $\mathcal{Z}$, and consequently $\mathbf{Dom}(\Gamma_\mathcal{Z})$ are finite.

Definition 3.8 We will call a formula $\mathbf{A} \in \mathit{wff}(\Sigma; \Gamma_\mathcal{Z})$

- **flexible,** iff $\mathcal{C}_\in(\mathbf{S}, \mathbf{A}) \subseteq \mathcal{X}$ for all symbols $\mathbf{S} \in \mathbf{Dom}(\Gamma) \cup \Sigma$
- **rigid,** iff $\mathcal{C}_\in(\mathbf{S}, \mathbf{A}) \subseteq \mathcal{C}$ for all $\mathbf{S} \in \mathbf{Dom}(\Gamma) \cup \Sigma$
- **a-monochrome,** iff there is a single colour constant a, such that $\mathcal{C}_\in(\mathbf{S}, \mathbf{A}) = \{\mathrm{a}\}$ for all $\mathbf{S} \in \mathbf{Dom}(\Gamma) \cup \Sigma$.
- **flexichrome,** iff **A** is flexible and any color variable occurs at most once in **A**.

Finally, we call a formula **A compatible** with a colour a, iff $\mathrm{a} \in \mathcal{C}$ implies that **A** is a-monochrome.

Example 3.9 $F_\mathrm{A}(s_\mathrm{d}(a_\mathrm{c}))$, $\lambda X.s_\mathrm{d}(X)$ are examples of colored λ-terms while $\lambda X.s_\mathrm{d}(X_\mathrm{B})$ is not a colored λ-term (bound variables may not have colours).

We denote the constants by lower case letters and the variables by upper case letters and use bold upper case letters $\mathbf{A}_\alpha$, $\mathbf{B}_{\alpha\to\beta}$, $\mathbf{C}_\gamma \ldots$ as syntactical variables for well-formed formulae. In order to make the notation of well-formed formulae more legible, we use the convention that the group brackets (and) associate to the left and that the square dot . denotes a left bracket, whose mate is as far right as consistent with the brackets already present. Additionally, we combine successive λ-abstractions, so that the well-formed formulae $(\lambda X^1.\lambda X^2.\ldots\lambda X^n.\mathbf{A}\mathbf{E}^1 \ldots \mathbf{E}^m)$ and $\lambda X^1 \ldots X^n.\mathbf{A}\mathbf{E}^1 \ldots \mathbf{E}^m$ and $\lambda\overline{X^n}.\mathbf{A}\overline{\mathbf{E}^m}$ stand for $(\lambda X^1(\lambda X^2 \ldots (\lambda X^n(\mathbf{A}\mathbf{E}^1)\mathbf{E}^2 \ldots \mathbf{E}^m)\cdots)$. Finally we will abbreviate a formula $\Pi^\alpha(\lambda X_\alpha.\mathbf{A})$ with $\forall X_\alpha.\mathbf{A}$ in order to re-obtain a more traditional appearance of quantified formulae.

Definition 3.10 (λ-Reduction) Let $\lambda \in \{\beta, \beta\eta, \eta\}$. We say that a well-formed formula **B** is obtained from a well-formed formula **A** by a **one-step λ-reduction** ($\mathbf{A} \longrightarrow_\lambda \mathbf{B}$), if it is obtained by applying one of the following rules to a well-formed part (which we call a **λ-redex**) of **A**.

β-Reduction $(\lambda X.\mathbf{C})\mathbf{D} \longrightarrow_\beta [\mathbf{D}/X]\mathbf{C}$.

η-Reduction If X is not free in $\mathbf{C}$, then $(\lambda X.\mathbf{C}X) \longrightarrow_\eta \mathbf{C}$.

As usual we denote the transitive closure of a reduction relation $\longrightarrow_\lambda$ with $\longrightarrow^*_\lambda$. These rules induce equivalence relations $=_\beta, =_\eta$, and $=_{\beta\eta}$ on $\mathit{wff}(\Sigma; \Gamma_\mathcal{Z})$, which we call the **λ-equality** relations. A formula that does not contain a λ-redex, and thus cannot be reduced by λ-reduction, is called a **λ-normal form.**

Remark 3.11 Clearly the colors annotating the atoms do not affect $\beta\eta$-convertibility, since bound variables are not colored. Therefore, the β-, η-, and $\beta\eta$-reduction relations are terminating and confluent, as in the simply typed case. Thus for any formula $\mathbf{A}$ there is a sequence of β-reductions $\mathbf{A} \longrightarrow^*_\beta \mathbf{B}$ such that $\mathbf{B}$ is a **β-normal form.**

To make arguments like the above more formal we define the erasure of a colored formula, this is a simply typed λ-term, which we obtain erasing all color-information:

Definition 3.12 (Erasure) The **erasure** of colored λ-terms to simply typed λ-terms is defined by:

- $|\mathbf{A}_\mathsf{a}| = \mathbf{A}$ if $\mathbf{A} \in \Sigma \cup \mathcal{V}$ and $\mathsf{a} \in \mathcal{C} \cup \mathcal{X}$
- $|(\mathbf{AB})| = (|\mathbf{A}||\mathbf{B}|)$
- $|(\lambda X_\alpha.\mathbf{A})| = \lambda X_\alpha.|\mathbf{A}|$

We call any colored formulae $\mathbf{A}$ and $\mathbf{B}$ **variants,** if $|\mathbf{A}| = |\mathbf{B}|$

We now have the tools for defining $\mathcal{C}$-substitutions, a restriction of well-typed substitutions that preserves syntactic color information, such as the skeleton (cf. 5).

Definition 3.13 ($\mathcal{C}$-Substitution) Let $\Gamma_\mathcal{Z}$, and $\Delta_\mathcal{W}$ be variable contexts, then a **$\mathcal{C}$-substitution** σ is a pair (σ^t, σ^r), where σ^t is a typed mapping $\mathcal{V}_\mathcal{W} \to \mathit{wff}(\Sigma; \Delta_\mathcal{W})$ and $\sigma^r : \mathcal{Z} \to \mathcal{W}$ such that

- If $\mathsf{a} \in \mathcal{C}$, then $\sigma(X_\mathsf{a})$ must be a-monochrome for any variable $X_\mathsf{a} \in \mathbf{Dom}(\Gamma_\mathcal{Z})$, i.e. $\sigma(X_\mathsf{a})$ must be compatible with a.
- $|\sigma(X_\mathsf{a})| = |\sigma(X_\mathsf{b})|$ for all $X_\mathsf{a}, X_\mathsf{b} \in \mathbf{Dom}(\Sigma)$.

A $\mathcal{C}$-substitution σ with domain $\Gamma_\mathcal{Z}$ and codomain $\Delta_\mathcal{W}$ can always be extended to a homomorphism $\sigma : \mathit{wff}(\Sigma; \Gamma_\mathcal{Z}) \to \mathit{wff}(\Sigma; \Delta_\mathcal{W})$ by the standard construction. Note that $\mathcal{C}$-substitutions always have a finite **domain** $\mathbf{Dom}(\sigma) := \{X_\mathsf{a} \in \mathbf{Dom}(\Gamma_\mathcal{Z}) | \sigma(X_\mathsf{a}) \neq X_\mathsf{a}\}$, since we have restricted ourselves to finite variable contexts. We will denote the set of all substitutions with domain $\Gamma_\mathcal{Z}$ (and codomain $\Delta_\mathcal{W}$) with $\mathbf{SUB}(\Sigma; \Gamma_\mathcal{Z} \to \Delta_\mathcal{W})$ $(\mathbf{SUB}(\Sigma; \Gamma_\mathcal{Z}))$.

Remark 3.14 Note that the second condition in the definition of $\mathcal{C}$-substitutions (instances of variants are variants) holds in general as a simple induction on the structure shows: If σ is a $\mathcal{C}$-substitution and $\mathbf{A}$ and $\mathbf{B}$ are variants, then $\sigma(\mathbf{A})$ and $\sigma(\mathbf{B})$ are variants.

4 Unification

The central data structure higher-order unification is that of unification problems, i.e. sets of pairs $\mathbf{A}^1 = \mathbf{B}^i$ of formulae together with two variable contexts $\Gamma_{\mathcal{Z}}$ and $\Upsilon_{\emptyset}$.

Definition 4.1 (Unification Problem) A **unification problem** is a formula of the form $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathbf{P}_1 \wedge \ldots \mathbf{P}_n$, where $\Gamma_{\mathcal{Z}}, \Upsilon_{\emptyset}$ are variable contexts and the $\mathbf{P}_1$ are (term or colour) **pairs**. Term pairs are of the form $\mathbf{A} =^t \mathbf{B}$, where $\mathbf{A}, \mathbf{B} \in wff_{\alpha}(\Sigma; \Gamma_{\mathcal{Z}} \cup \Upsilon_{\emptyset})$ for some type α, whereas color pairs are of the form $\mathtt{a} =^c \mathtt{b}$ with $\mathtt{a}, \mathtt{b} \in \mathcal{Z}$. If we do not want to specify whether a pair is a term- or colour pair, we use $\mathbf{A} \doteq \mathbf{B}$.

We have chosen this logical form, since the existence of $\mathcal{C}$-unifiers for $\mathcal{E}$ corresponds to validity of the formula $\mathcal{E}$. The existential context $\Gamma_{\mathcal{Z}}$ collects type and colour information of the free variables for which the problem is to be solved. The universal context $\Upsilon_{\emptyset}$ is needed for the specific set of transformations presented in this paper (cf. 4.9 which decompose abstractions and accumulate bound variables (which are uncoloured) in $\Upsilon_{\emptyset}$).

Definition 4.2 (Unifier) We call a $\mathcal{C}$-substitution $\theta \in \mathbf{SUB}(\Sigma; \Gamma_{\mathcal{Z}})$ with $\theta(\mathbf{A}^i) =_{\beta\eta} \theta(\mathbf{B}^i)$ for all $1 \leq i \leq n$ **$\mathcal{C}$-unifier** for $\mathcal{E}$ and we will denote the set of $\mathcal{C}$-unifiers of $\mathcal{E}$ with $\mathbf{U}(\mathcal{E})$.

We call a subset $\Psi \subseteq \mathbf{U}(\mathcal{E})$ **a complete set of $\mathcal{C}$-unifiers of $\mathcal{E}$**, iff for all $\theta \in \mathbf{U}(\mathcal{E})$ there is a $\sigma \in \Psi$ that is more general than θ, i.e. there is a $\mathcal{C}$-substitution ρ, such that $\sigma(X) =_{\beta\eta} \rho(\theta(X))$ 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 $\mathcal{E}$.

Definition 4.3 (Solved Form) Let $\mathcal{E} := \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathbf{A} \doteq \mathbf{B} \wedge \mathcal{E}'$ be a unification problem, then we call the pair $\mathbf{A} \doteq \mathbf{B}$ **solved in $\mathcal{E}$**, iff

- it is a term pair $X_{\mathtt{a}} =^t \mathbf{B}$ for some $X_{\mathtt{a}} \in \mathbf{Dom}(\Gamma_{\mathcal{Z}})$, such that
  - $X \notin \mathbf{free}(\mathbf{B})$ and
  - if $X_{\mathtt{b}} \in \mathbf{free}(\mathcal{E}'))$ for some $\mathtt{b} \in \mathcal{Z}$, then it occurs in the left hand side of a pair $X_{\mathtt{b}} =^t \mathbf{C}$ $|\mathbf{B}| = |\mathbf{C}|$, and furthermore
  - $\mathbf{Dom}(\Upsilon) \cap \mathbf{free}(\mathbf{A}) = \emptyset$,
- it is a color pair $\mathtt{A} =^c \mathtt{b}$ for some color variable $\mathtt{A} \in \mathcal{X}$.

We call $\mathcal{E}$ in solved form, iff all atoms in $\mathbf{Dom}(\Gamma_{\mathcal{Z}})$ are solved in $\mathcal{E}$. Clearly, any $\mathcal{C}$-substitution $\sigma = [\mathbf{A}^1/h^1_{\mathtt{a}_1}], \ldots, [\mathbf{A}^n/h^n_{\mathtt{a}_n}]$ uniquely determines a solved unification problem $\mathcal{E}_{\sigma} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.X^1_{\mathtt{a}_1} =^t \mathbf{A}^1 \wedge \ldots \wedge X^n_{\mathtt{a}_n} =^t \mathbf{A}^n$ in solved form. Conversely, the conditions on solved forms ensure that the corresponding substitutions are $\mathcal{C}$-substitutions: The first condition ensures well-definedness (occurs-check) and idempotence, the second ensures that $\sigma_{\mathcal{E}}$ is a $\mathcal{C}$-substitution and the third condition forbids bound variables in solutions (which would be unsound, as already the $\exists\forall$ notation suggests).

Lemma 4.4 *Let $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}_{\sigma}$ be a $\mathcal{C}$-unification problem in solved form, then σ is a most general $\mathcal{C}$-unifier for $\mathcal{E}$. In particular, for any $\mathcal{C}$-unifier of $\mathcal{E}$ we have $\theta =_{\beta\eta} \theta \circ \sigma[\mathcal{E}]$.*

Proof: We have $\sigma(F_{\mathtt{a}}) = \sigma(\mathbf{A})$ for any pair $F_{\mathtt{a}} =^t \mathbf{A} \in \mathcal{E}_{\sigma}$, thus σ is a $\mathcal{C}$-unifier for $\mathcal{E}$. If $\theta \in \mathbf{SUB}(\Sigma; \Gamma)$ is a unifier for $\mathcal{E}$, then $\theta \circ \sigma(X^i) = \theta(\mathbf{A}^i) =_{\beta\eta} \theta(X^i)$ and $\theta(Y) = \theta \circ \sigma(Y)$ for $Y \notin \mathbf{Dom}(\Gamma_{\mathcal{Z}})$, so that indeed $\theta =_{\beta\eta} \theta \circ \sigma$. Now it only remains to verify that $\sigma_{\mathcal{E}}$ is a $\mathcal{C}$-substitution. □

Lemma 4.5 $\mathrm{U}(\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}\wedge\mathcal{E}_\sigma)=\mathrm{U}(\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\sigma(\mathcal{E})\wedge\mathcal{E}_\sigma)$

Proof: Note that $\theta\in\mathrm{U}(\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}\wedge\mathcal{E}_\sigma)$, iff $\theta\in\mathrm{U}(\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}_\sigma)\cap\mathrm{U}(\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E})$, so $\theta=\theta\circ\sigma$ by 4.4(1). Now $\theta\circ\sigma\in\mathrm{U}(\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E})$, iff $\theta\in\mathrm{U}(\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\sigma(\mathcal{E}))$, which gives the assertion. □

4.1 Simplification

A $\mathcal{C}$-substitution θ has to obey the dependencies between different variables. Instantiating $X_{\mathtt{a}}$ and $X_{\mathtt{b}}$ we have to take care that $|\theta(X_{\mathtt{a}})|=|\theta(X_{\mathtt{b}})|$. Hence, if the unification process instantiates $X_{\mathtt{a}}$ with a formula **A**, we also have to instantiate $X_{\mathtt{b}}$ with suitable formula $\mathbf{A}'$ in order to satisfy the conditions for $\mathcal{C}$-substitutions in 3.13. In particular we have to have

- $|\mathbf{A}|=|\mathbf{A}'|$ and
- $[\mathbf{A}'/X_{\mathtt{b}}]$ has must be a $\mathcal{C}$-substitution

If $\mathtt{b}\in\mathcal{C}$, then there is a unique solution for $\mathbf{A}'$ which we call a **b-monochrome variant** of **A**. Intuitively we can obtain $\mathbf{A}'$ from **A** by re-dyeing all colors and color-variables to b. In case $\mathtt{b}\in\mathcal{X}$ the color annotations in $\mathbf{A}'$ are not restricted, so we only require $|\mathbf{A}|=|\mathbf{A}'|$. Thus we need some "most general pattern" which can be instantiated to any possible $\mathbf{A}'$. We call these patterns **flexichrome variants** and obtain a flexichrome variant for **A** by replacing each color or color-variable in **A** by new color-variables.

Definition 4.6 (a-Chrome Variant) Let $\mathbf{A},\mathbf{B}\in\mathit{wff}_\alpha(\Sigma;\Gamma_{\mathcal{Z}})$, then we call **B** a

- a **flexichrome variant** of **A**, iff **B** is flexichrome and $|\mathbf{A}|=|\mathbf{B}|$,
- a **a-monochrome variant** of **A**, iff **B** is a-monochrome and $|\mathbf{A}|=|\mathbf{B}|$,
- a **b-chrome variant** of **A**, iff $\mathtt{b}\in\mathcal{X}$ and **B** is a flexichrome variant of **A** or $\mathtt{b}\in\mathcal{C}$ and **B** is a b-monochrome variant of **A**

Note that a-monochrome variants are uniquely determined, since we can obtain them by replacing each color and color-variable by a.

Example 4.7 $s_{\mathtt{A}}(X_{\mathtt{B}})$ is a flexichrome variant of $s_{\mathtt{d}}(X_{\mathtt{c}})$, and $\lambda X.s_{\mathtt{A}}(s_{\mathtt{B}}X)$ one of $\lambda X.s_{\mathtt{d}}(s_{\mathtt{c}}X)$, but $\lambda X.s_{\mathtt{A}}(s_{\mathtt{A}}X)$ and $\lambda X.s_{\mathtt{A}}(s_{\mathtt{c}}X)$ are not. Furthermore, the formulae $s_{\mathtt{c}}(X_{\mathtt{c}}),\lambda X.s_{\mathtt{c}}(s_{\mathtt{c}}X)$ are d-chrome variants of $s_{\mathtt{d}}X_{\mathtt{A}}$ respectively $\lambda X.s_{\mathtt{d}}(s_{\mathtt{c}}X)$.

Lemma 4.8 *If a formula* **A** *is compatible with some color* $\mathtt{a}\in\mathcal{C}\cup\mathcal{X}$*, then there is a* a*-chrome variant* **G** *of* **A** *and a* $\mathcal{C}$*-substitution* ρ*, such that* $\rho(\mathbf{G})=\mathbf{B}$.

Proof: If a is a color variable, then by a simple induction on the structure of **A**, we see that there is a flexichrome variant **G** of **A** and furthermore that we can chose ρ to be a color substitution that re-dyes the color variables of **G**. If $\mathtt{a}\in\mathcal{C}$, then **A** must be a-monochrome by compatibility, and we can choose $\mathbf{G}=\mathbf{A}$ and ρ to be the identity substitution. □

Definition 4.9 ($\mathcal{SIM}$: Simplification of $\mathcal{C}$-Unification Problems) The rules for constraint simplification consist of the decomposition rules

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.(\lambda X_{\alpha}.\mathbf{A}) =^t (\lambda Y_{\alpha}.\mathbf{B})}{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon,[Z\colon\alpha].[Z/X]\mathbf{A} =^t [Z/Y]\mathbf{B}}\ \mathcal{SIM}(\alpha)$$

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.(\lambda X_{\alpha}.\mathbf{A}) =^t \mathbf{B}}{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon,[Z\colon\alpha].[Z/X]\mathbf{A} =^t (\mathbf{B}Z)}\ \mathcal{SIM}(\eta)$$

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.h_{\mathsf{a}}\overline{\mathbf{U}^n} =^t h_{\mathsf{b}}\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in \Sigma \cup \mathbf{Dom}(\Upsilon)}{\mathsf{a} =^c \mathsf{b} \wedge \mathbf{U}^1 =^t \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^t \mathbf{V}^n \wedge \mathcal{E}}\ \mathcal{SIM}(dec)$$

where $Z \notin \mathbf{Dom}(\Gamma_{\mathcal{Z}}) \cup \mathbf{Dom}(\Upsilon)$ is a new variable and the following variable elimination rules:

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathtt{A} =^c \mathsf{b} \wedge \mathcal{E} \quad \mathtt{A} \in \mathcal{X}}{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathtt{A} =^c \mathsf{b} \wedge [\mathsf{b}/\mathtt{A}]\mathcal{E}}\ \mathcal{SIM}(elim{:}col)$$

where $\mathsf{a} \in \mathbf{free}(\mathcal{E})$ but $\mathtt{A} \notin \mathbf{free}(\mathbf{A})$.

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.F_{\mathsf{a}}\overline{X^k} =^t \mathbf{A} \wedge \mathcal{E} \quad X^i \in \mathbf{Dom}(\Upsilon)}{\exists\Gamma_{\mathcal{Z}'}.\forall\Upsilon_{\emptyset}.F_{\mathsf{b}} =^t \mathbf{G} \wedge F_{\mathsf{a}}\overline{X^k} =^t \mathbf{A} \wedge [\mathbf{G}/F_{\mathsf{b}}]\mathcal{E}}\ \mathcal{SIM}(elim{:}term)$$

Here we assume that $F \notin \mathbf{free}(\mathbf{A})$, but $F_{\mathsf{b}} \in \mathbf{free}(\mathcal{E})$ for some color b and that $\mathbf{G}$ is a b-chrome variant of $\lambda\overline{X^k}.\mathbf{A}$[8]. The new set $\mathcal{Z}'$ of colors is the set $\mathcal{Z}$ possibly augmented by the new color variables needed for $\mathbf{G}$, if if is flexichrome.

We apply these rules with the understanding that the operators $\wedge$ and $=^t$ are commutative and associative, that trivial pairs may be dropped and that vacuous quantifications can be eliminated from the prefix. Furthermore after the application of each rule all formulae are reduced to head normal form. Finally, no rule may be applied to a solved pair.

Lemma 4.10 *If $\mathcal{D}\colon\mathcal{E} \vdash_{\mathcal{SIM}} \mathcal{E}'$, then* $\mathbf{U}(\mathcal{E}) = \mathbf{U}(\mathcal{E}')[\mathcal{E}]$.

Proof: Clearly it suffices to show the assertion for the case, where $\mathcal{D}$ consists of a single rule application. For $\mathcal{SIM}(\alpha)$ we have the following situation:

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.(\lambda X_{\alpha}.\mathbf{A}) =^t (\lambda Y_{\alpha}.\mathbf{B}) \wedge \mathcal{F}}{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon,[Z\colon\alpha]_{\emptyset}.[Z/X]\mathbf{A} =^t [Z/Y]\mathbf{B} \wedge \mathcal{F}}\ \mathcal{SIM}(\alpha)$$

where $\Delta' := \Delta,[Z\colon\alpha]$ and $Z \notin \mathbf{Dom}(\Delta)$. For any $\mathcal{C}$-substitution θ we have $\theta(\lambda X.\mathbf{A}) = \theta(\lambda Y.\mathbf{B})$, iff $\theta(\lambda Z.[Z/X]\mathbf{A}) = \theta(\lambda Z.[Z/Y]B)$ by α-conversion and

[8]In contrast to the simple higher-order unification we restrict the variable elimination rules to cases, where it is certain that the elementary substitutions are $\mathcal{C}$-substitutions. In particular the second rule does not immediately eliminate a colored variable F_{a}, but rather eliminates all variants F_{b} of F_{a} *first*, in order to ensure that the instantiations for all variants of F_{a} are variants.

$(\lambda Z.\theta([Z/X]\mathbf{A}) = (\lambda Z.\theta([Z/Y]B)$, since we can assume that $X, Y \neq \mathbf{Dom}(\Delta)$. However the last condition is equivalent to $\theta([Z/X]\mathbf{A}) = \theta([Z/Y]B)$. Thus the sets of substitutions that solve $\mathcal{E}$ and $\mathcal{E}'$ are identical.

In the presence of $\mathcal{SIM}(\alpha)$ the rule $\mathcal{SIM}(\eta)$ is equivalent to a direct consequence of η-conversion: let $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\lambda X_{\alpha}.\mathbf{A} =^{t} \mathbf{B}$, then $\mathbf{B} =_{\eta} \lambda X_{\alpha}.\mathbf{B}X$. Since $\mathcal{C}$-unifiability is defined modulo η-equality, the set of $\mathcal{C}$-unifiers does not change by replacing $\mathbf{B}$ by its η-expansion $(\lambda X_{\alpha}.\mathbf{B}X)$. Now we obtain the assertion for $\mathcal{SIM}(\eta)$ by that for $\mathcal{SIM}(\alpha)$.

In the $\mathcal{SIM}(dec)$ case we have

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.h_{\mathrm{a}}\overline{\mathbf{U}^{n}} =^{t} h_{\mathrm{b}}\overline{\mathbf{V}^{n}} \wedge \mathcal{E}}{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathrm{a} =^{c} \mathrm{b} \wedge \mathbf{U}^{1} =^{t} \mathbf{V}^{1} \wedge \ldots \wedge \mathbf{U}^{n} =^{t} \mathbf{V}^{n} \wedge \mathcal{E}} \; \mathcal{SIM}(dec)$$

where $h \in \Sigma \cup \mathbf{Dom}(\Upsilon)$. Let $\theta \in \mathbf{U}(\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathrm{a} =^{c} \mathrm{b}\mathbf{U}^{1} =^{t} \mathbf{V}^{1} \wedge \ldots \wedge \mathbf{U}^{n} =^{t} \mathbf{V}^{n} \wedge \mathcal{E})$, then $\theta(\mathrm{a}) = \theta(\mathrm{b})$ and $\theta(\mathbf{U}^{i}) =_{\beta\eta} \theta(\mathbf{V}^{i})$ for all $1 \leq i \leq n$ and therefore

$$\theta(h_{\mathrm{a}}\overline{\mathbf{U}^{n}}) = h_{\mathrm{b}}\overline{\theta(\mathbf{U}^{n})} =_{\beta\eta} h\overline{\theta(\mathbf{V}^{n})} = \theta(h\overline{\mathbf{V}^{n}})$$

Thus we have $\theta \in \mathbf{U}(\mathcal{E})$. The $\mathcal{SIM}(elim{:}col)$ case is a direct consequence of Lemma 4.5.

For $\mathcal{SIM}(elim{:}term)$ let $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}'$ be a $\mathcal{C}$-unification problem and $(F_{\mathrm{a}}\overline{X}) =^{t} \mathbf{A}$ be the pair in $\mathcal{E}'$ that the rule $\mathcal{SIM}(elim{:}term)$ acts upon. Furthermore let $F_{\mathrm{b}} \in \mathbf{free}(\mathcal{E}')$ for some $\mathrm{b} \neq \mathrm{a}$.

We show that for an arbitrary idempotent $\mathcal{C}$-unifier θ of $\mathcal{E}$, the b-chrome variant of the formula $\lambda\overline{X^{n}}.\mathbf{A}$ is more general than $\theta(F_{\mathrm{b}})$. So let θ be an arbitrary $\mathcal{C}$-unifier of $\mathcal{E}$, then

$$\theta(F_{\mathrm{a}}) =_{\beta\eta} \theta(\lambda\overline{X}.F_{\mathrm{a}}\overline{X}) =_{\beta\eta} \lambda\overline{X}.\theta(F_{\mathrm{a}}\overline{X}) =_{\beta\eta} \lambda\overline{X}.\theta(\mathbf{A}) =_{\beta\eta} \theta(\lambda\overline{X}.\mathbf{A})$$

since the X_{i} are not in $\mathbf{Dom}(\theta)$. Now we know that $|\theta(F_{\mathrm{b}})| = \theta(F_{\mathrm{a}})|$, since θ is a $\mathcal{C}$-substitution, on the other hand $\theta(F_{\mathrm{b}})$ is compatible with b, so there is a unique b-chrome variant $\mathbf{G}$ of $\lambda\overline{X}.\mathbf{A}$ and a substitution ρ, such that $\rho(\mathbf{G}) = \lambda\overline{X}.\mathbf{A}$ by lemma 4.8, thus we obtain the assertion by 4.5. □

Clearly the $\mathcal{SIM}$ Transformations are a generalization of the first-order colored unification algorithms as they have been presented in [Hut91]. Just as the first-order unification transformations they are terminating (if $\mathcal{E} \vdash_{\mathcal{SIM}} \mathcal{E}'$, then $\mu(\mathcal{E}') \prec \mu(\mathcal{E})$) and confluent up to associativity and commutativity of $\wedge$, $=^{t}$ and $=^{c}$. Thus it makes sense to speak of a $\mathcal{SIM}$-normal form. Unlike unification for first-order logic, the $\mathcal{SIM}$-normal forms are not solved forms, but can contain pairs of the form $h_{\mathrm{a}}\overline{\mathbf{U}} =^{t} k_{\mathrm{b}}\overline{\mathbf{U}}$, where at least one of the heads h_{a} and k_{a} is a colored variable.

4.2 General Unification

The classical approach to higher-order unification reduces the problem of finding solutions for $\mathcal{SIM}$-normal pairs as described above to the following (essential) part (the general binding problem, which is virtually trivial in the classical case): Given a type α and a symbol h, find the most general well-formed formula of type α that has head h.

We will proceed in the same manner and use standard techniques from [Sny91, Koh94] to obtain the unification algorithm (cf. section 4). For $\mathcal{C}$-unification we have to analyse the general binding problem more carefully than in the classical case.

Since we work in a colored context, where we have to ensure well-coloredness of substitutions we have to specialise the notion of general bindings by requiring them to be a-chrome.

Definition 4.11 (General Binding) Let $\Gamma_{\mathcal{Z}}$ be a variable context and $\mathtt{a} \in \mathcal{Z}$, then

$$\mathbf{G} := \lambda \overline{X^l_{\alpha_l}}.@(H^1_{\mathtt{a}_1}\overline{X^l})\dots(H^m_{\mathtt{a}_m}\overline{X^l})$$

is called a **a-chrome general binding** of type $\alpha = (\overline{\beta_l} \to \gamma)$ and head @, if @ is the bound variable $X^j_{\alpha_j}$ and $\alpha_j = (\overline{\delta_m} \to \gamma)$, or $@ = h_{\mathtt{b}}$ for some $h \in \Sigma \cup \mathbf{Dom}(\Gamma_{\mathcal{Z}})$ with type $\overline{\delta_m} \to \gamma$. Furthermore, if a is a

- colour constant, then $\mathtt{b} = \mathtt{a}_i = \mathtt{a}$ for all $1 \leq i \leq m$.
- colour variable, then b is a new colour variable and the $\mathtt{a}_i$ are distinct color variables that do not occur in $\mathbf{Dom}(\Gamma_{\mathcal{Z}})$ (in particular $\mathtt{B} \neq \mathtt{a}_i$)

The new variables H^i obtain their types from the context

$$\mathcal{A} := [H^1 : \overline{\beta_l} \to \delta^1], \dots, [H^m : \overline{\beta_l} \to \delta^m]$$

which is called the **context of variables introduced for G.** Note that general bindings are unique up to the choice of names for the variables H^i and the color variables $\mathtt{a}_j$ in $\mathcal{A}$. General bindings, where the head is a bound variable X^j are called **projection bindings** (we write them as $\mathcal{G}^j_{\alpha,\mathtt{a}}(\Sigma;\Gamma_{\mathcal{Z}};\mathcal{A})$) and **imitation bindings** (written $\mathcal{G}^{h_{\mathtt{b}}}_{\alpha,\mathtt{a}}(\Sigma;\Gamma_{\mathcal{Z}};\mathcal{A})$) else.

Since we need both imitation and projection bindings for higher-order unification, we collect them in the set of **approximating bindings for h and α**

$$\mathcal{AB}^{h_{\mathtt{b}}}_{\alpha,\mathtt{a}}(\Sigma;\Gamma_{\mathcal{Z}};\mathcal{A}) := \{\mathcal{G}^{h_{\mathtt{b}}}_{\alpha,\mathtt{a}}(\Sigma;\Gamma_{\mathcal{Z}};\mathcal{A})\} \cup \{\mathcal{G}^j_{\alpha,\mathtt{a}}(\Sigma;\Gamma_{\mathcal{Z}};\mathcal{A}) | j \leq l\}$$

Note that all general bindings in $\mathcal{AB}^{h_{\mathtt{b}}}_{\alpha,\mathtt{a}}(\Sigma;\Gamma_{\mathcal{Z}};\mathcal{A})$ are a-chrome, i.e. they are a-monochrome, if $\mathtt{a} \in \mathcal{C}$ and flexichrome, if a is a color variable.

Example 4.12 The following are a-chrome general bindings

1. $\lambda Y.s_{\mathtt{c}}(H_{\mathtt{a}}Y)$ is an imitation binding for $s_{\mathtt{c}}$, if either $\mathtt{a} \in \mathcal{X}$ or $\mathtt{a} \in \mathcal{C}$ and $\mathtt{c} = \mathtt{a}$.
2. $\lambda Y.s_{\mathtt{B}}(H_{\mathtt{C}}Y)$ is an imitation binding with head $s_{\mathtt{B}}$, if $\mathtt{a} \in \mathcal{X}$.
3. $\lambda X, Y, Z.Y(H_{\mathtt{A}}YZ)$ is a 2-projection binding.

It is easily verified that $\mathcal{G}^{h_{\mathtt{a}}}_{\alpha,\mathtt{b}} \in \mathit{wff}_\alpha(\Sigma;\Gamma_{\mathcal{Z}} \cup \mathcal{A})$, that it is b-monochrome and $\mathbf{head}(\mathbf{G}) = h_{\mathtt{a}}$, which explains the naming in the definition above. The following theorem is the basis of the unification transformations given below.

Theorem 4.13 (General Binding Theorem) *Let* $\mathbf{A} = (\lambda\overline{X^k}.h_{\mathtt{b}}\overline{\mathbf{U}^m}) \in \mathit{wff}_\alpha(\Sigma;\Gamma_{\mathcal{Z}})$ *be a long head normal form that is compatible with* $\mathtt{a} \in \mathcal{C} \cup \mathcal{X}$*, then there exists a* a-*chrome general binding* $\mathbf{G} = \mathcal{GB}^{h_{\mathtt{b}}}_{\alpha,\mathtt{a}}(\Sigma;\Gamma_{\mathcal{Z}};\mathcal{A})$ *and a* $\mathcal{C}$*-substitution* $\rho \in \mathbf{SUB}(\Sigma;\mathcal{A} \to \Gamma_{\mathcal{Z}})$*, such that* $\mathbf{dp}(\rho) < \mathbf{dp}(\mathbf{A})$ *(where* $\mathbf{dp}(\mathbf{A})$ *is the depth of* $\mathbf{A}$ *as a tree) and* $\rho(\mathbf{G}) =_{\beta\eta} \mathbf{A}$.

Proof: By definition $\mathcal{GB}^{h_{\mathtt{b}}}_{\alpha,\mathtt{a}}(\Sigma;\Gamma_{\mathcal{Z}};\mathcal{A}) = \lambda\overline{X^l_{\alpha_l}}.h_{\mathtt{b}}(H^1_{\mathtt{a}_1}\overline{X^l})\dots(H^m_{\mathtt{a}_m}\overline{X^l})$, so we can take $\rho = [(\lambda\overline{X^k}.\mathbf{U}^1)/H^1_{\mathtt{b}}], \dots, [(\lambda\overline{X^k}.\mathbf{U}^m)/H^m_{\mathtt{b}}]$: Thus the depth condition is met by construction. Now it only remains to show that ρ is $\mathcal{C}$-substitution. Since the H^1 are distinct variables, we only have to show compatibility: If $\mathtt{a} \in \mathcal{C}$, then $\mathbf{A}$ and the $\mathbf{U}^i$ must be a-monochrome, since $\mathbf{A}$ is compatible with $\mathbf{A}$. If $\mathtt{a} \in \mathcal{X}$ then there is nothing to show. □

Building upon the notion of general bindings we give a set of transformations for $\mathcal{C}$-unification, which we will prove correct and complete with the methods of [Sny91, Koh94].

Definition 4.14 (*$\mathcal{CUT}$*: **Transformations for $\mathcal{C}$-Unification**) Let $\mathcal{CUT}$ be the system $\mathcal{SIM}$ augmented by the following inference rules

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.F_{\mathsf{a}}\overline{\mathbf{U}^n} =^t F_{\mathsf{b}}\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad F \in \mathbf{Dom}(\Gamma) \quad \mathsf{a},\mathsf{b} \in \mathcal{C} \cup \mathcal{X}}{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathsf{a} =^c \mathsf{b} \wedge \mathbf{U}^1 =^t \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^t \mathbf{V}^n \wedge \mathcal{E}}\ \mathcal{CUT}(dec)$$

together with the following rules where $\mathbf{G} \in \mathcal{AB}^{h_{\mathsf{b}}}_{\alpha,\mathsf{a}}(\Sigma, \Delta, \mathcal{A})$

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.F_{\mathsf{a}}\overline{\mathbf{U}} =^t h_{\mathsf{b}}\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma(F) = \alpha}{\exists\Gamma_{\mathcal{Z}} \cup \mathcal{A}.\forall\Upsilon_{\emptyset}.F_{\mathsf{a}} =^t \mathbf{G} \wedge [\mathbf{G}/F_{\mathsf{a}}](F_{\mathsf{a}}\overline{\mathbf{U}} =^t h_{\mathsf{b}}\overline{\mathbf{V}} \wedge \mathcal{E})}\ \mathcal{CUT}(\mathit{flex/rig})$$

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.F_{\mathsf{a}}\overline{\mathbf{U}} =^t G_{\mathsf{b}}\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma(F) = \alpha \quad \Gamma(G) = \beta}{\exists\Gamma_{\mathcal{Z}} \cup \mathcal{A}.\forall\Upsilon_{\emptyset}.F_{\mathsf{a}} =^t \mathbf{G} \wedge [\mathbf{G}/F_{\mathsf{a}}](F\overline{\mathbf{U}} =^t H_{\mathsf{b}}\overline{\mathbf{V}} \wedge \mathcal{E})}\ \mathcal{CUT}(guess)$$

Just as in $\mathcal{SIM}$ leave the associativity and commutativity of $\wedge$, $=^t$, and $=^c$ implicit. We have combined the classical imitation (**G** has head h_{b}) and projection (**G** is a projection binding) transformations (see [Sny91]) into $\mathcal{CUT}(\mathit{flex/rig})$. This set of rules is used with the convention that all formulae are eagerly reduced to $\mathcal{SIM}$-normal form.

As an example let us reconsider the list example from the introduction (cf. 2.2), where we had used higher-order coloured unification for constraining a speculated lemma in an induction proof. We give a full trace of the unification involved there.

Example 4.15 Let the types λ and ϵ stand for lists and their elements, and

$$\Sigma = [app\colon \lambda \to \lambda \to \lambda], [rev\colon \lambda \to \lambda], [cons\colon \epsilon \to \lambda \to \lambda], [h\colon \epsilon], [nil\colon \lambda], [u, v\colon \lambda]$$

Furthermore let $\Gamma = [F\colon \lambda \to \lambda \to \epsilon], [Y\colon \lambda], [X\colon \epsilon]$ then the unification problem

$$\exists\Gamma_{\mathcal{Z}}.F_{\mathsf{g}}(rev_{\mathsf{w}}u_{\mathsf{w}})h_{\mathsf{g}}v_{\mathsf{g}} =^t app_{\mathsf{g}}(rev_{\mathsf{B}}Y_{\mathsf{A}})(cons_{\mathsf{g}}X_{\mathsf{g}}nil_{\mathsf{g}})$$

can be transformed to

$$\begin{aligned}\exists\Gamma,\ & [H, K\colon \lambda \to \epsilon \to \lambda \to \lambda].\\ & F_{\mathsf{g}} =^t (\lambda U_\lambda V_\epsilon W_\lambda.app_{\mathsf{g}}(HUVW)(KUVW))\wedge\\ & F_{\mathsf{g}}(rev_{\mathsf{w}}u_{\mathsf{w}})h_{\mathsf{g}}v_{\mathsf{g}} =^t app_{\mathsf{g}}(rev_{\mathsf{B}}Y_{\mathsf{A}})(cons_{\mathsf{g}}X_{\mathsf{g}}nil_{\mathsf{g}})\end{aligned}$$

by $\mathcal{CUT}(\mathit{flex/rig})$ and further to

$$\begin{aligned}\exists[Y\colon\lambda],\ & [X\colon\epsilon], [H, K\colon \lambda \to \epsilon \to \lambda \to \lambda].\\ & F_{\mathsf{g}} =^t (\lambda U_\lambda V_\epsilon W_\lambda.app_{\mathsf{g}}(HUVW)(KUVW))\wedge\\ & app_{\mathsf{g}}(H_{\mathsf{g}}(rev_{\mathsf{w}}u_{\mathsf{w}})h_{\mathsf{g}}v_{\mathsf{g}})(K_{\mathsf{g}}(rev_{\mathsf{w}}u_{\mathsf{w}})h_{\mathsf{g}}v_{\mathsf{g}}) =^t app_{\mathsf{g}}(rev_{\mathsf{B}}Y_{\mathsf{A}})(cons_{\mathsf{g}}X_{\mathsf{g}}nil_{\mathsf{g}})\end{aligned}$$

by $\mathcal{SIM}(\mathit{elim{:}term})$. Note that there are no variants of F, so F_{g} can be eliminated right away. Now we can proceed by decomposing the problem by $\mathcal{SIM}(dec)$ to

$$\begin{aligned}\exists[Y\colon\lambda],\ & [X\colon\epsilon], [H, K\colon \lambda \to \epsilon \to \lambda \to \lambda].\\ & F_{\mathsf{g}} =^t (\lambda U_\lambda V_\epsilon W_\lambda.app_{\mathsf{g}}(HUVW)(KUVW))\wedge\\ & H_{\mathsf{g}}(rev_{\mathsf{w}}u_{\mathsf{w}})h_{\mathsf{g}}v_{\mathsf{g}} =^t rev_{\mathsf{B}}Y_{\mathsf{A}} \wedge K_{\mathsf{g}}(rev_{\mathsf{w}}u_{\mathsf{w}})h_{\mathsf{g}}v_{\mathsf{g}}) =^t cons_{\mathsf{g}}X_{\mathsf{g}}nil_{\mathsf{g}}\end{aligned}$$

Here, we choose[9] the 1-projection binding $(\lambda U_\lambda V_\epsilon W_\lambda.U)$ for H_g in $\mathcal{CUT}(\mathit{flex/rig})$, subsequent elimination[10] of H_g with $\mathcal{SIM}(\mathit{elim{:}term})$ yields

$$\begin{aligned}\exists\Gamma,\quad & [H{:}\lambda \to \epsilon \to \lambda \to \lambda].\\ & F_g =^t (\lambda U_\lambda V_\epsilon W_\lambda.app_g X(KUVW))\\ & rev_{\mathbf{w}}u_{\mathbf{w}} =^t rev_{\mathbf{B}}Y_{\mathbf{A}} \wedge K_g(rev_{\mathbf{w}}u_{\mathbf{w}})h_g v_g =^t cons_g X_g nil_g\end{aligned}$$

finally we can decompose again and eliminate $Y_{\mathbf{A}}$ for $u_{\mathbf{w}}$ yielding

$$\begin{aligned}\exists\Gamma,\quad & [K{:}\lambda \to \epsilon \to \lambda \to \lambda].\\ & F_g =^t (\lambda U_\lambda V_\epsilon W_\lambda.app_g U(KUVW))\\ & Y_{\mathbf{A}} =^t u_{\mathbf{w}} \wedge K_g(rev_{\mathbf{w}}u_{\mathbf{w}})h_g v_g =^t cons_g X_g nil_g\end{aligned}$$

Now we choose the imitation $\lambda U, V, W.cons_g(M_gUVW)(N_gUVW)$ for K_g and decompose to arrive at

$$\begin{aligned}\exists\Gamma\quad & [M{:}\lambda \to \epsilon \to \lambda \to \epsilon], [N{:}\lambda \to \epsilon \to \lambda \to \lambda].\\ & F_g =^t (\lambda U_\lambda V_\epsilon W_\lambda.app_g(U, cons(M_gUVW)(N_gUVW))\\ & Y_{\mathbf{A}} =^t u_{\mathbf{w}} \wedge M_g(rev_{\mathbf{w}}u_{\mathbf{w}})h_g v_g =^t X_g \wedge N_g(rev_{\mathbf{w}}u_{\mathbf{w}})h_g v_g =^t nil_g\end{aligned}$$

Now we can solve N_g with the imitation binding $\lambda U, V, W.nil_g$ and simplify to

$$\begin{aligned}\exists\Gamma\quad & [M{:}\lambda \to \epsilon \to \lambda \to \epsilon].\\ & F_g =^t (\lambda U_\lambda V_\epsilon W_\lambda.app_g(U(cons_g(M_gUVW)nil_g))\\ & Y_{\mathbf{A}} =^t u_{\mathbf{w}} \wedge X_g =^t M_g(rev_{\mathbf{w}}u_{\mathbf{w}})h_g v_g\end{aligned}$$

which can be solved by choosing $(\lambda U_\lambda V_\epsilon W_\lambda.V)$[11] for M_g (the last pair is simplified to $h_g =^t h_g$).

Thus one final solution of the unification problem is

$$[\lambda U_\lambda V_\epsilon W_\lambda.app_g(U, cons_g(V, nil_g))/F_g], [u_{\mathbf{w}}/Y_{\mathbf{A}}], [h_g/X_g]$$

We have indicated the choice points for the other solutions in the footnotes.

Lemma 4.16 *Let $\mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}'$ by a single application of $\mathcal{CUT}(dec)$ to a pair $h_{\mathbf{a}}\overline{\mathbf{U}^n} =^t h_{\mathbf{a}}\overline{\mathbf{V}^n}$, then for any substitution θ we have*

1. *If $h_{\mathbf{a}} \in \mathbf{Dom}(\theta)$, then $\theta \in \mathbf{U}(\mathcal{E}')$ implies that $\theta \in \mathbf{U}(\mathcal{E})$.*
2. *If $h_{\mathbf{a}} \notin \mathbf{Dom}(\theta)$, then $\theta \in \mathbf{U}(\mathcal{E})$, iff $\theta \in \mathbf{U}(\mathcal{E}')$.*

Proof: Let $\theta \in \mathbf{U}(\mathcal{E}')$, so $\theta(\mathbf{U}^i) {=_\beta} \theta(\mathbf{V}^i)$ for all $1 \le i \le n$ and therefore

$$\theta(h_{\mathbf{a}}\overline{\mathbf{U}^n}) = \theta(h_{\mathbf{a}})\overline{\theta(\mathbf{U}^n)} {=_\beta} \theta(h_{\mathbf{a}})\overline{\theta(\mathbf{V}^n)} = \theta(h_{\mathbf{a}}\overline{\mathbf{V}^n})$$

Thus for any atom $h_{\mathbf{a}}$ we have $\theta \in \mathbf{U}(\mathcal{E})$. Now let $h_{\mathbf{a}} \notin \mathbf{Dom}(\theta)$ and $\theta \in \mathbf{U}(\mathcal{E})$, then $\theta(h_{\mathbf{a}}) = h_{\mathbf{a}}$, so in this case we have $\theta \in \mathbf{U}(\mathcal{E}')$. □

By applying the rules $\mathcal{CUT}(\mathit{flex/rig})$ and $\mathcal{CUT}(\mathit{guess})$ we effectively commit ourselves to a particular approximation of a solution, and thus cannot reasonably expect to conserve the set of $\mathcal{C}$-unifiers.

[9]The 2-projection binding is impossible for type reasons and the 3-projection binding leads to immediate subsequent clash. The imitation binding leads to a solution $\lambda UVW.app_g(rev_g(L_gUVW))(cons_gY nil)$ for F_g that is not wanted in our motivating example, so we will not pursue it here.

[10]since the variable H does not occur in the original problem, we need not record it in the unification problem.

[11]The 3-projection $(\lambda U_\lambda V_\epsilon W_\lambda.W)$ or the imitation binding $(\lambda U_\lambda V_\epsilon W_\lambda.Q_g)$ for some new variable Q would also have worked.

Lemma 4.17 *If $\mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}'$ by a $\mathcal{CUT}$-derivation only containing applications of the rules $\mathcal{CUT}$(flex/rig) and $\mathcal{CUT}$(guess), then* $\mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.

Proof: The transformations $\mathcal{CUT}$*(flex/rig)* and $\mathcal{CUT}$*(guess)* can be divided into three parts, first adding a pair $X =^t \mathbf{G}$, then eliminating the variable, and finally $\mathcal{SIM}$-reducing. Clearly adding a new pair does not create new $\mathcal{C}$-unifiers, so we must have $\mathbf{U}(\mathcal{E} \wedge X =^t \mathbf{A}) \subset \mathbf{U}(\mathcal{E})$. Thus obtain the assertion with 4.4(2) and 4.10. □

Theorem 4.18 (Soundness of $\mathcal{CUT}$) *If $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}' \vdash_{\mathcal{CUT}} \mathcal{E}''$ such that $\mathcal{E}''$ is in $\mathcal{C}$-solved form, then the substitution $\sigma_{\mathcal{E}''}|_{\mathbf{Dom}(\Gamma_{\mathcal{Z}})} \in \mathbf{U}(\mathcal{E})$.*

Proof: We prove $\sigma_{\mathcal{E}''} \in \mathbf{U}(\mathcal{E})$ by induction on the length of the transformation sequence using the above Lemmata in the induction step. The restriction of $\sigma_{\mathcal{E}''}$ does not affect the fact that $\sigma_{\mathcal{E}''}|_{\mathbf{Dom}(\Gamma_{\mathcal{Z}})}$ still $\mathcal{C}$-unifies $\mathcal{E}$. □

So if the algorithm $\mathcal{CUT}$ returns a substitution θ for an initial system $\mathcal{E}$, then θ is indeed a $\mathcal{C}$-unifier for $\mathcal{E}$. The main result of this section is the converse, namely, that given an initial $\mathcal{C}$-unification problem $\mathcal{E}$ and a $\mathcal{C}$-unifier θ, the algorithm $\mathcal{CUT}$ can compute a $\mathcal{C}$-unifier σ of $\mathcal{E}$, which is more general than θ.

As higher-order unification is undecidable [Gol81], our set of transformations cannot be terminating in general. We will prove, that $\mathcal{CUT}$ is a complete $\mathcal{C}$-unification procedure, that is, if for any given $\theta \in \mathbf{U}(\mathcal{E})$ there is a $\mathcal{CUT}$-derivation $\mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}'$ such that $\mathcal{E}'$ is a $\mathcal{C}$-unification problem in $\mathcal{C}$-solved form, and $\sigma'_{\mathcal{E}}$ is more general than θ. For this we only need termination for $\mathcal{CUT}$ inference rules that approximate θ.

The following measure provides the basis for the following semi-termination result for $\mathcal{C}$-unification.

Definition 4.19 Let $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}'$ be a $\mathcal{C}$-unification problem and θ be a $\mathcal{C}$-substitution, then

$$\mu(\mathcal{E},\theta) := (\mu_1(\mathcal{E},\theta), \mu_2(\mathcal{E}))$$

is called a **measure for $\mathcal{E}$ and θ**, iff $\mu_1(\mathcal{E},\theta)$ is a multiset of depths formulae $\theta(\mathbf{A}_a)$, where $\mathbf{A}_a \in \mathbf{Dom}(\theta)$ is unsolved in $\mathcal{E}$ and $\mu_2(\mathcal{E})$ is the multiset of depths of formulae in $\mathcal{E}$. Furthermore, let $\prec$ be the strict lexicographic ordering for the obvious component orderings.

Lemma 4.20 *Let $\mathcal{E}$ be a $\mathcal{C}$-unification problem in $\mathcal{SIM}$-normal form, but not in $\mathcal{C}$-solved form, $\theta \in \mathbf{U}(\mathcal{E})$, then there exists a $\mathcal{C}$-unification problem $\mathcal{E}'$, and an $\mathcal{E}'$-substitution θ', such that $\mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}'$, and*

1. $\theta = \theta'[\mathcal{E}]$,
2. $\theta' \in \mathbf{U}(\mathcal{E}')$,
3. $\mu(\mathcal{E}',\theta') \prec \mu(\mathcal{E},\theta)$.

Proof: Let $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}''$ and $\mathbf{A} =^t \mathbf{B}$ be a pair in $\mathcal{E}''$, that is not $\mathcal{C}$-solved. Furthermore let $\mathbf{A} = \mathbf{F}\overline{\mathbf{U}}$ and $\mathbf{B} = \mathbf{G}\overline{\mathbf{V}}$. We observe that $\mathbf{F}$ and $\mathbf{G}$ must be atoms and cannot be equal constants, and moreover we cannot have $\mathbf{A} =_{\beta\eta} \mathbf{B}$, since $\mathcal{E}$ is in $\mathcal{SIM}$-normal form.

If $\mathbf{F}, \mathbf{G} \in \mathcal{V} \setminus \mathbf{Dom}(\theta)$, then $\mathcal{CUT}$*(dec)* applies. By 4.16 we have $\theta \in \mathbf{U}(\mathcal{E}')$ and $\mu(\mathcal{E}',\theta) \prec \mu(\mathcal{E},\theta)$, since $\mu_1(\mathcal{E}',\theta) \preceq \mu_1(\mathcal{E},\theta)$ and $\mu_2(\mathcal{E}') \prec \mu(\mathcal{E})$.

Otherwise either $\mathbf{F} \neq \mathbf{G}$ or $\mathbf{F} = \mathbf{G} \in \mathbf{Dom}(\theta)$. In both cases, since $\mathcal{E}$ is $\mathcal{C}$-unifiable, either $\mathbf{F}$ or $\mathbf{G}$ is an colored variable $F_a \in \mathbf{Dom}(\theta)$ with $\Gamma(F_a) = \alpha$ at the head. Without loss of generality we assume that $F_a = \mathbf{F}$. By the general binding theorem 4.13 there exists a general binding $\mathbf{G} \in \mathcal{AB}_{\alpha,\mathbf{a}}^{\mathbf{head}(\theta(F_{\mathbf{a}}))}(\Sigma, (\Xi, \Gamma_{\mathcal{Z}}), \mathcal{A})$ of type α and a $\mathcal{C}$-substitution ρ, such that $\mathbf{Dom}(\rho) = \mathbf{Dom}(\mathcal{A})$ and $\rho(\mathbf{G}) =_{\beta\eta} \theta(F_{\mathbf{a}})$. Therefore,

- if $\mathbf{head}(\mathbf{G}) \notin \mathbf{Dom}(\theta)$, then $\mathcal{CUT}(flex/rig)$ applies.
- if $\mathbf{head}(\mathbf{G}) \in \mathbf{Dom}(\theta)$ then $\mathcal{CUT}(guess)$ applies.

In all these cases we set $\theta' := \theta \cup \rho$ and have $\theta = \theta'[\mathcal{E}]$, since $\mathbf{Dom}(\rho) \cap \mathbf{Dom}(\Gamma_{\mathcal{Z}}) = \mathbf{Dom}(\mathcal{A}) \cap \mathbf{Dom}(\Gamma_{\mathcal{Z}}) = \emptyset$ and $\theta' \in \mathbf{U}(\mathcal{E}')$ by 4.18.

The rules $\mathcal{CUT}(guess)$ and $\mathcal{CUT}(flex/rig)$ remove $F_{\mathbf{a}}$ from the set of variables in $\mathbf{Dom}(\theta)$ that are not $\mathcal{C}$-solved in $\mathcal{E}$ and replace it with the set $\mathbf{Dom}(\mathcal{A}) = \mathbf{Dom}(\rho)$. Since the depth of ρ is smaller than that for $\theta(F)$, we have $\mu_1(\mathcal{E}', \theta') \prec \mu_1(\mathcal{E}, \theta)$. Thus we have $\mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$. □

If we call such a transformation **μ-prescribed**, then each application of a μ-prescribed transformation decreases the well-founded measure μ. Thus any sequence of μ-prescribed transformations must terminate. The previous lemma also guarantees that any system obtained by exhaustively applying μ-prescribed transformations to a $\mathcal{C}$-unifiable system must be $\mathcal{C}$-solved, since otherwise it guarantees another μ-prescribed transformation.

Corollary 4.21 *If $\mathcal{E}$ is a $\mathcal{C}$-unifiable unification problem such that no μ-prescribed transformation rule from $\mathcal{CUT}$ is applicable, then $\mathcal{E}$ is in $\mathcal{C}$-solved form.*

Theorem 4.22 (Completeness Theorem for $\mathcal{CUT}$) *For any $\mathcal{C}$-unification problem $\mathcal{E}$ and any $\mathcal{C}$-substitution $\theta \in \mathbf{U}(\mathcal{E})$, there is a $\mathcal{CUT}$-derivation $\mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}'$ such that $\mathcal{E}'$ is in $\mathcal{C}$-solved form and $\sigma'_{\mathcal{E}} \leq_{\beta\eta} \theta[\mathcal{E}]$.*

Proof: Let $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}''$ and $\mathcal{D}: \mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}'$ be a maximal μ-prescribed $\mathcal{CUT}$-derivation out of $\mathcal{E}$. By 4.20 this is always finite, so we can prove the assertion by induction on the number n of nodes in $\mathcal{D}$. If $n = 0$, then $\mathcal{E}$ is in $\mathcal{C}$-solved form and $\sigma_{\mathcal{E}}$ is a most general $\mathcal{C}$-unifier for $\mathcal{E}$. In particular, we have $\sigma_{\mathcal{E}} \leq_{\beta\eta} \theta[\mathcal{E}]$.

If $n > 0$, then there is a μ-prescribed transformation $\mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}'$ and a $\mathcal{C}$-substitution θ' satisfying 4.20. By inductive hypothesis there is a $\mathcal{CUT}$-derivation $\mathcal{E}' \vdash_{\mathcal{CUT}} \mathcal{E}''$ such that $\sigma''_{\mathcal{E}} =_{\beta\eta} \theta'[\mathcal{E}']$. By 4.18 we have $\sigma''_{\mathcal{E}} \in \mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$. Furthermore, by inspection of the inference rules we see that $\mathcal{CUT}$ rules only expand the set of colored variables in $\Gamma_{\mathcal{Z}}$, so $\sigma_{\mathcal{E}''} \leq_{\beta\eta} \theta'[\mathcal{E}']$ implies $\sigma_{\mathcal{E}''} \leq_{\beta\eta} \theta'[\mathcal{E}]$, which in turn yields the assertion with the conclusion $\theta' = \theta[\mathcal{E}]$ of 4.20. □

If we combine the soundness results theorem 4.18 with the completeness result from theorem 4.22, we can characterise the set of solutions found by the algorithm $\mathcal{CUT}$ by the following corollary.

Corollary 4.23 *For any $\mathcal{C}$-unification problem $\mathcal{E}$ the set*

$$\mathcal{CUT}(\mathcal{E}) := \{\sigma'_{\mathcal{E}} | \mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}' \text{ and } \mathcal{E}' \text{ is in } \mathcal{C}\text{-solved form}\}$$

is a complete set of $\mathcal{C}$-unifiers for $\mathcal{E}$.

4.3 Pre-$\mathcal{C}$-Unification

As for unification in the simply typed lambda calculus, the rule $\mathcal{CUT}(guess)$ 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.

In case of the colored lambda-calculus a flex-flex pair may have no solution if the top-level variables of both terms are annotated by different colors. Consider the following examples:

Example 4.24 Let $F, G \in \mathbf{Dom}(\Gamma)$, then the unification problem $\exists\Gamma_{\mathcal{Z}}.F_{\mathsf{d}}a_{\mathsf{d}} =^t G_{\mathsf{c}}a_{\mathsf{c}}$ has no unifier. On the other hand $\exists\Gamma_{\mathcal{Z}}.F_{\mathsf{d}}a_{\mathsf{c}} =^t G_{\mathsf{c}}a_{\mathsf{c}}$ has an unifier $[\lambda X.X/F_{\mathsf{d}}], [\lambda X.X/G_{\mathsf{c}}]$.

The reason for this is the fact that projections, i.e. terms of the form $\lambda\overline{X^k}.X^i$, have no color informations but are valid instances of colored variables like F_{d} or G_{c}. Hence, in order to solve such flex-flex pairs we have to map one of the top-level variables to a projection formula. This gives rise to the following definition:

Definition 4.25 (Flexible Chain) Let $\mathcal{E}$ be a $\mathcal{C}$-unification problem, then a set $\mathcal{E}' = \mathbf{A}^1 =^t \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^t \mathbf{B}^n$ of flex/flex pairs in $\mathcal{E}$ is called a a **flexible chain** of $\mathcal{E}$ iff $\mathbf{head}(\mathbf{A}^i) = \mathbf{head}(\mathbf{B}^{i-1}) \in \mathcal{V}_{\mathcal{X}}$ for $2 \le i \le n$. We call $\mathbf{head}(\mathbf{A}^1) = F_{\mathsf{c}}$ and $\mathbf{head}(\mathbf{B}^n) = G_{\mathsf{d}}$ the left and right ends of $\mathcal{E}'$.

If $\mathsf{c}, \mathsf{d} \in \mathcal{C}$ and $\mathsf{c} \neq \mathsf{d}$ then we call $\mathcal{E}'$ a **reducible chain**, otherwise **safe chain**, similarly, we call a pair in $\mathcal{E}'$ **safe**, iff there is no reducible chain in $\mathcal{E}'$ that contains it, and a unification problem, if it does not contain reducible chains.

It will turn out that safe chains always have solutions, whereas a reducible chain in a system $\mathcal{E}$ indicates a clash of different color annotations to the top-level variables. As mentioned above the resolution of this clash will be to map one of these top-level variables to a projection formula. Thus, we can step by step reduce the number of reducible chains in $\mathcal{E}$.

Lemma 4.26 *Let $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}' \wedge \mathcal{E}_r$, where $\mathcal{E}_r = \mathbf{A}^1 =^t \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^t \mathbf{B}^n$ is a reducible chain, then for each $\mathcal{C}$-unifier σ of $\mathcal{E}$, there is a number $1 \le i \le n$, such that $\sigma(\mathbf{head}(\mathbf{A}^i))$ or $\sigma(\mathbf{head}(\mathbf{B}^i))$ is a projection formula.*

Proof: Let $F^i_{\mathsf{a}_i} = \mathbf{head}(\mathbf{A}^i)$ and $G^i_{\mathsf{b}_i} = \mathbf{head}(\mathbf{B}^i)$, then $\mathsf{a}_1, \mathsf{b}_n \in \mathcal{C}$, but $\mathsf{a}_1 \neq \mathsf{b}_n$, since $\mathcal{E}_r$ is reducible by assumption. If we assume that none of the $F^i_{\mathsf{a}_i} = \mathbf{head}(\mathbf{A}^i)$ and $G^i_{\mathsf{b}_i} = \mathbf{head}(\mathbf{B}^i)$ is a projection, then we have

$$\begin{aligned}
\mathbf{head}(\sigma(F^1_{\mathsf{a}_1})) &= \mathbf{head}(\sigma(\mathbf{A}^1)) = \mathbf{head}(\sigma\mathbf{B}^1)) = \mathbf{head}(\sigma(G^1_{\mathsf{b}_n})) = \mathbf{head}(\sigma(F^2_{\mathsf{a}_2}))\\
&= \mathbf{head}(\sigma(\mathbf{A}^2)) = \ldots = \mathbf{head}(\sigma(G^n_{\mathsf{b}_n}))
\end{aligned}$$

However $\sigma(F^1_{\mathsf{a}_1})$ and $\sigma(G^n_{\mathsf{b}_n})$ must be monochrome, as σ is well-colored and therefore $\mathsf{a}_1 = \mathsf{b}_n$, which contradicts our assumption that $\mathcal{E}_r$ is reducible. $\Box$

Definition 4.27 (Pre-$\mathcal{C}$-Solved Form) Let $\mathcal{E}$ be a $\mathcal{C}$-unification problem the we call a pair $\mathbf{A} =^t \mathbf{B}$ in $\mathcal{E}$ **pre-solved** in $\mathcal{E}$, iff $\mathbf{A} =^t \mathbf{B}$ is solved in $\mathcal{E}$ or $\mathbf{A} =^t \mathbf{B}$ is a safe flex/flex pair. We call $\mathcal{E}$ pre-$\mathcal{C}$-solved, iff all of its pairs are. Thus $\mathcal{E}$ is pre-$\mathcal{C}$-solved, iff all of its pairs are solved or flex/flex and safe.

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

Definition 4.28 (Color Restriction) Let $\mathcal{E}$ be a safe system, then the **color restriction** $cr(X_{\mathsf{a}}, \mathcal{E})$ of a colored variable X_{a} with respect to $\mathcal{E}$ is defined by

- $cr(X_{\mathsf{a}}, \mathcal{E}) = \mathsf{d}$ if $\mathsf{a} \in \mathcal{X}$ and there is flexible chain $\mathcal{E}'$ in $\mathcal{E}$ with left head X_{a} and right head Y_{d} for some $\mathsf{d} \in \mathcal{C}$.
- $cr(X_{\mathsf{a}}, \mathcal{E}) = \mathsf{a}$ otherwise.

Given a safe system $\mathcal{E}$ the notion of color restriction is well-defined. Suppose, there are two subsets of $\mathcal{E}$ satisfying the condition of the definition above which result in different color restrictions c and c' for a colored variable atom X_{a}. Merging both sets we would obtain a reducible chain in $\mathcal{E}$, which contradicts our assumption that $\mathcal{E}$ is safe. Note that for nay flex/flex pair $F_{\mathsf{a}}\overline{\mathbf{U}} =^t G_{\mathsf{b}}\overline{\mathbf{V}}$ in $\mathcal{E}$ either

- $cr(F_{\mathsf{a}}, \mathcal{E}) = cr(G_{\mathsf{b}}, \mathcal{E}) \in \mathcal{C}$ or
- both $cr(F_{\mathsf{a}}, \mathcal{E})$ and $cr(G_{\mathsf{b}}, \mathcal{E})$ are color variables.

In the first case we furthermore know that either $\mathsf{a} \in \mathcal{X}$ or $cr(F_{\mathsf{a}}, \mathcal{E}) = \mathsf{a}$ (and similarly for b and $cr(G_{\mathsf{b}}, \mathcal{E})$).

Example 4.29 Both unification problems $\exists\Gamma_{\mathcal{Z}}.F_{\mathsf{d}}a_{\mathsf{d}} =^t G_{\mathsf{c}}a_{\mathsf{c}}$ and $\exists\Gamma_{\mathcal{Z}}.F_{\mathsf{d}}a_{\mathsf{c}} =^t G_{\mathsf{c}}a_{\mathsf{c}}$ from example 4.24 are reducible flexible chains, so any unifier has to be a projection. Indeed for the second one, the projection bindings $[\lambda X.X/F_{\mathsf{d}}], [\lambda X.X/G_{\mathsf{c}}]$ succeed, whereas they clash for the first problem.

The problem $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.F_{\mathsf{a}}a_{\mathsf{c}} =^t \mathbf{G}_{\mathsf{A}}b_{\mathsf{c}} \wedge G_{\mathsf{A}} =^t H_{\mathsf{B}}b_{\mathsf{d}}$ is safe, and $cr(G_{\mathsf{A}}, \mathcal{E}) = cr(H_{\mathsf{B}}, \mathcal{E}) = \mathsf{a}$. Finally $\mathcal{F} = \exists\Gamma_{\mathcal{Z}}.F_{\mathsf{A}}a_{\mathsf{c}} =^t G_{\mathsf{B}}b_{\mathsf{d}}$ is safe with $cr(F_{\mathsf{A}}, \mathcal{F}) = \mathsf{A}$ and $cr(G_{\mathsf{B}}, \mathcal{F}) = \mathsf{B}$.

Definition 4.30 (Trivial Unifier) Let $\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}'$ be a pre-$\mathcal{C}$-solved $\mathcal{C}$-unification problem, such that $\mathcal{E}'' := F_{\mathsf{a}}\overline{\mathbf{U}^n} =^t G_{\mathsf{b}}\overline{\mathbf{V}^m}$ is a pre-$\mathcal{C}$-solved pair in $\mathcal{E}$ with $\Delta(F) = \overline{\alpha^n} \to \beta$, $\Delta(G) = \overline{\gamma^m} \to \beta$. Furthermore let $\mathcal{H} := \{H^\beta | \beta \in \mathcal{T}\}$ be a reserved set of typed variables with $\mathcal{H} \cap \mathbf{Dom}(\Gamma) = \emptyset$. Furthermore let

$$\zeta_{\mathcal{E}''} := [\lambda X^1_{\alpha_1} \ldots X^n_{\alpha_n}.H^\beta_{cr(F_{\mathsf{a}},\mathcal{E})}/F_{\mathsf{a}}], [\lambda X^1_{\gamma_1} \ldots X^m_{\gamma_m}.H^\beta_{cr(G_{\mathsf{b}},\mathcal{E})}/G_{\mathsf{b}}]$$

If $cr(F_{\mathsf{a}}, \mathcal{E}) \neq cr(G_{\mathsf{b}}, \mathcal{E})$, then both are color variables and $\zeta_{\mathcal{E}''}$ is augmented by the color substitution $[cr(G_{\mathsf{b}}, \mathcal{E})/cr(F_{\mathsf{a}}, \mathcal{E})]$. Finally, we define $\zeta_{\mathcal{E}}$ as the union of the $\zeta_{\mathcal{E}''}$ for all flex-flex pairs $\mathcal{E}''$ in $\mathcal{E}'$.

The next Lemma shows that pre-$\mathcal{C}$-unifiers can always be extended to $\mathcal{C}$-unifiers by finding trivial $\mathcal{C}$-unifiers for the pre-$\mathcal{C}$-solved pairs. Therefore a Σ-unification problem $\mathcal{E}$ is pre-$\mathcal{C}$-unifiable, iff it is $\mathcal{C}$-unifiable.

Lemma 4.31 *Let $\mathcal{E}$ be a pre-$\mathcal{C}$-solved unification problem, then $\sigma_{\mathcal{E}} \cup \zeta_{\mathcal{E}}$ is a $\mathcal{C}$-unifier of $\mathcal{E}$*

Proof: Let $\mathcal{E}''$ and $\zeta_{\mathcal{E}''}$ be as in 4.30, then $\zeta_{\mathcal{E}''}$ is a $\mathcal{C}$-unifier for $\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E}''$, since

$$\zeta_{\mathcal{E}''}(F_{\mathsf{a}}\overline{\mathbf{U}^n}) =_{\beta\eta} H_{cr(G_{\mathsf{b}},\mathcal{E})} =_{\beta\eta} \zeta_{\mathcal{E}''}(G_{\mathsf{b}}\overline{\mathbf{V}^m})$$

and either $cr(F_{\mathsf{a}}, \mathcal{E}) = cr(G_{\mathsf{b}}, \mathcal{E})$ or they are identified by $\zeta_{\mathcal{E}''}$. Consequently, $\sigma_{\mathcal{E}} \cup \zeta_{\mathcal{E}}$ is a pre-$\mathcal{C}$-unifier of $\mathcal{E}$, since $\sigma_{\mathcal{E}}$ unifies the $\mathcal{C}$-solved pairs in $\mathcal{E}$ and $\zeta_{\mathcal{E}}$ the flex/flex ones.

To show that $\zeta_{\mathcal{E}}$ is a $\mathcal{C}$-substitution, we verify the conditions of 3.13: We have two cases

- $cr(F_{\mathsf{a}}, \mathcal{E}) = cr(G_{\mathsf{b}}, \mathcal{E}) \in \mathcal{C}$ and $\mathsf{a} \in \mathcal{C}$ (in which case $\zeta_{\mathcal{E}}(F_{\mathsf{a}}) = \lambda X^1_{\alpha_1} \ldots X^n_{\alpha_n}.H_{cr(G_{\mathsf{b}},\mathcal{E})}$ is a $= cr(G_{\mathsf{b}}, \mathcal{E})$-monochrome) or
- $\mathsf{a} \in \mathcal{X}$, which is unproblematic.

The argumentation for $G_{\mathbf{b}}$ is analogous

For the consistency conditions on erasures note that for any variable X and colors $\mathbf{e}, \mathbf{f}$ we have $|\zeta_{\mathcal{E}}(X_{\mathbf{e}})| = |\zeta_{\mathcal{E}}(X_{\mathbf{f}})|$, since the head H^{β} and thus the erasure itself is uniquely determined by the type of X. □

Definition 4.32 ($\mathcal{CPT}$:Transformations for $\mathcal{C}$-Pre-Unification)
We define the set $\mathcal{CPT}$ of **transformations for pre-$\mathcal{C}$-unification** by modifying the $\mathcal{CUT}$ rules $\mathcal{CUT}(\mathit{dec})$ and $\mathcal{CUT}(\mathit{flex/rig})$ by requiring that they may not be performed on $\mathcal{C}$-pre-solved pairs and replacing $\mathcal{CUT}(\mathit{guess})$ by the following rule of inference.

$$\frac{\exists \Gamma_{\mathcal{Z}}.\forall \Upsilon_{\emptyset}.F_{\mathbf{a}}\overline{\mathbf{U}} =^t G_{\mathbf{b}}\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma(F) = \alpha}{\exists \Delta_{\mathcal{Z}} \cup \mathcal{A}\forall \Upsilon_{\emptyset}.F_{\mathbf{a}} =^t \mathbf{G} \wedge [\mathbf{G}/F](F_{\mathbf{a}}\overline{\mathbf{U}} =^t h_{\mathbf{b}}\overline{\mathbf{V}} \wedge \mathcal{E})}\ \mathcal{CPT}(\mathit{flex/flex})$$

for some general projection binding $\mathbf{G} \in \mathcal{GB}^{j}_{\alpha,\mathbf{a}}(\Sigma; \Delta; \mathcal{A})$, where $F_{\mathbf{a}}\overline{\mathbf{U}} =^t G_{\mathbf{b}}\overline{\mathbf{V}}$ is a pair of a reducible chain of the hypothesis.

With the definitions above we obtain a completeness result for $\mathcal{CPT}$ similar to 4.22.

Theorem 4.33 *For any $\mathcal{C}$-unification problem $\mathcal{E}$ the set*

$$\mathcal{CPT}(\mathcal{E}) := \{\sigma'_{\mathcal{E}} | \mathcal{E} \vdash_{\mathcal{CPT}} \mathcal{E}' \text{ and } \mathcal{E}' \text{ is in pre-}\mathcal{C}\text{-solved form}\}$$

is a complete set of pre-$\mathcal{C}$-unifiers for $\mathcal{E}$.

Proof sketch: The proof goes through with exactly the same methods, as we have used them in section 4: Most of the technical difficulties are encapsulated in the general binding theorem and in the analogue of 4.20 we use Lemma 4.26 to account for the restricted flex/flex-case. □

Note that in contrast to classical higher-order pre-unification we cannot drop the $\mathcal{CUT}(\mathit{guess})$ and $\mathcal{CUT}(\mathit{dec})$ rules altogether, but the restriction for is severe enough to make pre-$\mathcal{C}$-unification tractable. In particular the restriction alleviates the need for unspecified imitations in $\mathcal{CUT}(\mathit{guess})$, which makes full unification infinitely branching.

4.4 Higher-Order Patterns

There are certain syntactic fragments of the simply typed lambda calculus, where the higher-order unification problem has better properties than in the general case. We will concentrate on *higher-order patterns* [Mil92], where the problem is unitary for the uncolored case. In the colored case, the problem is slightly more complex, and we will profit from the understanding of colored flex/flex pairs that we have achieved in the last section. Higher-order pattern extensions rippling have already been studied in the context of program synthesis in [Kra94], without arriving at an satisfying algorithm or treatment of the meta-theory. The theory presented below can a posteriori be taken as a logical basis for the Kraan's work.

For the colored λ-calculus, the definition of higher-order patterns is exactly as in the uncolored case (we will reiterate it here, to make the paper self-contained).

Definition 4.34 (Higher-Order Pattern) We call a formula $\mathbf{A} \in \mathit{wff}$ a **higher-order pattern**, iff any occurrence of a free variable $F \in \mathbf{Dom}(\Gamma_{\mathcal{Z}})$ in $\mathbf{A}$ must be in a subformula $\mathbf{B}$ of $\mathbf{A}$ of the form $FX^{\varphi(1)} \ldots XF^{\varphi(n)}$, where the $X^1, \ldots, X^n$

are bound in $\mathbf{A}$ and φ is a **partial permutation** from k into n, i.e. an injective mapping from $\{1,\dots,k\}$ into $\{1,\dots,n \geq k\}$, where k is the length of the type $\Gamma(F)$. In other words, all free variables of a higher-order pattern occur at the leaves, or applied to a list of distinct bound variables.

We will call a formula $\mathbf{A}$ in a unification problem $\mathcal{E} := \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.\mathcal{E} \wedge \mathbf{A} =^t \mathbf{B}$ a higher-order pattern, iff the X^i are bound in $\mathbf{A}$ or in $\mathbf{Dom}(\Upsilon)$; i.e. the universally quantified variables from the declaration are also allowed for the as the arguments for the existential ones (which take the rôle of the free ones). We will call $\mathcal{E}$ a pattern unification problem, iff all formulae occurring in its pairs are higher-order patterns.

Finally, we will call a $\mathcal{C}$-substitution $\sigma \in \mathbf{SUB}(\Sigma; \Delta \to \Gamma)$ a pattern substitution, iff for all X_{a} in $\mathbf{Dom}(\sigma)$, $\sigma(X_{\mathrm{a}})$ is a higher-order pattern.

Example 4.35 Let f, a be constants and F, G be variables of appropriate type, then $\lambda XYZ.F_{\mathrm{a}}ZX$ and $\lambda X f_{\mathrm{A}}(\lambda Y F_{\mathrm{b}}XY)a_{\mathrm{a}}(G_{\mathrm{a}}X)$ are higher-order patterns, while Fa, $\lambda X.F_{\mathrm{c}}Xa_{\mathrm{B}}$, $\lambda XF_{\mathrm{a}}XX$, and $\lambda XYF_{\mathrm{c}}X(YX)$ are not. Furthermore, all first-order formulae and all closed formulae are higher-order patterns, since they do not contain free function variables. Finally, rigid general bindings are higher-order patterns, while flexible are not in general.

Lemma 4.36 *The class of higher-order patterns is closed under the application of pattern $\mathcal{C}$-substitutions and β-reduction.*

Proof: We will only summarise the main idea of the proof, which can be made formal by a simple simultaneous induction over the structure of the formulae involved.

Consider a subformula $\mathbf{B} = FX^{\varphi(1)} \dots XF^{\varphi(n)}$ of a higher-order pattern $\mathbf{A}$, where $F \in \mathbf{Dom}(\sigma)$ and $\sigma(F) = \lambda\overline{Y^n}.\mathbf{C}$. Then $\sigma(\mathbf{B}) = (\lambda\overline{Y^n}.\mathbf{C})\overline{X^{\varphi(n)}}$ which is obviously a higher-order pattern (only the variable F has been eliminated). Furthermore, β-reduction of $\sigma(\mathbf{B})$ will only replace the variables Y^i in $\mathbf{C}$ that were bound in $\sigma(F)$ for the X^i that are bound in $\mathbf{A}$. $\square$

However, unlike to the uncolored case, colored pattern unification cannot be unitary, since conflicting colors on flex/flex pairs can force the instantiations to be (uncolored) projections. As we have seen above, conflicting colors can entail that flex/flex pairs are unsolvable, on the other hand, for pattern unification, they can also lead multiple solutions (the erasures of which can be represented by a more general uncolored higher-order pattern). Consider for instance the pair

$$\lambda X^{\alpha}, Y^{\alpha}, Z^{\alpha}, W^{\alpha}.F_{\mathrm{a}}XYZW =^t \lambda X^{\alpha}, Y^{\alpha}, Z^{\alpha}, W^{\alpha}.F_{\mathrm{b}}YXZW$$

where α is a base type and $\mathrm{a}, \mathrm{b} \in \mathcal{C}$. Obviously, there are two most general solutions

$$\begin{aligned}
\sigma_3 &:= [\lambda X^{\alpha}, Y^{\alpha}, Z^{\alpha}, W^{\alpha}.Z/F_{\mathrm{a}}], [\lambda X^{\alpha}, Y^{\alpha}, Z^{\alpha}, W^{\alpha}.Z/F_{\mathrm{b}}] \\
\sigma_4 &:= [\lambda X^{\alpha}, Y^{\alpha}, Z^{\alpha}, W^{\alpha}.W/F_{\mathrm{a}}], [\lambda X^{\alpha}, Y^{\alpha}, Z^{\alpha}, W^{\alpha}.W/F_{\mathrm{b}}]
\end{aligned}$$

This syntactic fragment allows to specialise the unification rules from 4.14.

Definition 4.37 (Transformations for Pattern Unification) The inference rules for $\mathcal{UP}at$ are those of $\mathcal{CPT}$, together with the following additional rules for safe flex/flex pairs. The first one handles the case, where the heads are identical and the second one, where they are distinct.

Let $\Upsilon = [X^1{:}\alpha_1], \dots, [X^n{:}\alpha_n]$, $\Gamma(F) = \overline{\alpha_k} \to \beta$, and φ, ψ be partial permutations from k to n

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.F_{\mathrm{a}}\overline{X^{\varphi(k)}} =^t F_{\mathrm{b}}\overline{X^{\psi(k)}} \wedge \mathcal{E}}{\begin{array}{c}\exists\Gamma, [H{:}\overline{\alpha_{\rho(l)}} \to \beta].\forall\Upsilon_{\emptyset}.cr(F_{\mathrm{a}}, \mathcal{E}) =^c cr(F_{\mathrm{b}}, \mathcal{E}) \wedge \mathcal{E} \wedge F_{\mathrm{a}}\overline{X^{\varphi(k)}} =^t F_{\mathrm{b}}\overline{X^{\psi(k)}} \\ \wedge F_{\mathrm{a}} =^t \lambda\overline{Y^k_{\alpha_{\varphi(k)}}}.H_{\mathrm{a}}\overline{Y^{\rho(l)}} \\ \wedge F_{\mathrm{b}} =^t \lambda\overline{Y^k_{\alpha_{\varphi(k)}}}.H_{\mathrm{b}}\overline{Y^{\rho(l)}}\end{array}} \mathcal{UP}at(same)$$

where ρ is a partial permutation from k to l, such that $\rho(i) = \varphi(j)$, iff $\varphi(j) = \psi(j)$, i.e. ρ picks out all arguments, where φ and ψ coincide.

For the case with distinct heads let $\Gamma(F) = \overline{\alpha_k} \to \beta$, $\Gamma(G) = \overline{\alpha_l} \to \beta$ and φ, ψ be partial permutations from k (l) to n.

$$\frac{\exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.F_{\mathsf{a}}\overline{X^{\varphi(k)}} =^t G_{\mathsf{b}}\overline{X^{\psi(l)}} \wedge \mathcal{E}}{\begin{array}{c}\exists\Gamma, [H{:}\overline{\alpha_{\varphi'(m)}} \to \beta].\forall\Upsilon_{\emptyset}.\mathtt{A} =^c \mathtt{b} \wedge F_{\mathsf{a}}\overline{X^{\varphi(k)}} =^t G_{\mathsf{b}}\overline{X^{\psi(l)}} \wedge \mathcal{E} \\ \wedge F_{\mathsf{a}} =^t \lambda\overline{Y^k_{\alpha_{\varphi(k)}}}.H_{\mathsf{a}}\overline{Y^{\varphi'(m)}} \\ \wedge G_{\mathsf{b}} =^t \lambda\overline{Y^l_{\alpha_{\psi(l)}}}.H_{\mathsf{b}}\overline{Y^{\psi'(m)}}\end{array}}\;\mathcal{UP}at(diff)$$

where φ' and ψ' are partial permutations from m into k, l, such that $\varphi'(m) = i$ and $\psi'(m) = j$, iff $\varphi(i) = \psi(j)$.

Soundness of these rules immediately comes from the fact that they are specialisations of the $\mathcal{CUT}$ rules. We will now show completeness.

Lemma 4.38 *If $\mathcal{E}$ is a pattern unification problem and $\mathcal{E} \vdash_{\mathcal{UP}} at\mathcal{F}$, then* $\mathbf{PatU}(\mathcal{E}) = \mathbf{PatU}(\mathcal{F})$.

Proof: The assertion can be verified by a simple inspection of the rules. For $\mathcal{SIM}$ we have already verified this in 4.10. Furthermore for higher-order patterns the $\mathcal{CUT}$*(flex/rig)* rule is deterministic, that is, all but the imitation or one projection immediately lead to failure. Thus by Lemma 4.20 $\mathcal{CUT}$*(flex/rig)* must conserve the set of unifiers.

For $\mathcal{UP}at(same)$ we have

$$\mathcal{E} = \exists\Gamma_{\mathcal{Z}}.\forall\Upsilon_{\emptyset}.F_{\mathsf{a}}\overline{X^{\varphi(k)}} =^t F_{\mathsf{b}}\overline{X^{\psi(l)}} \wedge \mathcal{E}$$

where $\Upsilon = [X^1{:}\alpha_1], \ldots, [X^n{:}\alpha_n]$, $\Gamma(F) = \overline{\alpha_k} \to \beta$, and φ, ψ be partial permutations from k to n. Now, for any solution $\sigma \in \mathbf{PatU}(\mathcal{E})$ we have $\sigma(F_{\mathsf{a}}) =_{\beta\eta} \lambda\overline{Y_k}.\mathbf{A}$ and $\sigma(F_{\mathsf{b}}) =_{\beta\eta} \lambda\overline{Y_k}.\mathbf{B}$, where $|\mathbf{A}| = |\mathbf{B}|$. Since the color conditions are trivial, we disregard the colors and reason about the erasures alone: if we assume that $\mathbf{A}$ contains an occurrence of Y_i (say at position p) with $\varphi(i) \neq \psi(i)$, then $\sigma(F)\overline{X^{\varphi(k)}} =_{\beta\eta} [\overline{X^{\varphi(k)}/Y^k}]\mathbf{A}$ and $\sigma(F)\overline{X^{\psi(k)}} =_{\beta\eta} [\overline{X^{\psi(k)}/Y^k}]\mathbf{A}$, so these differ at position p. This contradicts the assumption that σ is a unifier of $\mathcal{E}$, since this would entail that $\sigma(F)\overline{X^{\varphi(k)}} =_{\beta\eta} \sigma(F)\overline{X^{\psi(k)}}$. Thus $\mathbf{A}$ can only contain occurrences of Y^i where $\varphi(i) = \psi(i)$ and therefore is an instance of $H\overline{Y^{\rho(l)}}$ as in $\mathcal{UP}at(same)$.

Finally, for $\mathcal{UP}at(diff)$ we use a similar argumentation with $\sigma(F) =_{\beta\eta} \lambda\overline{Y_k}.\mathbf{A}$ and $\sigma(G) =_{\beta\eta} \lambda\overline{Z_l}.\mathbf{B}$. Since σ is a unifier of $\mathcal{E}$, we have

$$[\overline{X^{\varphi(k)}/Y^k}]\mathbf{A} =_{\beta\eta} \sigma(F)\overline{X^{\varphi(k)}} =_{\beta\eta} \sigma(G)\overline{X^{\psi(l)}} =_{\beta\eta} [\overline{X^{\psi(l)}/Z^l}]\mathbf{B}$$

so in particular if Y_i occurs in $\mathbf{A}$, then Z_j must occur in $\mathbf{B}$ at the same position for some i, j, where $\varphi(i) = \psi(j)$, and vice versa. If any other Y_i and Z_j would occur, then the terms would differ. Furthermore, $\mathbf{A}$ and $\mathbf{B}$ must be equal up to these differences. This concludes the proof of the assertion. □

Theorem 4.39 (Completeness for $\mathcal{UP}at$) *Let $\mathcal{E}$ be an unification problem, then $\mathcal{UP}at$ is terminating and yields an irreducible problem $\mathcal{F}$, such that either*

- *$\mathcal{F}$ is solved and $\sigma_{\mathcal{F}}$ is a most general unifier of $\mathcal{E}$ or*
- *$\mathcal{F}$ is not solved and $\mathcal{E}$ is not unifiable.*

Furthermore, $\mathcal{UP}at$ is confluent except for $\mathcal{CPT}$(flex/flex), which is finitely branching.

Proof: Since the erasures of all transformation rules involved are those for uncolored pattern unification, termination is a trivial consequence of the termination result for the uncolored case.

Confluence and completeness can be verified by a simple inspection of the rules. For $\mathcal{SIM}$ we have already verified confluence. Furthermore for higher-order patterns the $\mathcal{CUT}$(*flex*/*rig*) rule is deterministic, that is, all but the imitation or one projection immediately lead to failure. The rules $\mathcal{UP}at$(*diff*) and $\mathcal{UP}at$(*same*) have disjoint accessibility conditions and cannot lead to divergence. Finally, the rule $\mathcal{CPT}$(*flex*/*flex*) can diverge, as we have seen above, but since there are only finitely many projection (bounded by the length of the type), it must be finitely branching. $\square$

Now the completeness result above directly yields the fact that there can be at most one most general solution to a pattern unification problem.

Corollary 4.40 *Higher-order colored pattern unification is decidable and finitary, i.e. pattern unification problems have at most finitely many most general unifiers.*

5 Skeleton

In this section we will define and discuss the notion of a skeleton of a colored λ-term as it will be used to guide induction proofs. Analogously to the first order case we shall define the skeleton $\Omega_{\mathcal{D}}(\mathbf{A})$ of a colored term $\mathbf{A}$ with respect to a set $\mathcal{D} \subseteq \mathcal{C}$ of legal colors as a set of of colored λ-terms, where the illegal colors of the colored λ-term have been filtered out.

Originally in the first-order case, the skeleton of a colored term describes which symbols of specific colors (resp. color-variables) occur within a colored term and how these occurrences are related to each other wrt. the subterm-relation. Consider, for instance, the skeleton $\{s_c(a_c), s_c(b_c)\}$ wrt. $\{c\}$ of a colored term $s_c(plus_d(a_c, b_c)))$. In this case the skeleton encodes the information that a and b occur independently within the argument of s. In general one can prove that for first order terms, the skeletons of two colored terms are equal if the subterm-relations of the specifically colored occurrences of symbols coincide in both terms. Thus, the skeleton represents the relation between specifically colored occurrences of symbols wrt. the term construction. The next remarks give a small insight into the difficulties in lifting the notion of skeletons to a higher order setting.

While for first order terms, the different arguments of a function, e.g. *plus*, are considered to be independent wrt. the subterm-relation, there is an implicit dependency in a higher order setting. Consider for instance the first order term $plus_d(a_c, b_c)$ which is represented as $((plus_d a_c) b_c)$ in the λ-calculus (Currying). b_c denotes the argument to a function $(plus_d a_c)$. Thus, b_c is related (wrt. subterms) to a_c in contrast to the standard first order case. This fact has serious consequences on the possible definitions of skeletons. Given, for instance, two functions $g \in wff_{(\alpha\to\alpha)\to\alpha\to\alpha}$ and $h \in wff_{\alpha\to\alpha}$ consider the possible skeletons of $(g_d f_c)$ and f_c. From an first-order point of view both colored terms should agree on their skeleton, namely f_c. But then consider the terms $((g_d f_c) a_c)$. Corresponding to our example of *plus* we would expect to obtain $\{f_c, a_c\}$ as a skeleton while the latter example - combined with the restriction of a skeleton being a congruence on terms - suggests $\{(f_c a_c)\}$ to be the skeleton.

For practical reasons we prefer a solution which is compatible to the first-order case. The reason is that (up to now) the main application of the colored λ-calculus are first-order problems. Solving these problems by methods like middle-out reasoning [Hes91] involves higher-order variables which are used as meta-variables denoting non-specified wave-fronts or contexts. Thus, the terms under consideration are usually of base types in this area.

Thus in the example above, the skeleton $\Omega_{\mathcal{D}}((f_d t_1) t_2)$ should be the union of the skeletons of its arguments. Since $(f_d t_1)$ is a well-formed term and the skeleton has to denote a congruence on terms $\Omega_{\mathcal{D}}((f_d t_1) t_2)$ has to be a function of $\Omega_{\mathcal{D}}(f_d t_1)$ and $\Omega_{\mathcal{D}}(t_2)$ yielding $\Omega_{\mathcal{D}}(t_1) \cup \Omega_{\mathcal{D}}(t_2)$ as a result.

In the $\mathcal{C}$-calculus the symbol λ and all bound variables have no specific color annotation. This is due to the fact that λ-expressions closely relate to functions in the framework of functional programming and the β-rule is used to open up a function call. Thus, it seems to be useless to color the formal parameters since the color information will be inherited by the actual parameters and using β-reduction the formal parameters disappear. In addition, compatibility and subterm-stability together with β-reduction implies that a certain amount of information concerning λ-expressions have to be saved within the skeleton regardless whether λ occurs within a denoted context or skeleton. Consider the following example: given an equation $f_d = \lambda X.a_d$ we can manipulate $f_d a_c$ to $(\lambda X.a_d) a_c$ which reduces to a_d. Although the above equation contains no symbols colored by a variable or some color of $\mathcal{D}$, its application "destroys" the intended skeleton of the given term. Thus, f_d and $\lambda X.a_d$ must have different skeletons.

Computing the skeleton of a colored term we want to "clue" the occurrences of symbols of specific colors together regardless what types they have. Since we want to conserve well-typedness during the process of skelettification, we presuppose for any pair α, β of types the existence of syntactical type-conversion functions $f_\omega^{\alpha\beta}$ of type $\alpha \to \beta$. These functions do not carry any semantic status and are only used to conserve well-typedness, which is a technical convenience for the termination proofs. In some terms, occurrence of such constants are redundant, so we use the following equality theory to remove the redundancies.

Definition 5.1 (ω-Reduction ($\longrightarrow_\omega$)) We say that a well-formed formula **B** is obtained by a well-formed formula **A** by a one-step **ω-reduction** ($A \longrightarrow_\omega$), if it is obtained by applying one of the following rules to a well-formed part of **A**

- $f_\omega^{\beta\to\gamma}(f_\omega^{\alpha\to\beta}X_\alpha) \longrightarrow_\omega f_\omega^{\alpha\to\gamma}X_\alpha$
- $f_\omega^{\alpha\to\alpha}X_\alpha \longrightarrow_\omega X_\alpha$
- $\lambda x.x \longrightarrow_\omega f_\omega^{\alpha\to\alpha}$

As usual we define the transitive closure of the reduction relation $\longrightarrow_\omega$ with $\longrightarrow_\omega^*$ and we use $\longrightarrow_{\beta\eta\omega}$ for the union of the reduction relations $\longrightarrow_\beta$, $\longrightarrow_\eta$, and $\longrightarrow_\omega$.

As the next Lemmata will state, ω-reduction commutes with $\beta\eta$-equality and there exists a $\beta\eta\omega$-normal form $t\downarrow_{\beta\eta\omega}$ of a term t:

Lemma 5.2 (Strong Normalisation) *Every sequence of $\beta\eta\omega$-reductions is finite.*

Proof: The result is proven by the logical-relation method in the appendix. See theorem A.8 there. □

In order to prove the confluence of $\beta\eta\omega$-reduction we show that $\beta\eta\omega$-reduction is local confluent. Since $\beta\eta\omega$-reductions are finite it ensures that $\beta\eta\omega$-reduction is confluent.

Lemma 5.3 (Local Confluence) *If* $\mathbf{A} \longrightarrow_{\beta\eta\omega} \mathbf{A}'$ *and* $\mathbf{A} \longrightarrow_{\beta\eta\omega} \mathbf{A}''$ *then there must exit some term* **B** *such that* $\mathbf{A}' \longrightarrow_{\beta\eta\omega}^* \mathbf{B}$ *and* $\mathbf{A}'' \longrightarrow_{\beta\eta\omega}^* \mathbf{B}$.

Proof: Obviously both, $\longrightarrow_\omega^*$ and $\longrightarrow_{\beta\eta}^*$ satisfy the Church Rosser-property. Thus, in the following we have only to prove that in case $\mathbf{A} \longrightarrow_{\beta,\eta} \mathbf{A}'$ and $\mathbf{A} \longrightarrow_\omega \mathbf{A}''$ there is a term **B** with $\mathbf{A}' \longrightarrow_{\beta\eta\omega}^* \mathbf{B}$ and $\mathbf{A}'' \longrightarrow_{\beta\eta\omega}^* \mathbf{B}$. As in [Bar80] we use M[t] to denote a term with a specific subterm t inside a so-called context **M**. Let $\mathbf{A} \longrightarrow_\beta \mathbf{A}'$ then $\mathbf{A} = M[(\lambda X.\mathrm{C})\mathrm{C}']$ and $\mathbf{A}' = \mathrm{M}[[\mathrm{C}'/X]\mathrm{C}]$. In case a ω-rule has been applied inside **C** then obviously the same rule is also applicable in $[\mathrm{C}'/X]\mathrm{C}$. Analogously an application of a ω-rule on $\mathbf{C}'$ can be simulated by several applications of the same ω-rule on each occurrence of C' in $[\mathrm{C}'/X]\mathrm{C}$. On the other hand suppose the ω-rule has been applied inside the context **M** then the same ω-rule is also applicable in $\mathrm{M}[[\mathrm{C}'/X]\mathrm{C}]$. Analogous considerations can be made in case of the η-rule.

Hence, we may restrict ourselves to the following conflicts:

- let $\mathbf{A} = \mathrm{M}[\lambda X.(f_\omega^{\alpha\to\alpha}X)]$ and $\mathbf{A}' = \mathrm{M}[f_\omega^{\alpha\to\alpha}]$ and $\mathbf{A}'' = \mathrm{M}[\lambda X.X]$. With $\mathbf{B} = \mathbf{A}'$ $\mathbf{A}'' \longrightarrow_\omega \mathbf{B}$.
- let $\mathbf{A} = \mathrm{M}[((\lambda X.(f_\omega^{\alpha\to\alpha}\mathrm{C}))\mathrm{B})]$ and $\mathbf{A}' = \mathrm{M}[(f_\omega^{\alpha\to\alpha}[\mathrm{D}/X]\mathrm{C})]$ and $\mathbf{A}'' = \mathrm{M}[(\lambda X.\mathrm{C})\mathrm{D}]$ Let $\mathbf{B} = \mathrm{M}[[\mathrm{D}/X]\mathrm{C}]$ then $\mathbf{A}' \longrightarrow_\omega \mathbf{B}$ and $\mathbf{A}'' \longrightarrow_{\beta\eta} \mathbf{B}$.
- let $\mathbf{A} = \mathrm{M}[(\lambda X.X)\mathrm{D}]$ and $\mathbf{A}' = D$ and $\mathbf{A}'' = M[(f_\omega^{\alpha\to\alpha}B)]$. Let $\mathbf{B} = \mathbf{D}$ then $\mathbf{A}'' \longrightarrow_\omega \mathbf{B}$. □

Due to our previous comments we obtain the following lemma:

Lemma 5.4 (Church-Rosser Theorem) *If* $\mathbf{A} \longrightarrow^*_{\beta\eta\omega} \mathbf{A}'$ *and* $\mathbf{A} \longrightarrow^*_{\beta\eta\omega} \mathbf{A}''$ *then there must exit some term* $\mathbf{B}$ *such that* $\mathbf{A}' \longrightarrow^*_{\beta\eta\omega} \mathbf{B}$ *and* $\mathbf{A}'' \longrightarrow^*_{\beta\eta\omega} \mathbf{B}$.

Based on the remarks mentioned above we define the skeleton of a colored term as follows:

Definition 5.5 (Skeleton) Let $\mathcal{D} \subset \mathcal{C}$ be a set of colors, and $\Gamma_{\mathcal{D}}$ the context, such that $\Gamma^1_{\mathcal{D}} = \Gamma^1$ and $\Gamma^2_{\mathcal{D}}(X) = \Gamma^2(X) \cap (\mathcal{D} \cup \mathcal{X})$. Then we inductively define the **skeleton** $\Omega_{\mathcal{D}}$ by

- $\Omega_{\mathcal{D}}(h_a) = \begin{cases} \{h_a\} & \text{if } a \in \mathcal{D} \cup \mathcal{X} \text{ or } h \text{ is bound} \\ \{\lambda\overline{x_n}.f_\omega^{\alpha_1 \to \beta} x_1, \ldots, \lambda\overline{x_n}.f_\omega^{\alpha_n \to \beta} x_n\} & \text{if } h_a \in \mathit{wff}_{\overline{\alpha_n} \to \beta} \text{ else} \\ \emptyset & \text{if } h_a \in \mathit{wff}_\beta \end{cases}$
- $\Omega_{\mathcal{D}}(\lambda X.\mathbf{A}) = \{(\lambda X.\mathbf{A}') \downarrow_{\beta\eta\omega} | \mathbf{A}' \in \Omega_{\mathcal{D}}(\mathbf{A})\})$
- $\Omega_{\mathcal{D}}(\mathbf{AB}) = \{(\mathbf{A}'\mathbf{B}') \downarrow_{\beta\eta\omega} | \mathbf{A}' \in \Omega_{\mathcal{D}}(\mathbf{A}) \text{ and } \mathbf{B}' \in \Omega_{\mathcal{D}}(\mathbf{B})\}$

Example 5.6 Let a, p and s be constants of types α, $\alpha \to \beta$ and $\alpha \to \alpha$, let $\Gamma = [C{:}\alpha \to \alpha], [D{:}\alpha \to \beta \to \alpha], [X{:}\alpha]$. Furthermore let $\mathcal{D} = \{\mathtt{c}\}$, then the following holds:

$$\begin{array}{rcl} \Omega_{\mathcal{D}}(s_{\mathtt{d}}) & = & \{f_\omega^{\alpha \to \alpha}\} \\ \Omega_{\mathcal{D}}(s_{\mathtt{d}} X_{\mathtt{A}}) & = & \{X_{\mathtt{A}}\} \\ \Omega_{\mathcal{D}}(C_{\mathtt{A}} a_{\mathtt{c}}) & = & \{C_{\mathtt{A}} a_{\mathtt{c}}\} \\ \Omega_{\mathcal{D}}(\lambda X.C_{\mathtt{A}}(s_{\mathtt{d}} X)) & = & \{\lambda X.C_{\mathtt{A}} X\} \\ \omega_{\mathcal{D}}(\lambda X_\alpha.D_{\mathtt{d}}(s_{\mathtt{c}} X)(p_{\mathtt{A}} X)) & = & \{s_{\mathtt{c}}, \lambda X.f_\omega^{\alpha \to \beta}(p_{\mathtt{A}} X)\} \end{array}$$

Just as in the first-order case, the skeleton is stable with respect to subterm replacement:

Lemma 5.7 *For all* $\mathbf{A}, \mathbf{B} \in \mathit{wff}_\alpha(\Sigma; \Gamma_{\mathcal{Z}})$ $\Omega_{\mathcal{D}}(\mathbf{A}) = \Omega_{\mathcal{D}}(\mathbf{B}|_\pi)$ *implies* $\Omega_{\mathcal{D}}(\mathbf{B}) = \Omega_{\mathcal{D}}([\mathbf{A}/\pi]\mathbf{B})$.

Proof: This lemma is a trivial consequence from the definition of the skeleton function. □

Also, the skeleton is invariant wrt $\beta\eta$-reductions:

Lemma 5.8 *For all* $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma; \Gamma_{\mathcal{Z}}) : \mathbf{A} \longrightarrow_{\beta,\eta} \mathbf{B}$ *implies* $\Omega_{\mathcal{D}}(\mathbf{A}) \to_\alpha \Omega_{\mathcal{D}}(\mathbf{B})$

Proof:

$\longrightarrow_\eta$: Let $\mathbf{A} = \lambda X.(\mathbf{B}X)$:
$\Omega_{\mathcal{D}}(\mathbf{A}) = \Omega_{\mathcal{D}}(\lambda X.(\mathbf{B}X)) = \{(\lambda X.\mathbf{B}') \downarrow_{\beta\eta\omega} | \mathbf{B}' \in \Omega_{\mathcal{D}}(\mathbf{B}X)\}$
$= \{(\lambda X.(\mathbf{B}'X) \downarrow_{\beta\eta\omega}) \downarrow_{\beta\eta\omega} | \mathbf{B}' \in \Omega_{\mathcal{D}}(\mathbf{B})\} = \{(\lambda X.(\mathbf{B}'X)) \downarrow_{\beta\eta\omega} | \mathbf{B}' \in \Omega_{\mathcal{D}}(\mathbf{B})\}$
$= \{\mathbf{B}' \downarrow_{\beta\eta\omega} | \mathbf{B}' \in \Omega_{\mathcal{D}}(\mathbf{B})\} = \Omega_{\mathcal{D}}(\mathbf{B})$

$\longrightarrow_\beta$: Let $\mathbf{A} = ((\lambda X.\mathbf{C})\mathbf{D})$ and $\mathbf{B} = [\mathbf{D}/X]\mathbf{C}$
we first prove $\Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{C}) = \{([\mathbf{D}'/X]\mathbf{C}') \downarrow_{\beta\eta\omega} | \mathbf{C}' \in \Omega_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\}$ by induction:

- Let $\mathbf{C} = h_a$ and $X \neq \mathbf{C}$:
  $\Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{C}) = \Omega_{\mathcal{D}}(\mathbf{C}) = \{([\mathbf{D}'/X]\mathbf{C}') \downarrow_{\beta\eta\omega} | \mathbf{C}' \in \Omega_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\}$

- $\mathbf{C} = X$:
  $\Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{C}) = \Omega_{\mathcal{D}}(\mathbf{D}) = \{[\mathbf{D}'/X]X \mid \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\}$
  $= \{([\mathbf{D}'/X]X) \downarrow_{\beta\eta\omega} \mid \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} = \{([\mathbf{D}'/X]\mathbf{C}') \downarrow_{\beta\eta\omega} . \mid \mathbf{C}' \in \Omega_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\}$
- $\mathbf{C} = \lambda y.\mathbf{C}'$ and as induction hypothesis we assume:

$$\Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{C}') = \{[\mathbf{D}'/X]\mathbf{C}'' \mid \mathbf{C}'' \in \Omega_{\mathcal{D}}(\mathbf{C}'), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\}$$

$$\begin{aligned}
& \Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{C}) \\
= \ & \Omega_{\mathcal{D}}([\mathbf{D}/X](\lambda y.\mathbf{C}')) \\
= \ & \Omega_{\mathcal{D}}(\lambda y.[\mathbf{D}/X]\mathbf{C}') \\
= \ & \{(\lambda y.\mathbf{C}'') \downarrow_{\beta\eta\omega} \mid \mathbf{C}'' \in \Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{C}')\} \\
= \ & \{(\lambda y.([\mathbf{D}'/X]\mathbf{C}'') \downarrow_{\beta\eta\omega}) \downarrow_{\beta\eta\omega} \mid \mathbf{C}'' \in \Omega_{\mathcal{D}}(\mathbf{C}'), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \{([\mathbf{D}'/X](\lambda y.\mathbf{C}'')) \downarrow_{\beta\eta\omega} \mid \mathbf{C}'' \in \Omega_{\mathcal{D}}(\mathbf{C}'), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \{([\mathbf{D}'/X]\mathbf{C}') \downarrow_{\beta\eta\omega} \mid \mathbf{C}' \in \Omega_{\mathcal{D}}(\lambda y.\mathbf{C}'), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \{([\mathbf{D}'/X]\mathbf{C}') \downarrow_{\beta\eta\omega} \mid \mathbf{C}' \in \Omega_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\}
\end{aligned}$$

- $\mathbf{C} = (\mathbf{E}'\mathbf{F}')$ and as induction hypotheses we assume:

$$\begin{aligned}
\Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{E}) = \{([\mathbf{D}'/X]\mathbf{E}') \downarrow_{\beta\eta\omega} \mid \mathbf{E}' \in \Omega_{\mathcal{D}}(\mathbf{E}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
\Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{F}) = \{([\mathbf{D}'/X]\mathbf{F}') \downarrow_{\beta\eta\omega} \mid \mathbf{F}' \in \Omega_{\mathcal{D}}(\mathbf{F}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\}
\end{aligned}$$

$$\begin{aligned}
& \Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{C}) \\
= \ & \Omega_{\mathcal{D}}([\mathbf{D}/X](\mathbf{E}\mathbf{F})) \\
= \ & \Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{E}[\mathbf{D}/X]\mathbf{F}) \\
= \ & \{(\mathbf{E}'\mathbf{F}') \downarrow_{\beta\eta\omega} \mid \mathbf{E}' \in \Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{E}), \mathbf{F}' \in \Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{F})\} \\
= \ & \{(\mathbf{E}' \downarrow_{\beta\eta\omega} \mathbf{F}' \downarrow_{\beta\eta\omega}) \downarrow_{\beta\eta\omega} \mid \\
& \quad \mathbf{E}' \in [\mathbf{D}'/X]\Omega_{\mathcal{D}}(\mathbf{E}), \mathbf{F}' \in [\mathbf{D}'/X]\Omega_{\mathcal{D}}(\mathbf{F}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \{(\mathbf{E}'\mathbf{F}') \downarrow_{\beta\eta\omega} \mid \mathbf{E}' \in [\mathbf{D}'/X]\Omega_{\mathcal{D}}(\mathbf{E}), \mathbf{F}' \in [\mathbf{D}'/X]\Omega_{\mathcal{D}}(\mathbf{F}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \{([\mathbf{D}'/X](\mathbf{E}'\mathbf{F}')) \downarrow_{\beta\eta\omega} \mid \mathbf{E}' \in \Omega_{\mathcal{D}}(\mathbf{E}), \mathbf{F}' \in \Omega_{\mathcal{D}}(\mathbf{F}), \mathbf{D} \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \{([\mathbf{D}'/X]\Omega_{\mathcal{D}}(\mathbf{E}\mathbf{F})) \downarrow_{\beta\eta\omega} \mid \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\}
\end{aligned}$$

Thus, we know

$$\begin{aligned}
& \Omega_{\mathcal{D}}((\lambda X.\mathbf{C})\mathbf{D}) \\
= \ & \{(\mathbf{C}'\mathbf{D}') \downarrow_{\beta\eta\omega} \mid \mathbf{C}' \in \Omega_{\mathcal{D}}(\lambda X.\mathbf{C}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \{((\lambda X.\mathbf{C}'') \downarrow_{\beta\eta\omega} \mathbf{D}') \downarrow_{\beta\eta\omega} \mid \mathbf{C}'' \in \Omega_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \{([\mathbf{D}'/X]\mathbf{C}'') \downarrow_{\beta\eta\omega} \mid \mathbf{C}'' \in \Omega_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \Omega_{\mathcal{D}}(\mathbf{D})\} \\
= \ & \Omega_{\mathcal{D}}([\mathbf{D}/X]\mathbf{C})
\end{aligned}$$

holds. □

The skeleton does not have all nice properties it has in the first-order logic: In particular, it is not stable with respect to $\mathcal{C}$-substitutions. In first-order skeletons are substitution stable, i.e. if $\Omega_{\mathcal{D}}(s) = \Omega_{\mathcal{D}}(t)$ then $\Omega_{\mathcal{D}}(\sigma(s)) = \Omega_{\mathcal{D}}(\sigma(t))$ holds for all s, t and substitutions σ. In case of the lambda calculus any meaningful definition of a skeleton will violate this restriction. Consider, for example, the terms $F_{\mathrm{d}}a_{\mathrm{c}}b_{\mathrm{d}}$ and $f_{\mathrm{d}}a_{\mathrm{c}}b_{\mathrm{d}}$ which both coincide in their skeletons like a_{c}. Instantiating F_{d} by $\lambda X.(\lambda Y.Y)$ will (after β-reduction) result in terms b_{d} and $f_{\mathrm{d}}a_{\mathrm{c}}b_{\mathrm{d}}$ which do obviously not coincide in their skeleton. The reason is that the instantiation enables the use of the β-rule which then, deletes parts of the skeleton. Two possibilities to get rid of this problem immediately suggest themselves: adding function variables (regardless of their annotations) always to the skeleton or restricting admissible substitutions in order to avoid these substitutions. However both will be to restrictive for practical reasons. Thus, the skeleton as defined in 5.5 is in general not substitution-stable if some variable of a non-base type is affected by the substitution. This is no major drawback in using a $\mathcal{C}$-calculus for deduction. For instance in case of instantiating a $\mathcal{C}$-equation we have only to test whether the skeletons of both instantiated sides still coincides.

6 Conclusion

We have extended the first-order rippling/coloring method to higher-order logic and present unification, pre-unification and pattern unification algorithms that we prove correct and complete. Thus we have provided a formal basis to the implementation of rippling in a higher-order setting which is required e.g. in case of middle-out reasoning [Hes91, IB96] and also a logical basis for an interface for linguistic extra-semantical information in the construction of natural-language semantics [GK96].

Furthermore, the work presented in this paper provides a starting point for the mechanization of higher-order reasoning with equality along the lines of [WNB92, CH94] which develop heuristics that guide the difference reduction process in first-order equality calculi such as [Mor69, Dig81]. These difference reducing approaches seem to be more promising for higher-order logic, since they can reduce the search spaces (comparing to those induced by encoding equality via the Leibniz formula) without needing reduction orderings, which become very weak in the presence of higher-order (function)-variables.

¿From an abstract point of view, the coloring technique allows adding annotations to symbol occurrences in λ-terms. Thus in contrast to other semantic annotation techniques like sorts, it is possible to encode syntactic information and use that to guide inferencing processes. In this sense coloring is orthogonal to the introduction of sorts, and the combination of the two mechanisms would an interesting subject for further investigation, since it is reasonable to expect a multiplication of the filtering effects provided by both methods in automated deduction.

References

[Bar80] Hendrik P. Barendregt. *The Lambda-Calculus: Its Syntax and Semantics.* North-Holland, 1980.

[BSvH+93] Alan Bundy, Andrew Stevens, Frank van Harmelen, Andrew Ireland, and Alan Smaill. Rippling: a heuristic for guiding inductive proofs. *AI*, 62:185–253, 1993.

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings of the 9th Conference on Automated Deduction*, number 310 in LNCS, pages 111–120, Argonne, Illinois, USA, 1988.

[CH94] J. Cleve and D. Hutter. A methodology for equational reasoning. In jr. Eds : Jay F. Nunamaker and jr. Ralph H. Sprague, editors, *Proceedings Hawaii International Conference on System Sciences 27 Volume III: Information Systems : DSS/Knowledge-based Systems*, pages 569 – 578. IEEE Computer Society Press, Los Alamitos, California, 1994.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Dig81] Vincent J. Digricoli. The efficacy of RUE resolution, experimental results and heuristic theory. In Ann Drinan, editor, *Proceedings of the 7th International Joint Conference on Artificial Intelligence (ICJAI)*, pages 539–547, Vancouver, Canada, 1981. Morgan Kaufmann, San Mateo, California, USA.

[Dow92] Gilles Dowek. Third order matching is decidable. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science (LICS-7)*, pages 2–10. IEEE Computer Society Press, 1992.

[DSP91] M. Dalrymple, S. Shieber, and F. Pereira. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14:399–452, 1991.

[GK96] Claire Gardent and Michael Kohlhase. Higher-order coloured unification and natural language semantics. Submitted to COLING'96, 1996.

[Gol81] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[Hes91] Jane Hesketh. *Using Middle-Out Reasoning to Guide Induction.* PhD thesis, University of Edinburgh, 1991.

[HS86] J. Hindley and J. Seldin. *Introduction to Combinators and Lambda Calculus.* Cambridge University Press, 1986.

[Hut90] Dieter Hutter. Guiding induction proofs. In Mark Stickel, editor, *Proceedings of the 10th Conference on Automated Deduction*, number 449 in LNCS, pages 147–161, Kaiserslautern, Germany, 1990.

[Hut91] Dieter Hutter. *Mustergesteuerte Strategien zum Beweisen von Gleichheiten.* PhD thesis, Universität Karlsruhe, Karlsruhe, 1991.

[IB96] Andrew Ireland and Alan Bundy. Productive use of failure in inductive proof. *Special Issue of the Journal of Automated Reasoning*, to appear, 1996.

[Koh94] Michael Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle.* PhD thesis, Universität des Saarlandes, 1994.

[Kra94] Ina Kraan. *Proof Planning for Logic Program Synthesis.* PhD thesis, University of Edinburgh, 1994.

[Mil92] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14:321–358, 1992.

[Mon74] R. Montague. The proper treatment of quantification in ordinary english. In R. Montague, editor, *Formal Philosophy. Selected Papers.* Yale University Press, New Haven, 1974.

[Mor69] James B. Morris. E-resolution: Extension of resolution to include the equality relation. In Donald E. Walker and Lewis Norton, editors, *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 287–294, 1969.

[Sny91] Wayne Snyder. *A Proof Theory for General Unification.* Progress in Computer Science and Applied Logic. Birkhäuser, 1991.

[Sta85] R. Statman. Logical relations and the typed lambda calculus. *Information and Computation*, 65, 1985.

[Tai67] W. Tait. Intensional interpretation of functionals of finite type I. *Information and Computation*, 32:198–212, 1967.

[WNB92] Toby Walsh, A. Nunes, and Alan Bundy. The use of proof plans to sum series. In D. Kapur, editor, *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, pages 325–339, Saratoga Springs, NY, USA, 1992. Springer Verlag.

A Termination of $\beta\eta\omega$-Reduction

In this appendix we will prove termination of the $\beta\eta\omega$-Reduction used in the definition of a Skeleton using the well-known logical-relations method due to Tait [Tai67] and Statman [Sta85]

Definition A.1 (Logical Relation) A unary typed relation $\mathcal{L} \subset \mathit{wff}_\alpha(\Sigma; \Gamma_{\mathcal{Z}})$ is called a **logical relation**, iff for all types $\alpha = \gamma \to \delta$ and all $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma; \Gamma_{\mathcal{Z}})$ we have $\mathcal{L}_\alpha(\mathbf{A})$, iff $\mathcal{L}_\delta(\mathbf{AC})$ for all $\mathbf{C} \in \mathit{wff}_\gamma$ with $\mathcal{L}_\gamma(\mathbf{C})$. Clearly, logical relations are totally determined by their values on base types. For any other unary typed relation $\mathcal{S}$, we call the (unique) logical relation $\mathcal{L}$ that coincides with $\mathcal{S}$ on base types the **logical relation induced by** $\mathcal{S}$.

Definition A.2 (Admissible) Let $\mathcal{L}$ be a logical relation, then a head reduction

$$(\lambda X_\alpha.\mathbf{A}_\alpha)\mathbf{B} \longrightarrow_\beta [\mathbf{B}/X]\mathbf{A}$$

is called **admissible**, iff $\mathcal{L}_\beta(\mathbf{B})$. A logical Relation R is said to be **admissible** if $\mathcal{L}_0$ is closed under admissible head expansions. (Formally: Let α be a base type, and let $A_\alpha, \mathbf{B}_\alpha$ be terms with $\mathbf{A} \longrightarrow^h \mathbf{B}$, then a logical Relation $\mathcal{L}$ is called **admissible** if $\mathcal{L}_\alpha(\mathbf{B})$ implies that $\mathcal{L}_\alpha(\mathbf{A})$.)

The key tool for the logical relation method is the fundamental theorem for logical relations. A proof can be found in [Sta85].

Theorem A.3 (Fundamental Theorem for Logical Relations) *If $\mathcal{L}$ is admissible and $\mathbf{A}$ in $\mathit{wff}_\alpha(\Sigma; \Gamma_{\mathcal{Z}})$, then $\mathcal{L}^*_\alpha(\mathbf{A})$.*

Definition A.4 (Terminating at A)
We say that $\beta\eta\omega$-reduction is **terminating at A**, iff any sequence of $\beta\eta\omega$-reductions starting with $\mathbf{A}$ terminates. Let $\mathcal{S}$ be the typed relation, such that $\mathcal{S}_\alpha(\mathbf{A})$, iff $\beta\eta\omega$-reduction is terminating at $\mathbf{A}$. Furthermore let $\mathcal{L}$ be the logical relation induced by $\mathcal{S}$

Lemma A.5 *If $\beta\eta\omega$-reduction is terminating at* $\mathbf{C}$, *and* $\mathbf{B}$ *is a subformula of* $\mathbf{C}$, *then $\beta\eta\omega$-reduction is terminating at* $\mathbf{B}$

Proof: Any infinite reduction sequence from $\mathbf{B}$ can be transformed to one from $\mathbf{C}$. □

The proof of termination will proceed in two steps: We will first show that $\mathcal{L} \subseteq \mathcal{S}$ and then that $\mathcal{L}$ is admissible. Combining these results with the fundamental theorem, we obtain that $\mathcal{L}_\alpha(\mathbf{A})$ for any formula $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma; \Gamma_{\mathcal{Z}})$ and therefore $\mathcal{S}_\alpha(\mathbf{A})$, which implies universal termination of $\beta\eta\omega$-reduction.

Lemma A.6 ($\mathcal{L} \subseteq \mathcal{S}$)

1. *Let h be a constant or variable of type $\alpha = \overline{\alpha^n} \to \beta$ and $\mathcal{S}_{\alpha^i}(\mathbf{A}^i)$, then $\mathcal{L}_\beta(h\overline{\mathbf{A}^n})$.*
2. *$\mathcal{L}_\alpha(\mathbf{A})$ implies $\mathcal{S}_\alpha(\mathbf{A})$.*

Proof: We prove the assertion by a simultaneous induction over the type α. If α is a base type, the second assertion is a trivial consequence of the definition of $\mathcal{L}$. For the first assertion we have to consider two cases: If $h \neq f_\omega^{\alpha \to \beta}$, then we obtain the fact that $\beta\eta\omega$-reduction is terminating at $h\overline{\mathbf{A}^n}$, since the $\mathbf{A}^i$ are (since the arguments of h are independent, there can be no $\beta\eta\omega$-redexes that are not in the $\mathbf{A}_i$), thus $\mathcal{L}_\beta(h\overline{\mathbf{A}^n})$ since $\alpha \in \mathcal{BT}$ ($\mathcal{S} = \mathcal{L}$ there).

If $h = f_\omega^{\alpha\to\beta}$, then

$$h\overline{\mathbf{A}^n} = (f_\omega(f_\omega(\dots(f_\omega\mathbf{B})\dots))\mathbf{A}^2\dots\mathbf{A}^n)$$

where the head of $\mathbf{B}$ is not f_ω. We show the assertion by an induction over the number n, using the previous case for $n = 0$. For the inductive case we note that the first two ω-rules transform $h\overline{\mathbf{A}^n}$ into a formula of the same form with reduced n and the third rule cannot apply at the top-level of $h\mathbf{A}^j$ at all. Thus we have completed the proof of the first assertion in the case type case.

For the inductive case let $\alpha = \beta \to \gamma$, and $\mathcal{L}_\beta(\mathbf{B})$. By the second inductive hypothesis we have $\mathcal{S}_\beta(\mathbf{B})$, by the first inductive hypothesis $\mathcal{L}_\gamma(h\overline{\mathbf{A}^n}\mathbf{B})$. Thus $\mathcal{L}_\beta(h\overline{\mathbf{A}^n})$ by definition and we have proven the first assertion.

For the second assertion let $\mathcal{L}_\alpha(\mathbf{A})$ and $X_\beta \notin \mathbf{free}(\mathbf{A})$. With the first inductive hypothesis ($n = 0$) we have $\mathcal{L}_\beta(X)$, and thus $\mathcal{L}_\gamma(\mathbf{A}X)$ by definition. Now we see that $\beta\eta\omega$-reduction is terminating at $\mathbf{A}$, since by the second inductive hypothesis we have $\mathcal{S}_\gamma(\mathbf{A}X)$ and $\mathcal{S}_\alpha(\mathbf{A})$ by A.5. □

Lemma A.7 *$\mathcal{L}$ is admissible.*

Proof: For the case of the unary relation $\mathcal{L}$ we have to show that $\mathcal{L}_\alpha([\mathbf{B}/X]\mathbf{A})$ implies $\mathcal{L}_\alpha((\lambda X_\beta.\mathbf{A})\mathbf{B})$. Now let $\alpha = \overline{\gamma} \to \delta$, such that $\delta \in \mathcal{BT}$, furthermore let $\mathcal{L}_\alpha(\mathbf{A})$, and $\mathcal{L}_{\gamma^i}(\mathbf{C}^i)$. Then (by an iterated application of the definition of $\mathcal{L}$) it is sufficient to show that $\mathcal{S}_\delta((\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}})$ ($\beta\eta\omega$-reduction is terminating at $(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$), since δ is a base type.

So let us assume that $\mathcal{L}_\alpha([\mathbf{B}/X]\mathbf{A})$, then by definition of $\mathcal{L}$ we have $\mathcal{L}_\delta([\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}})$ and thus $\mathcal{S}_\delta([\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}})$. In particular $\mathcal{S}_\alpha([\mathbf{B}/X]\mathbf{A})$ and $\mathcal{S}_{\gamma^i}(\mathbf{C}^i)$ by A.5. Furthermore the head reduction is admissible and therefore $\mathcal{L}_\beta(\mathbf{B})$ and thus $\mathcal{S}_\beta(\mathbf{B})$ by A.6. Finally, $\beta\eta\omega$-reduction must be terminating at $\mathbf{A}$, since an infinite reduction from $\mathbf{A}$ would imply one from $[\mathbf{B}/X]\mathbf{A}$ ($\beta\eta\omega$-reduction is invariant under instantiation). Thus there cannot be an infinite sequence of reductions from $(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ that only contracts redexes from $[\mathbf{B}/X]\mathbf{A}$ and the $\mathbf{C}^i$. Thus such a reduction sequence from $(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ has the form

$$\begin{aligned}(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}} &\longrightarrow^*_{\beta\eta\omega} (\lambda X.\mathbf{A}')\mathbf{B}'\overline{\mathbf{C}'}\\ &\longrightarrow_{\beta\eta\omega} [\mathbf{B}'/X]\mathbf{A}'\overline{\mathbf{C}'}\\ &\longrightarrow^*_{\beta\eta\omega} \cdots\end{aligned}$$

where $\mathbf{A} \longrightarrow^*_\beta \mathbf{A}'$, $\mathbf{B} \longrightarrow^*_\beta \mathbf{B}'$ and $\mathbf{C}^i \longrightarrow^*_\beta \mathbf{C}'^i$. Thus $[\mathbf{B}/X]\mathbf{A} \longrightarrow^*_\beta [\mathbf{B}'/X]\mathbf{A}'$ and in particular (in contradiction to our assumption), we have constructed an infinite reduction

$$\begin{aligned}[\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}} &\longrightarrow^*_{\beta\eta\omega} [\mathbf{B}'/X]\mathbf{A}'\overline{\mathbf{C}'}\\ &\longrightarrow^*_{\beta\eta\omega} \cdots\end{aligned}$$

□

Corollary A.8 (Termination) *$\beta\eta\omega$-Reduction is terminating.*

Proof: Let $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma;\Gamma_\mathcal{Z})$ be an arbitrary formula, by A.7 and the fundamental theorem A.3 we have $\mathcal{L}_\alpha(\mathbf{A})$ and thus $\mathcal{S}_\alpha(\mathbf{A})$ by A.6. □