Managing Structural Information by Higher-Order Colored Unification

Dieter Hutter
German Research Center for Artificial Intelligence,
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany,
E-mail: hutter@dfki.de

Michael Kohlhase
FB Informatik, Universität des Saarlandes,
D-66041 Saarbrücken, Germany,
E-mail: kohlhase@cs.uni-sb.de

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 symbol occurrences in 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 λ calculus with color annotations and present appropriate algorithms for the general, pre- and pattern unification problems. 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.

Keywords: Inductive Theorem Proving, Rippling, Annotations

1. Introduction

In the field of inductive theorem proving syntactical differences between the induction hypothesis and induction conclusion are used in order to guide proofs (cf. [5, 4], or [19, 21]). 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 terms 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 by an inference technique called difference unification [2]. These formulae 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/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 one to reduce the differences between conclusion and hypothesis in a goal directed way and thus leads to a situation, where the inductive hypothesis can be applied.

This paper extends the coloring method to higher-order logic and presents algorithms for enumerating higher-order colored unifiers and pre-unifiers and prove them correct and complete. For the fragment of higher-order patterns, we show that decidability is maintained for the colored case, while uniqueness of solutions is lost.

Thus our work provides a formal basis for the implementation of rippling in a higher-order setting which is required e.g. in case of middle-out reasoning [17] or generalization of theorems using proof critics [23]. In the latter the unknown generalized version of a formula is described by a schematic formula containing parts of the original formula and higher-order variables denoting the unknown syntactical extensions of it. In the process of 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 method allows one to add arbitrary information to *occurrences* of (λ-)terms and to inherit this information during the inference process. This differentiates coloring from other semantic annotation techniques like sorts which maintain attributes of symbols but not attributes of single symbol *occurrences*. Hence, coloring techniques can be generally used to maintain for instance initial knowledge about an internal structure of a term.

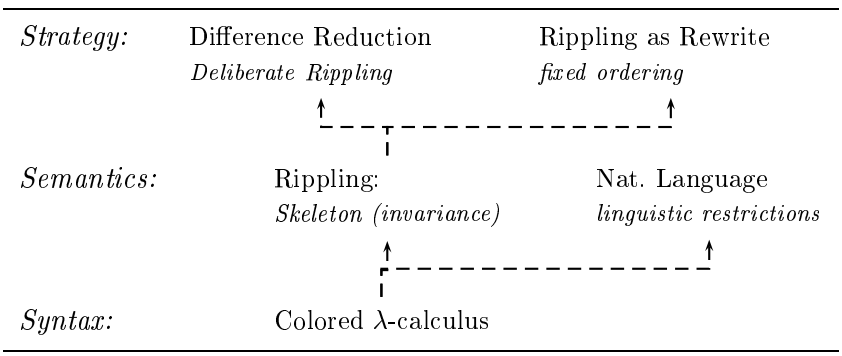

Figure 1. Conceptual Hierarchy

The colored λ-calculus presented in this paper is a general procedure to

control this kind of information. In addition, it can be easily enlarged to deal with more complex knowledge by enriching the representation formalism for colors to formulate appropriate annotations (see [14] for a linguistic application that uses feature terms as colors).

The flexibility of our approach is mirrored in the wide range of possible applications (cf. figure 1). In this paper we present two areas of them. The first area is concerned with "classical" rippling as it is used in inductive theorem proving while in the latter we use the presented calculus for a semantic analysis of natural language. The domain specific semantics is represented in the different *interpretations* of the annotated colors. For example in case of rippling, the so-called *skeleton* of an annotated term denotes the invariant of a proof of an induction step, while in the natural language example the colors are used to encode the so-called *primary occurrence restriction*.

In the rest of this section we will briefly sketch some applications of the higher-order coloring method, informally introducing the relevant notions as we go along, before we turn to a formal definition of the colored λ-calculus in section 2. The next part of the paper is devoted to the formal definition of a general-, pre- and pattern-unification procedure operating on this calculus in section 3. The description of skeletons in a higher-order setting in given in section 4, which also illustrates more general aspects of the specific solution we have chosen.

1.1. Inductive Proofs

Rippling was developed for proving theorems by induction and has been applied to a large number of practical examples from this domain [5, 4, 19, 21]. It is based on the 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 formulae are colored "grey". Also left- and right-hand sides of given equations are *difference unified* in a sense that the common structure of both terms of a given equation is annotated by color variables while differences are colored grey. The grey parts are called *wave-fronts* while the non-grey parts denote the *skeleton*.

In [4, 2] an ordering, which evaluates the position and size of the wave-fronts within the skeleton, is used to build up a rewrite system on annotated terms. Each application of an annotated rewrite rule (so-called *wave-rule*) results in a term which is less (wrt. the given ordering) than the original one. In recent years the technique of rippling has been

applied also to non-inductive theorem proving yielding a more complex planning of the rippling process than in inductive proofs. Possible target positions of wave-fronts are no longer statically predefined but have to be planned during the proof which gives rise to a kind of *deliberate* rippling [20, 22].

Rippling restricts the search space for inductive theorem proving by forbidding all deduction steps which do not preserve the skeleton, i.e. do not change the non-grey[1] parts of the formula, and only applies those that move the difference out of the way leaving behind the skeleton. In their simplest form, these equations to be used are of the form $f(g(t)) = h(f(t))$. By design, the skeleton $f(t)$ 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/coloring techniques.

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

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

We formalize summation by a binary function *sum* that takes a function (that is summed over) and an upper bound as arguments. Furthermore, we will use the following definition of *sum* (let f, g, H be of type[2] $nat \to nat$ and N, n be of type nat):

$$\forall H \,.\, sum(H, 0) = 0 \tag{1}$$

$$\forall H, N \,.\, sum(H, s(N)) = sum(H, N) + H(s(N)) \tag{2}$$

Then our theorem takes the form

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

[1] 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.

[2] Since for the purposes of this paper types largely play a theoretical role (they for instance make $\beta\eta$-reduction terminating and therefore $\beta\eta$-equality decidable), we will only specify them where needed.

To prove this, simple heuristics employed by most inductive provers suggest induction on n which results in the following step case[3].

$$sum(f,n) + sum(g,n) = sum(\lambda Z \,.\, f(Z) + g(Z), n)$$
$$\rightarrow sum(f, s(n)) + sum(g, s(n)) = sum(\lambda Z \,.\, f(Z) + g(Z), s(n))$$

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

$$sum(f,n) + sum(g,n) = sum(\lambda Z \,.\, f(Z) + g(Z), n) \qquad (3)$$
$$\rightarrow sum(f, s(n)) + sum(g, s(n)) = sum(\lambda Z \,.\, f(Z) + g(Z), s(n))$$

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 lemmata that preserve the skeleton of the term being rewritten. Corresponding to the recursive definitions for *sum* we have the following annotated equation of (2).

$$sum(H, s(n)) = sum(H, N) + H(s(N)) \qquad (4)$$

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. To simplify the conclusion of (3) by rippling we apply (4) on both sides[4] yielding the modified conclusion:

$$(sum(f,n) + f(s(n))) + (sum(g,n) + g(s(n)))$$
$$= sum(\lambda Z \,.\, f(Z) + g(Z), n) + (f(s(n)) + g(s(n)))$$

Applying associativity and commutativity law of + results in

$$((sum(f,n) + sum(g,n)) + f(s(n)) + g(s(n)))$$
$$= (sum(\lambda Z \,.\, f(Z) + g(Z), n) + (f(s(n)) + g(s(n))))$$

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

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

[4] In this case the unification of higher-order formulae is nearly trivial, binding N to n and H to f, g, and $\lambda Z \,.\, f(Z) + g(Z)$ respectively.

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

1.2. 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, where appropriate equations are missing, Ireland & Bundy [23] propose 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. For a discussion of their formal approach to unification see section 5.

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

$$\forall u, v \,.\, rev(app(rev(v), u)) = app(rev(u), v) \tag{5}$$

Here u and v are of type γ for lists and *rev* and *app* stand for the operations of reversing and concatenating lists, they have types $\gamma \to \gamma$ and $\gamma \to \gamma \to \gamma$ respectively. Using induction on v we obtain the following formula as an induction conclusion in the step case (h is a new element constant of type ϵ and *cons* the list constructor of type $\epsilon \to \lambda \to \lambda$):

$$rev(app(rev(cons(h, v)), u)) = app(rev(u), cons(h, v)) \tag{6}$$

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

$$rev(app(app(rev(v), cons(h, nil)), u)) = app(rev(u), cons(h, v)) \tag{7}$$

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

$$app(X, cons(Y, Z)) = app(F_1(X, Y, Z), Z) \tag{8}$$

which can be used to move the blocked wave-front on the right-hand side towards the sink[7] u. 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:

$$rev(app(app(rev(v), cons(h, nil)), u)) = app(F_1(rev(u), h, v), v) \tag{9}$$

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

[7] Universally quantified variables are called sinks in rippling, because they can be used to swallow up wave front material, since they can be arbitrarily instantiated.

To enable the use of the induction hypothesis in this example the wave front has to be moved in front of the sink u. Thus, we use the annotated equation

$$app(rev(Y), cons(X, nil)) = rev(cons(X, Y)) \tag{10}$$

in order to ripple the wave-front on the right-hand side towards u. In order for (10) to be applicable to (9), we must unify[8] $F_1(rev(u), h, v)$ and $app(rev(Y), cons(X, nil))$. Higher-order colored unification, or HOCU for short, results in a solution (see the example in section 1.5 for a trace of the computation)

$$[\lambda UVW.\, app(U, cons(V, nil))/F_1], [h/X], [u/Y]. \tag{11}$$

Applying the instance of (10) under (11) to the right-hand side of (9) the wave-front is moved towards the sink u:

$$rev(app(app(rev(v), cons(h, nil)), u)) = app(rev(cons(h, u)), v) \tag{12}$$

The unifier used to perform this step also refines the schema of the speculated annotated equation (8) which we have previously used to unblock the rippling process, to

$$app(X, cons(Y, Z)) = app(app(X, cons(Y, nil)), Z) \tag{13}$$

The aopplication of this speculated equation (13) on the left-hand side finally yields:

$$rev(app(rev(v), cons(h, u))) = app(rev(cons(h, u)), v)$$

which enables the use of the induction hypothesis and completes this particular proof. Proving by induction the speculated lemma (13), which is the instance of the speculated equation (7) using (11), finishes the overall proof.

1.3. A Colored λ-calculus

Before we turn to the linguistic application, let us informally introduce some notation and generalize[9] the set of colors from "grey" and

[8] To ease readability we have slightly simplified the method of Ireland and Bundy; Actually, the occurrence of the meta-variable $F_1(rev(u), h, v)$ is replaced by a nested term $F_2(F_1(rev(u), h, v), h, v)$ in order to allow the speculation of more complex wave-fronts using F_2 in the later rippling process. In our example it would only be instantiated to the projection $\lambda UVW.U$.

[9] This generalization does not make the theory more complicated, in the contrary, it makes the concepts involved much clearer, and it allows to treat more applications (see [14] for a linguistic application that uses feature terms as colors).

"white" to an arbitrary set of colors (we will make this totally formal in section 2). 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 [6] (see [1, 18] for an introduction), where occurrences of constants and free variable can be annotated with so-called **colors** which are either **color constants** $\mathcal{C} = \{\mathtt{a}, \mathtt{b}, \ldots\}$ or **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. We call a formula $\mathbf{M}$ **c-monochrome** (definition 2), if all symbols (except bound variables) in $\mathbf{M}$ are annotated by a common color $\mathtt{c}$.

It is crucial for our logical 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. $h_{\mathtt{c}} X_{\mathtt{a}} X_{\mathtt{b}}$) and that symbols that carry different colors are treated differently. This observation leads to the notion of colored substitutions[10] (definition 4), 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 $X_{\mathtt{c}}$ 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 $\mathtt{A}$ and $\mathtt{B}$ are different colors, then $|\sigma(X_{\mathtt{A}})| = |\sigma(X_{\mathtt{B}})|$, where $|\mathbf{M}|$ is the **color erasure** of $\mathbf{M}$, i.e. the formula obtained from $\mathbf{M}$ by erasing all color annotations in $\mathbf{M}$ (definition 3).
- If $\mathtt{c} \in \mathcal{C}$ is a color constant, then $\sigma(X_{\mathtt{c}})$ is $\mathtt{c}$-monochrome.

The first condition ensures that the color erasure of a $\mathcal{C}$-substitution (defined in the obvious manner) 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.

Note that since bound variables do not carry color information, $\beta\eta$-reduction (cf. definition 28) in the colored λ-calculus is just the classical notion and inherits its good properties (decidability and normalization).

The constraints on $\mathcal{C}$-substitutions given above allow us to specialize higher-order unification to an inference procedure that manages color information: a higher-order unifier σ of a given equation $\mathbf{M} = \mathbf{N}$

[10] We will denote the substitution of a term $\mathbf{N}$ for all free occurrences of X in $\mathbf{M}$ with $[\mathbf{N}/X]\mathbf{M}$.

(i.e. $\sigma(\mathbf{M}) =_{\beta\eta} \sigma(\mathbf{N})$) must be a $\mathcal{C}$-substitution in order to be a colored higher-order unifier (definition 7) of $\mathbf{M}$ and $\mathbf{N}$. In particular, $\mathcal{C}$-unification will only succeed if parallel sub-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.

It is well-known, that in first-order logic 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 individuals only. In fact there might be even solvable equations that have infinite chains of unifiers which are more and more general. In other words most general unifiers need not to exist in general.

1.4. Higher–Order Unification and Natural Language semantics

In this section we will present a different kind of application of higher-order colored unification in the area of natural language semantics. In [11, 13, 12, 15] 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 briefly sketch the underlying ideas in case of verb-phrase ellipsis; i.e. the phenomenon that parts of natural language sentences (here verb phrases) can be replaced by utterances like "does too". For a thorough treatment of cases like focus constructions, second-occurrence expressions, and adverbial quantification, the reader is refered to [11] and the references in [15].

The basic idea [7] underlying the use of higher-order unification for natural language semantics is very simple: Following [28], the simply typed λ–calculus is used as a semantic representation language while semantically under-specified elements (e.g. anaphoric references or ellipses) are represented by free variables whose value is determined by solving higher-order equations. For instance, the discourse (14) has has the semantic representation (15), where the value of the predicate variable R is determined by equation (16).

$$\textit{Dan likes his wife. Peter does too} \tag{14}$$

$$\mathit{like}(\mathit{dan}, \mathit{w\_of}(\mathit{dan})) \wedge R(\mathit{peter}) \tag{15}$$

$$\mathit{like}(\mathit{dan}, \mathit{w\_of}(\mathit{dan})) =^{t} R(\mathit{dan}) \tag{16}$$

Higher-order unification calculates the solutions (17) and (18).

$$[\lambda Z \,.\, \mathit{like}(Z, \mathit{w\_of}(\mathit{dan}))/R] \quad \text{and} \quad [\lambda Z \,.\, \mathit{like}(Z, \mathit{w\_of}(Z))/R] \tag{17}$$

$$[\lambda Z \,.\, \mathit{like}(\mathit{dan}, \mathit{w\_of}(\mathit{dan}))/R] \quad \text{and} \quad [\lambda Z \,.\, \mathit{like}(\mathit{dan}, \mathit{w\_of}(Z))/R] \tag{18}$$

However, only the first two of the solutions (17) lead to the linguistically desired solutions (19) and (20), whereas those in (18) lead to (21) and (22) which are clearly not the desired readings of the discourse.

$$like(dan, w\_of(dan)) \quad \wedge \quad like(peter, w\_of(dan)) \tag{19}$$
$$like(dan, w\_of(dan)) \quad \wedge \quad like(peter, w\_of(peter)) \tag{20}$$
$$like(dan, w\_of(dan)) \quad \wedge \quad like(dan, w\_of(dan)) \tag{21}$$
$$like(dan, w\_of(dan)) \quad \wedge \quad like(dan, w\_of(peter)) \tag{22}$$

To remedy this shortcoming, Dalrymple & Shieber & Pereira, who have pioneered this analysis in [7], propose an informal restriction, the **primary occurrence restriction**, which deletes any solution from the set of linguistically valid solutions, 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 modeled as follows: Primary occurrences are $\mathtt{p}$-colored while free variables are $\mathtt{s}$-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 $\mathtt{p}$-colored symbol) as it was required by the primary occurrence restriction. For instance, the colored representation of (14) is (23) together with the colored unification problem (24) which only has the $\mathcal{C}$-unifiers in (25). Higher-Order Unification without the monotonicity constraint would have the solutions (26) which are not well-colored.

$$like(dan_{\mathtt{p}}, w\_of(dan)) \wedge R_{\mathtt{s}}(peter) \tag{23}$$
$$like(dan_{\mathtt{p}}, w\_of(dan)) =^{t} R_{\mathtt{s}}(dan_{\mathtt{p}}) \tag{24}$$
$$[\lambda Z\,.\,like_{\mathtt{s}}(Z, w\_of_{\mathtt{s}}(dan_{\mathtt{s}}))/R_{\mathtt{s}}] \text{ and } [\lambda Z\,.\,like_{\mathtt{s}}(Z, w\_of_{\mathtt{s}}(Z))/R_{\mathtt{s}}] \tag{25}$$
$$[\lambda Z\,.\,like(dan_{\mathtt{p}}, w\_of(dan))/R_{\mathtt{s}}] \text{ and } [\lambda Z\,.\,like(dan_{\mathtt{p}}, w\_of(\mathbf{Z}))/R_{\mathtt{s}}] \tag{26}$$

Note that the analysis hinges on the fact that colors (unlike types or sorts) provide a means to distinguish between symbol occurrences (in our example the different occurrences of *dan*) by annotating them with different colors. Thus the HOCU approach keeps the desired property of being able to derive the so-called *sloppy/strict ambiguity*[11] while solving the *over-generation problem*, i.e. that classical HOU predicts more readings (e.g. those in (18)) than actually exist in natural language.

Even though we have only sketched the relevant ideas, it should be clear, that higher-order colored unification provides a general frame-

[11] the reading where Peter loves his own wife is called the strict reading, because the reference *his* is interpreted strictly. The other one is sloppy, as the reference is.

work for specifying the linguistic information for the construction process that avoids over-generalization (i.e. the construction of linguistically undesired readings of discourses).

1.5. 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 transforms sets of equations into solved forms (definition 8). 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 $\mathtt{c} =^c \mathtt{d}$ for color equations). Sets $\mathcal{E}$ of equations in solved form have a unique most general $\mathcal{C}$-unifier $\sigma_{\mathcal{E}}$ that also $\mathcal{C}$-unifies the initial equation.

There are several rules that decompose the syntactic structure of formulae. We will only discuss two of them here (cf. definition 9 for a full set). The rule for abstractions transforms equations of the form $\lambda U.\mathbf{M} =^t \lambda V.\mathbf{N}$ to $[c^*/U]\mathbf{M} =^t [c^*/V]\mathbf{N}$, where c^* is a new constant[12], while the rule for applications decomposes $h_{\mathtt{a}}\mathbf{M}^1 \ldots \mathbf{M}^n =^t h_{\mathtt{b}}\mathbf{N}^1 \ldots \mathbf{N}^n$ to the set $\{\mathtt{a} =^c \mathtt{b}, \mathbf{M}^1 =^t \mathbf{N}^1, \ldots, \mathbf{M}^n =^t \mathbf{N}^n\}$, 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 us 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 $\{X_{\mathtt{c}} =^t f_{\mathtt{c}}, X_{\mathtt{d}} =^t g_{\mathtt{d}}\}$. The correct variable elimination rule transforms a set $\mathcal{E}$ of equations with $X =^t \mathbf{M} \in \mathcal{E}$ and $X \notin \mathbf{free}(\mathbf{M})$ into

$$[\![\mathbf{M}/X]\!]_{\mathcal{E}} \cup [\![X =^t \mathbf{M}]\!]_{\mathcal{E}}$$

where $[\![\mathbf{M}/X]\!]_{\mathcal{E}} = [\mathbf{M}^1/X_{\mathtt{c}_1}], \ldots, [\mathbf{M}^n/X_{\mathtt{c}_n}]$, such that the $\mathtt{c}_i$ are all colors of the variable X occurring in $\mathcal{E}$ and the $\mathbf{M}^i$ are appropriately colored variants (same color erasure) of $\mathbf{M}$. Similarly, $[\![X =^t \mathbf{M}]\!]_{\mathcal{E}} = \{X_{\mathtt{c}_1} =^t \mathbf{M}^1, \ldots, X_{\mathtt{c}_n} =^t \mathbf{M}^n\}$. Note that the induced substitution $[\![\mathbf{M}/X]\!]_{\mathcal{E}}$ (cf. definition 6) is independent of the colors in $\mathbf{M}$ and X. Correctness of the term elimination rule hinges on the fact that all color variants of X are eliminated simultaneously and that the substitution $[\![\mathbf{M}/X]\!]_{\mathcal{E}}$ induced by a pair $X =^t \mathbf{M}$ is also applied to the

[12] Actually, in definition 9 we use special variables, which behave somewhat like constants.

pair itself. Thus an unsolvable pair, such as $X_{\mathtt{a}} =^t c_{\mathtt{b}}$ gives rise to the induced substitution $[c_{\mathtt{a}}/X_{\mathtt{a}}]$ and leads to the pair $c_{\mathtt{a}} =^t c_{\mathtt{b}}$, where the decomposition rule can detect a color clash.

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 $X_{\mathtt{c}}\mathbf{M}^1 \dots \mathbf{M}^n =^t h_{\mathtt{d}}\mathbf{N}^1 \dots \mathbf{N}^m$.

The standard solution (due to Gérard Huet, cf. [32] for an introduction) for finding a complete set of solutions in this so-called **flex/rigid** situation is to substitute a term for X of type δ that will enable decomposition to be applicable afterwards: a so-called **general bindings** (definition 10) of the following form:

$$\mathcal{G}^h_\delta = \lambda Z^{\alpha_1} \dots Z^{\alpha_n}.@(H^1\overline{Z}) \dots (H^m\overline{Z})$$

where

- $\delta = \overline{\beta_n} \to \alpha$ and @ has type $\overline{\gamma_m} \to \alpha$
- one of the following holds:
  - $@ = Z^{\alpha_j}$ and $\alpha_j = \overline{\gamma_m} \to \alpha$ for some $j \leq n$, then $h = j$ and we call $\mathcal{G}^j_\delta$ a **projection binding** or
  - $@ = \mathbf{S}$ for some constant or free variable $\mathbf{S}$ of type $\overline{\gamma_m} \to \alpha$, then $h = \mathbf{S}$ and we call $\mathcal{G}^h_\delta$ an **imitation binding**.
- the H^i are new variables of type $\overline{\beta_n} \to \gamma_i$,

We can actually use colors[13] to get a better understanding of the situation. Therefore consider the unification problem of $X_{\mathtt{d}}a_{\mathtt{c}} =^t a_{\mathtt{d}}$. For the imitation solution $(\lambda Z.a_{\mathtt{d}})$ we "imitate" the right hand side, so the color on a must be $\mathtt{d}$. For the projection solution we instantiate $(\lambda Z.Z)$ for X and obtain $(\lambda Z.Z)a_{\mathtt{c}}$, which β-reduces to $a_{\mathtt{c}}$. We see that this "lifts" the constant $a_{\mathtt{c}}$ from the argument position to the top. Incidentally, the projection is only a $\mathcal{C}$-unifier of our colored example, if the color constants $\mathtt{c}$ and $\mathtt{d}$ are identical. However for calculating solutions for flex/rigid pairs, we do not need colors, since the color erasure of the instance is determined by the general bindings and the color annotations are added by the induced substitution. Thus the general rule for flex/rigid equations (definition 11) transforms equations of the form

$$\mathcal{E} \wedge X^\alpha_{\mathtt{c}}\mathbf{M}^1 \dots \mathbf{M}^n =^t h_{\mathtt{d}}\mathbf{N}^1 \dots \mathbf{N}^m$$

[13] This is another (didactic) application of higher-order colored unification, where we use colors to distinguish different symbol occurrences.

into

$$\mathcal{E} \wedge X_{\mathtt{c}}\mathbf{M}^1 \ldots \mathbf{M}^n =^t h_{\mathtt{d}}\mathbf{N}^1 \ldots \mathbf{N}^m \wedge X_{\mathtt{c}} =^t \mathcal{G}^h_\alpha$$

with immediate variable elimination of the new equation.

Finally we are left with 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 combinatory 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 often 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 3.3). 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 (cf. definition 12)) and specialize the unification algorithm accordingly (definition 16).

- In the linguistic applications, sketched in section 1.4, 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 the equations reduces the problem of finding solutions to the so-called **higher-order matching** problem, of which decidability has been proven for the subclass of third-order formulae [9] (see [30, 29] for other tractable fragments). This class, (intuitively allowing one only to nest functions as arguments up to depth three) covers all examples studied so far.

Before we discuss the applications, let us fortify our intuition about calculating higher-order colored unifiers for the following problem which aroused in our lemma speculation example in section 1.2.

$$F_{\mathtt{g}}(rev_{\mathtt{w}}(u_{\mathtt{w}}), h_{\mathtt{g}}, v_{\mathtt{g}}) =^t app_{\mathtt{g}}(rev_{\mathtt{B}}(Y_{\mathtt{A}}), cons_{\mathtt{g}}(X_{\mathtt{g}}, nil_{\mathtt{g}})) \qquad (27)$$

Since F is a function variable, we are in a flex/rigid situation, and have the possibilities of projection and imitation. There are three possible projections, $(\lambda UVW.U)$, $(\lambda UVW.V)$, $(\lambda UVW.W)$, which all lead to immediate failure, since they project up the rigid subterms $rev_{\mathtt{w}}(u_{\mathtt{w}})$, $h_{\mathtt{g}}$

or $v_{\mathtt{g}}$, which would clash with the head $app_{\mathtt{g}}$ of the right hand side. So we only have the imitation binding $\mathbf{B} := \lambda UVW\,.\,app(H(U,V,W), K(U,V,W))$ for F. Binding F to that (i.e. applying the induced $\mathcal{C}$-substitution $[\![\mathbf{B}/F]\!]_{\mathtt{g}} = [(\lambda UVW\,.\,app_{\mathtt{g}}(H_{\mathtt{g}}(U,V,W), K_{\mathtt{g}}(U,V,W))/F_{\mathtt{g}})]$) and decomposing the instantiated problem, we are left with the equations

$$\begin{array}{lcl} H_{\mathtt{g}}(rev_{\mathtt{w}}(u_{\mathtt{w}}), h_{\mathtt{g}}, v_{\mathtt{g}}) & =^t & rev_{\mathtt{B}}(Y_{\mathtt{A}}) \\ K_{\mathtt{g}}(rev_{\mathtt{w}}(u_{\mathtt{w}}), h_{\mathtt{g}}, v_{\mathtt{g}}) & =^t & cons_{\mathtt{g}}(X_{\mathtt{g}}, nil_{\mathtt{g}}). \end{array}$$

Choosing[14] the imitation $\lambda UVW\,.\,cons_{\mathtt{g}}(M_{\mathtt{g}}(U,V,W), N_{\mathtt{g}}(U,V,W))$ for $K_{\mathtt{g}}$ and the 1-projection binding $(\lambda UVW\,.\,U)$ for $H_{\mathtt{g}}$ we obtain

$$\begin{array}{rcl} rev_{\mathtt{w}}(u_{\mathtt{w}}) & =^t & rev_{\mathtt{B}}(Y_{\mathtt{A}}) \\ cons_{\mathtt{g}}(M_{\mathtt{g}}(rev_{\mathtt{w}}(u_{\mathtt{w}}), h_{\mathtt{g}}, v_{\mathtt{g}}), N_{\mathtt{g}}(rev_{\mathtt{w}}(u_{\mathtt{w}}), h_{\mathtt{g}}, v_{\mathtt{g}})) & =^t & cons_{\mathtt{g}}(X_{\mathtt{g}}, nil_{\mathtt{g}}) \end{array}$$

We can decompose again and obtain

$$\begin{array}{rcl} \mathtt{A} & =^c & \mathtt{w} \\ u_{\mathtt{w}} & =^t & Y_{\mathtt{A}} \\ X_{\mathtt{g}} & =^t & M_{\mathtt{g}}(rev_{\mathtt{w}}(u_{\mathtt{w}}), h_{\mathtt{g}}, v_{\mathtt{g}}) \\ N_{\mathtt{g}}(rev_{\mathtt{w}}(u_{\mathtt{w}}), h_{\mathtt{g}}, v_{\mathtt{g}}) & =^t & nil_{\mathtt{g}} \end{array}$$

The first two equations can directly be solved by eliminating $\mathtt{A}$ for $\mathtt{w}$ and $Y_{\mathtt{A}}$ (which is actually $Y_{\mathtt{w}}$ after the previous elimination) for $u_{\mathtt{w}}$. The third equation cannot be solved this way, since $M_{\mathtt{g}}(rev_{\mathtt{w}}(u_{\mathtt{w}}), h_{\mathtt{g}}, v_{\mathtt{g}})$ is not $\mathtt{g}$-monochrome, so we choose the 2-projection binding[15] $\lambda UVW\,.\,V$ for $X_{\mathcal{C}}$ and solve the fourth equation with the imitation binding $\lambda UVW\,.\,nil_{\mathtt{g}}$ and for $N_{\mathtt{g}}$. Eliminating these bindings allows us to simplify the equations to the trivial set $h_{\mathtt{g}} =^t h_{\mathtt{g}}$ and $nil_{\mathtt{g}} =^t nil_{\mathtt{g}}$. Thus one final solution of the unification problem is

$$[\lambda UVW\,.\,app_{\mathtt{g}}(U, cons_{\mathtt{g}}(V, nil_{\mathtt{g}}))/F_{\mathtt{g}}], [u_{\mathtt{w}}/Y_{\mathtt{A}}], [h_{\mathtt{g}}/X_{\mathtt{g}}]$$

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

[14] The 2-projection bindings for H and K are impossible for type reasons and all projection bindings but the 1-projection binding for H lead to immediate subsequent clash. The imitation binding for H leads to a solution $\lambda UVW\,.\,app_{\mathtt{g}}(rev_{\mathtt{g}}(L_{\mathtt{g}}(U,V,W)), cons_{\mathtt{g}}(Y, nil))$ for $F_{\mathtt{g}}$ that is not wanted in our motivating example, so we will not pursue it here.

[15] The imitation binding $\lambda UVW\,.\,Q_{\mathtt{g}}$ (Q a new variable) would also have worked.

2. The Colored λ-Calculus

In this section, we make the intuitive concepts introduced above formal by extending the simply typed λ-Calculus (see [18] for an introduction) with a concept of color annotations for constant and variable occurrences.

The set $\mathcal{T}$ of types is generated from a set $\mathcal{BT}$ of **base types** by function type construction ($\alpha \to \beta$). We write $\overline{\beta_n} \to \alpha$ for the type $\beta_1 \to \ldots \to \beta_n \to \alpha$ of n-ary functions.

The set of colors is built up from **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 definition of well-formed formulae differs from the standard one in the treatment of bound variables, which do not carry color annotations in the colored λ-calculus. Therefore we provide a separate class of variables for them. Concretely, we fix

- a **signature** $\Sigma = \bigcup_{\alpha\in\mathcal{T}} \Sigma^\alpha$ of **constant** symbols (we will use lower-case letters for these)
- countably infinite sets $\mathcal{V}^\alpha$ of **(free) variables** for each type $\alpha \in \mathcal{T}$ (we will use the upper-case letters X, Y, F, G, H and $\mathcal{V} := \bigcup_{\alpha\in\mathcal{T}} \mathcal{V}^\alpha$)
- countably infinite sets $\mathcal{LV}^\alpha$ of **local variables** for each type $\alpha \in \mathcal{T}$ (we will use the upper-case letters U, V, W, Z for these and $\mathcal{LV} := \bigcup_{\alpha\in\mathcal{T}} \mathcal{LV}^\alpha$)
- sets $\mathcal{C}$ of **color constants** and $\mathcal{X}$ of **color variables**.

Atomic occurrences of symbols in well-formed formulae can have a color annotation, therefore we fix the notation $\Sigma_\mathcal{Z}$ for the set $\{c_\mathtt{a} | c \in \Sigma, \mathtt{a} \in \mathcal{Z}\}$ for some subset $\mathcal{Z} \subset \mathcal{C} \cup \mathcal{X}$, and analogously for $\mathcal{V}_\mathcal{Z}$.

DEFINITION 1. (Well-Formed Formulae). For each $\alpha \in \mathcal{T}$ we inductively define the set $\mathit{wff}_\alpha(\Sigma)$ of **well-formed formulae of type** α by

- if $\mathbf{S} \in \Sigma^\alpha_\mathcal{Z} \cup \mathcal{V}^\alpha_\mathcal{Z} \cup \mathcal{LV}^\alpha$, then $\mathbf{S} \in \mathit{wff}_\alpha(\Sigma)$,
- if $\mathbf{A} \in \mathit{wff}_{\beta\to\alpha}(\Sigma)$ and $\mathbf{B} \in \mathit{wff}_\beta(\Sigma)$, then $\mathbf{AB} \in \mathit{wff}_\alpha(\Sigma)$, and
- if $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ and $Z \in \mathcal{LV}^\beta$, then $(\lambda Z\mathbf{.A}) \in \mathit{wff}_{\beta\to\alpha}(\Sigma)$. Just as in the standard λ-calculus, we will call any occurrence of the local variable Z in $\lambda Z\mathbf{.A}$ **bound**.

We call formulae of the form **AB applications**, and formulae of the form $\lambda Z_\alpha\mathtt{.}\mathbf{A}$ **abstractions**. Finally, we call a formula **A proper**, iff all occurrences of local variables in $\mathbf{A}$ are bound, we will denote the set of proper well-formed formulae of type α with $\mathit{pwff}_\alpha(\Sigma)$.

Note that with this definition the notion of free variables coincides with the standard one in the case of proper formulae. We will write $\mathbf{free}(\mathbf{A})$ for the set of (color and term) variables in $\mathbf{A}$. 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.

EXAMPLE 1. *$F_{\mathtt{A}}(s_{\mathtt{d}}(a_{\mathtt{c}}))$, $\lambda Z\mathtt{.}s_{\mathtt{d}}(Z)$ are examples of colored λ-terms while $\lambda Z\mathtt{.}s_{\mathtt{d}}(Z_{\mathtt{B}})$ is not (bound variables may not have colors).*

DEFINITION 2. We will first define 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 proper formula $\mathbf{B}$, which we also define inductively: $\mathcal{C}_\in(\mathbf{S}_{\mathtt{a}},\mathbf{S}_{\mathtt{a}}) = \{\mathtt{a}\}$ for symbols, $\mathcal{C}_\in(\mathbf{S},\mathbf{AB}) = \mathcal{C}_\in(\mathbf{S},\mathbf{A}) \cup \mathcal{C}_\in(\mathbf{S},\mathbf{B})$ for applications and finally $\mathcal{C}_\in(\mathbf{S},\lambda Z\mathtt{.}\mathbf{A}) = \mathcal{C}_\in(\mathbf{S},\mathbf{A})$. We will call a formula $\mathbf{A} \in \mathit{pwff}(\Sigma)$

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

Finally, we call a formula **A compatible** with a color $\mathtt{a}$, iff either $\mathtt{a}$ is a color variable or $\mathbf{A}$ is $\mathtt{a}$-monochrome.

Clearly the colors annotating the atoms do not affect $\beta\eta$-convertibility, since bound variables are not colored. Therefore,

$$(\lambda Z\mathtt{.}\mathbf{C})\mathbf{D} \longrightarrow_\beta [\mathbf{D}/Z]\mathbf{C} \qquad \text{and} \qquad (\lambda Z\mathtt{.}\mathbf{A}Z) \longrightarrow_\eta \mathbf{A} \tag{28}$$

where the local variable Z does not occur not freely in $\mathbf{A}$. Since bound variables do not carrry color information and consequently there are no restrictions on β-redexes or the substitution in β-reduction, we can lift all the known theoretical results to the colored calculus. In particular, we know that $\beta\eta$-reduction always terminates producing unique $\beta\eta$-normal forms and that $\beta\eta$-equality can be decided by reducing to $\beta\eta$-normal form and comparing for alphabetic equality. Based on this, we

can use the traditional versions of (long) $\beta\eta$ normal forms and (long) head normal forms.

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

DEFINITION 3. (Erasure). The **erasure** of colored λ-terms to simply typed λ-terms is defined by:

- $|\mathbf{S}_{\mathtt{a}}| = \mathbf{S}$ if $\mathbf{S} \in \Sigma \cup \mathcal{V}$ and $\mathtt{a} \in \mathcal{C} \cup \mathcal{X}$,
- $|X| = X$ if $X \in \mathcal{LV}$,
- $|(\mathbf{AB})| = (|\mathbf{A}||\mathbf{B}|)$, and
- $|(\lambda Z\,.\,\mathbf{A})| = \lambda Z\,.\,|\mathbf{A}|$.

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 specialization of well-typed substitutions that preserves syntactic color information, such as the skeleton (cf. section 4).

DEFINITION 4. ($\mathcal{C}$-Substitution). Let $\mathcal{Z} \subseteq \mathcal{X}$ and $\mathcal{W} \subseteq \mathcal{C} \cup \mathcal{X}$ be two finite, disjoint sets of colors, then a **$\mathcal{C}$-substitution** σ is a pair (σ^t, σ^c), where σ^t is a type-preserving mapping $\mathcal{V}_{\mathcal{W}} \to \mathit{pwff}(\Sigma)$ and $\sigma^c\colon \mathcal{Z} \to \mathcal{W}$ such that the **domain** $\mathbf{Dom}(\sigma) = \mathbf{Dom}^t(\sigma) \cup \mathbf{Dom}^c(\sigma)$ of σ with $\mathbf{Dom}^t(\sigma) := \{X_{\mathtt{a}} \in \mathcal{V}_{\mathcal{Z}} | \sigma^t(X_{\mathtt{a}}) \neq X_{\mathtt{a}}\}$ and $\mathbf{Dom}^c(\sigma) := \{\mathtt{A} \in \mathcal{Z} | \sigma^c(\mathtt{A}) \neq \mathtt{A}\}$, is finite. Furthermore, we assume that

- if $\mathtt{a} \in \mathcal{C}$, then $\sigma(X_{\mathtt{a}})$ must be $\mathtt{a}$-monochrome for any variable $X_{\mathtt{a}} \in \mathbf{Dom}^t(\sigma)$, and
- $|\sigma(X_{\mathtt{a}})| = |\sigma(X_{\mathtt{b}})|$ for all $X_{\mathtt{a}}, X_{\mathtt{b}} \in \mathbf{Dom}^t(\sigma)$.

Note that local variables can never appear in the codomain of the term substitution, since we have restricted that to proper formulae.

REMARK 1. *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.*

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 $\mathbf{A}$, we also have to instantiate $X_{\mathtt{b}}$ with suitable formula $\mathbf{A}'$ in order to satisfy the conditions for $\mathcal{C}$-substitutions in definition 4. In particular we have to guarantee that

– $|\mathbf{A}| = |\mathbf{A}'|$ and

– $[\mathbf{A}'/X_{\mathtt{b}}]$ is 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 $\mathbf{A}$. Intuitively we can obtain $\mathbf{A}'$ from $\mathbf{A}$ by re-dyeing all colors and color-variables to $\mathtt{b}$. 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 schema" which can be instantiated to any possible $\mathbf{A}'$. We call these schemata **flexichrome variants** and obtain a flexichrome variant for $\mathbf{A}$ by replacing each color or color-variable in $\mathbf{A}$ by distinct new color-variables.

DEFINITION 5. (a-Chrome Variant).
Let $\mathbf{A}, \mathbf{B} \in \mathit{pwff}_\alpha(\Sigma)$*, then we call* $\mathbf{B}$ *a*

– **flexichrome variant** *of* $\mathbf{A}$*, iff* $\mathbf{B}$ *is flexichrome and* $|\mathbf{A}| = |\mathbf{B}|$*,*

– **c-monochrome variant** *of* $\mathbf{A}$*, iff* $\mathbf{B}$ *is* c*-monochrome and* $|\mathbf{A}| = |\mathbf{B}|$*,*

– **b-chrome variant** *of* $\mathbf{A}$*, iff*

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

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

EXAMPLE 2. $s_{\mathtt{A}}(X_{\mathtt{B}})$ *is a flexichrome variant of* $s_{\mathtt{d}}(X_{\mathtt{c}})$*, and* $\lambda Z_{\cdot} s_{\mathtt{A}}(s_{\mathtt{B}} Z)$ *one of* $\lambda Z_{\cdot} s_{\mathtt{d}}(s_{\mathtt{c}} Z)$*, but* $\lambda Z_{\cdot} s_{\mathtt{A}}(s_{\mathtt{A}} Z)$ *and* $\lambda Z_{\cdot} s_{\mathtt{A}}(s_{\mathtt{c}} Z)$ *are not. Furthermore, the formulae* $s_{\mathtt{c}}(X_{\mathtt{c}})$ *and* $\lambda Z_{\cdot} s_{\mathtt{c}}(s_{\mathtt{c}} Z)$ *are* c*-chrome variants of* $s_{\mathtt{d}} X_{\mathtt{A}}$ *and* $\lambda Z_{\cdot} s_{\mathtt{d}}(s_{\mathtt{c}} Z)$ *respectively.*

LEMMA 1. *If a formula* $\mathbf{A}$ *is compatible with some color* $\mathtt{a} \in \mathcal{C} \cup \mathcal{X}$*, then there is an* a*-chrome variant* $\mathbf{G}$ *of* $\mathbf{A}$ *and a* $\mathcal{C}$*-substitution* ρ*, such that* $\rho(\mathbf{G}) = \mathbf{A}$*.*

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

DEFINITION 6. (Induced $\mathcal{Z}$-Substitution). Let $\mathbf{A} \in \mathit{pwff}_\alpha(\Sigma)$ be a proper formula, $X \in \mathcal{V}_\alpha$, and $\mathcal{Z} \subseteq \mathcal{C} \cup \mathcal{X}$, then we say that the $\mathcal{Z}$-substitution

$$[\![\mathbf{A}/X]\!]_{\mathcal{Z}} = \{[\mathbf{A}^{\mathtt{a}}/X_{\mathtt{a}}] | \mathtt{a} \in \mathcal{Z}\}$$

is **induced** by $[\mathbf{A}/X]$ iff for all $\mathtt{a} \in \mathcal{Z}$, the term $\mathbf{A}^{\mathtt{a}}$ is the $\mathtt{a}$-chrome variant of $\mathbf{A}$. Note that $[\![\mathbf{A}/X]\!]_{\mathcal{Z}}$ is a $\mathcal{C}$-substitution that has finite support whenever $\mathcal{Z}$ is finite. Furthermore it is unique up to the choice of new color variants in the flexichrome variants of $\mathbf{A}$. Since the induced $\mathcal{Z}$-substitution only depends on the erasures of $\mathbf{A}$ and X, we will also use it for uncolored formulae.

The significance of the induced $\mathcal{Z}$-substitutions is that for any $\mathcal{C}$-substitution σ with $\mathbf{Dom}(\sigma) = \{X\}_{\mathcal{Z}}$, we have $\sigma = [\![|\sigma(X)|/X]\!]_{\mathcal{Z}}$. In other words, the well-formedness conditions for $\mathcal{C}$-substitutions ensure, that the $X_{\mathcal{Z}}$-part of a substitution can be induced from the erasure alone. In the unification we will use the fact that if we know the structure of the color erasure of a general binding, then we can already fix the X part of the solution.

EXAMPLE 3. *Let* $\mathcal{Z} = \{\mathtt{a}, \mathtt{b}, \mathtt{A}\}$ *and* $\mathbf{A} = f_{\mathtt{c}}(g(X_{\mathtt{a}}))$, *then*

$$[\![\mathbf{A}/X]\!]_{\mathcal{Z}} = [f_{\mathtt{a}}(g_{\mathtt{a}}(X_{\mathtt{a}}))/X_{\mathtt{a}}], [f_{\mathtt{b}}(g_{\mathtt{b}}(X_{\mathtt{b}}))/X_{\mathtt{b}}], [f_{\mathtt{C}}(g_{\mathtt{D}}(X_{\mathtt{E}}))/X_{\mathtt{A}}]$$

3. Unification

The central data structure for higher-order unification is that of unification problems, i.e. sets of pairs $\mathbf{A} =^t \mathbf{B}$ of formulae with coinciding types and pairs of colors $\mathtt{a} =^c \mathtt{b}$. We will represent these sets as conjunctions and write $\doteq$, if it is irrelevant whether we mean $=^t$ or $=^c$. Note that we do not restrict ourselves to proper unification problems in this paper (we will call a unification problem **proper** iff all of the formulae occurring in it are).

DEFINITION 7. ($\mathcal{C}$-Unifier). We call a $\mathcal{C}$-substitution $\theta = (\theta^t, \theta^c)$ a **$\mathcal{C}$-unifier** of a unification problem

$$\mathcal{E} = \mathbf{A}^1 =^t \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^t \mathbf{B}^n \wedge \mathtt{a}^1 =^c \mathtt{b}^1 \wedge \ldots \wedge \mathtt{a}^m =^c \mathtt{b}^m$$

iff

- $\theta^t(\mathbf{A}^i) =_{\beta\eta} \theta^t(\mathbf{B}^i)$ for all $1 \leq i \leq n$ and
- $\theta^c(\mathtt{a}^i) = \theta^c(\mathtt{b}^i)$ for all $1 \leq i \leq m$

and we will denote the set of $\mathcal{C}$-unifiers of $\mathcal{E}$ with $\mathbf{U}(\mathcal{E})$.

For a set $\mathcal{W} \subseteq \mathcal{V}_{\mathcal{Z}} \cup \mathcal{X}$ of variables, we call $\mathcal{C}$-substitutions σ and ρ equal on $\mathcal{W}$ (we denote that by $\sigma = \rho[\mathcal{W}]$), iff for all $X \in \mathcal{W}$ $\sigma(X) = \rho(X)$. We will use the obvious extension of this equality to $\beta\eta$-equality and to sets of equations, furthermore we will abbreviate $\sigma = \rho[\mathbf{free}(\mathcal{E})]$ with $\sigma = \rho[\mathcal{E}]$ for a unification problem $\mathcal{E}$.

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 $\sigma =_{\beta\eta} \rho \circ \theta[\mathcal{E}]$. 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 8. (Solved Form). Let $\mathcal{E} := \mathbf{A} \doteq \mathbf{B} \wedge \mathcal{E}'$ be a unification problem, then we call the pair $\mathbf{A} \doteq \mathbf{B}$ **solved** in $\mathcal{E}$, iff either

– it is a term pair $X_{\mathtt{a}} =^t \mathbf{B}$ for some variable $X \in \mathcal{V}$, some proper formula $\mathbf{B}$ and some color $\mathtt{a} \in \mathcal{C} \cup \mathcal{X}$ and

- $X_{\mathtt{a}} \notin \mathbf{free}(\mathbf{B})$ and
- if $X_{\mathtt{c}} \in \mathbf{free}(\mathcal{E}'))$ for some $\mathtt{c} \in \mathcal{Z}$, then it occurs exactly as the left hand side of a pair $X_{\mathtt{c}} =^t \mathbf{C}$, such that $|\mathbf{B}| = |\mathbf{C}|$ and $\mathbf{C}$ is $\mathtt{c}$-chrome, or

– it is a color pair $\mathtt{A} =^c \mathtt{b}$ for some color variable $\mathtt{A} \in \mathcal{X}$, such that $\mathtt{A} \neq \mathtt{b}$ and $\mathtt{A} \notin \mathbf{free}(\mathcal{E}')$.

We say that $\mathcal{E}$ is in **solved form**, iff all its pairs are solved in $\mathcal{E}$. Clearly, any proper $\mathcal{C}$-substitution

$$\sigma = [\mathbf{A}^1/X^1_{\mathtt{a}_1}], \ldots, [\mathbf{A}^n/X^n_{\mathtt{a}_n}], [\mathtt{a}_1/\mathtt{A}_1], \ldots [\mathtt{a}_m/\mathtt{A}_m]$$

uniquely determines a solved unification problem

$$\mathcal{E}_\sigma = X^1_{\mathtt{a}_1} =^t \mathbf{A}^1 \wedge \ldots \wedge X^n_{\mathtt{a}_n} =^t \mathbf{A}^n \wedge \mathtt{A}_1 =^c \mathtt{a}_1 \wedge \mathtt{A}_m =^c \mathtt{a}_m$$

in solved form. We will use $[\![X =^t \mathbf{A}]\!]$ for $\mathcal{E}_{[\mathbf{A}/X]}$, where $[\![\mathbf{A}/X]\!]$ is the induced substitution (cf. definition 6).

Conversely, the conditions on solved forms ensure that the corresponding substitutions are $\mathcal{C}$-substitutions: The first condition ensures well-definedness (occurs-check), idempotence, and properness while the second ensures that $\sigma_{\mathcal{E}}$ is a $\mathcal{C}$-substitution.

$$\frac{(\lambda U_\alpha . \mathbf{A}) =^t (\lambda V_\alpha . \mathbf{B}) \quad Z \in \mathcal{LV} \text{ new}}{[Z/U]\mathbf{A} =^t [Z/V]\mathbf{B}} \mathcal{SIM}(\alpha)$$

$$\frac{(\lambda U_\alpha . \mathbf{A}) =^t \mathbf{B} \quad Z \in \mathcal{LV} \text{ new}}{[Z/U]\mathbf{A} =^t (\mathbf{B}Z)} \mathcal{SIM}(\eta)$$

$$\frac{h_{\mathtt{a}}\overline{\mathbf{U}^n} =^t h_{\mathtt{b}}\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in \Sigma \cup \mathcal{LV}}{\mathtt{a} =^c \mathtt{b} \wedge \mathbf{U}^1 =^t \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^t \mathbf{V}^n \wedge \mathcal{E}} \mathcal{SIM}(dec)$$

Figure 2. Decomposition Rules in $\mathcal{SIM}$

$$\frac{\mathtt{A} =^c \mathtt{b} \wedge \mathcal{E} \quad \mathtt{A} \in \mathcal{X} \cap \mathbf{free}(\mathcal{E})}{\mathtt{A} =^c \mathtt{b} \wedge [\mathtt{b}/\mathtt{A}]\mathcal{E}} \mathcal{SIM}(elim{:}col)$$

$$\frac{\mathcal{E} \quad \begin{array}{c} F \notin \mathbf{free}(\mathbf{A}) \quad F_{\mathtt{a}}\overline{Z^k} =^t \mathbf{A} \in \mathcal{E} \\ \mathcal{LV}(\mathbf{A}) \subseteq \{Z^i\} \subseteq \mathcal{LV} \end{array}}{[\![F =^t \lambda\overline{Z^k} . \mathbf{A}]\!]_{\mathcal{C}_\in(F,\mathcal{E})} \wedge [\![\lambda\overline{Z^k} . \mathbf{A}/F]\!]_{\mathcal{C}_\in(F,\mathcal{E})}(\mathcal{E})} \mathcal{SIM}(elim{:}term)$$

Figure 3. Variable Elimination Rules in $\mathcal{SIM}$

3.1. Simplification

DEFINITION 9. ($\mathcal{SIM}$: Simplification of $\mathcal{C}$-Unification Problems). The rules for constraint simplification consist of the decomposition rules in figure 2 and the variable elimination rules in figure 3

In contrast to the simple higher-order unification we have to ensure that the resulting solutions are $\mathcal{C}$-substitutions, therefore, we have to apply $[\![\lambda\overline{X^k} . \mathbf{A}/F]\!]_{\mathcal{C}_\in(F,\mathcal{E})}$ to $\mathcal{E}$ if we eliminate $F_{\mathtt{a}}$ with $\lambda\overline{X^k} . \mathbf{A}$. Note that this approach works even if the pair $F_{\mathtt{a}}\overline{Z^k} =^t \mathbf{A}$ is not well-colored, since any color clash (say if $\mathbf{A}$ contains a $\mathtt{b}$-colored symbol) would be detected during decomposition of the resulting pair $\mathbf{A}^{\mathtt{a}} =^t \mathbf{A}$.

We apply these rules with the understanding that the operators $\wedge$ and $=^t$ are commutative and associative and that trivial pairs may be dropped. 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 2. *If* $\mathcal{D}\colon \mathcal{E} \vdash_{\mathcal{SIM}} \mathcal{E}'$, *then* $\mathbf{U}(\mathcal{E}) = \mathbf{U}(\mathcal{E}')[\mathcal{E}]$.

Proof. We will only concern ourselves with $\mathcal{SIM}(elim{:}term)$, since the other rules are like the uncolored ones. So let $\mathcal{E}$ be a $\mathcal{C}$-unification problem and $(F_{\mathtt{a}}\overline{Z}) =^t \mathbf{A}$ be the pair in $\mathcal{E}$ that the rule $\mathcal{SIM}(elim{:}term)$ acts upon. Furthermore let $F_{\mathtt{b}} \in \mathbf{free}(\mathcal{E})$ for some $\mathtt{b} \neq \mathtt{a}$.

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

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

since the Z_i are not in $\mathbf{Dom}(\theta)$. Now we know that $|\theta(F_{\mathtt{b}})| = \theta(F_{\mathtt{a}})|$, since θ is a $\mathcal{C}$-substitution, on the other hand $\theta(F_{\mathtt{b}})$ is compatible with $\mathtt{b}$, so there is a unique $\mathtt{b}$-chrome variant $\mathbf{G}$ of $\lambda\overline{Z}\mathbf{.A}$ and a substitution ρ, such that $\rho(\mathbf{G}) = \lambda\overline{Z}\mathbf{.A}$ by lemma 1, since $\mathtt{b}$ was chosen arbitrarily in the set $\mathcal{C}_{\in}(F, \mathcal{E})$, we obtain the assertion. □

Clearly the $\mathcal{SIM}$ Transformations are a joint generalization of the first-order colored unification algorithm as it has been presented in [21] and Huet's simplification rules [32]; they are terminating 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 $h_{\mathtt{a}}\overline{\mathbf{U}} =^t k_{\mathtt{b}}\overline{\mathbf{U}}$, where at least one of the heads $h_{\mathtt{a}}$ and $k_{\mathtt{a}}$ is a colored variable.

3.2. General Unification

The classical approach to higher-order unification on un-colored terms reduces the problem of finding solutions for un-colored $\mathcal{SIM}$-normal pairs to the the general binding problem: Given a type α and a symbol @, find the most general well-formed formula of type α that has head @. Indeed it is sufficient to instantiate the head variables in $\mathcal{SIM}$-normal pairs with such general bindings to obtain a complete set of HOU transformations. Since we have already generalized the simplification rules to deal with the color annotations, and we know from the classical case, that the erasures of the instantiations of head variables must be general bindings, it is sufficient to employ the uncolored rules for general

higher-order unification. This will allow us to use most of the meta-theory directly from the un-colored case (see for instance [32, 24]). To keep this paper self-contained, let us restate the definitions.

DEFINITION 10. (General Binding). *We call the formula*

$$\mathcal{GB}^h_\delta = \lambda Z^{\alpha_1} \ldots Z^{\alpha_n}.@(H^1\overline{Z}) \ldots (H^m\overline{Z})$$

a **general binder** *iff*

- $\delta = \overline{\beta_n} \to \alpha$ *and* $@$ *has type* $\overline{\gamma_m} \to \alpha$
- *one of the following holds:*
  - $@ = Z^{\alpha_j}$ *and* $\alpha_j = \overline{\gamma_m} \to \alpha$ *for some* $j \leq n$*, then* $h = j$ *and we call* $\mathcal{G}^j_\delta$ *a* **projection binding** *or*
  - $@ = \mathbf{S}$ *for some constant or free variable* $\mathbf{S}$ *of type* $\overline{\gamma_m} \to \alpha$*, then* $h = \mathbf{S}$ *and we call* $\mathcal{G}^h_\delta$ *an* **imitation binding**.
- *the* H^i *are new variables of type* $\overline{\beta_n} \to \gamma_i$*,*

Note that this definition is unique up to the names of the new variables H^i *and only depends on the signature* Σ*. Finally, for given type* α*, and head* h *we collect the imitation binding and all projection bindings (there need not be projection bindings for all types) in set of* **approximations** *of* h *and* α

$$\mathcal{AB}^h_\alpha(\Sigma) := \{\mathcal{GB}^h_\alpha(\Sigma)\} \cup \{\mathcal{GB}^j_\alpha(\Sigma) | j \leq n\}$$

The significance of this class of formulae is given by the following theorem, which is a simple consequence of the normal form theorem.

THEOREM 1. (General Binding Theorem). *Let* $\mathbf{A} \in pwff_\alpha(\Sigma)$ *be a formula with* $\mathbf{head}(\mathbf{A}) = h$*, then there exists a substitution* ρ*, such that* $\rho(\mathbf{G}) =_{\beta\eta} \mathbf{A}$ *where* $\mathbf{G} = \mathcal{GB}^h_\alpha(\Sigma)$ *is the general binding for* h *and* α*. Moreover, if* $\mathbf{A}$ *is a head normal form, then the depth of* ρ *is strictly less than that of* $\mathbf{A}$*.*

DEFINITION 11. ($\mathcal{CUT}$: Transformations for $\mathcal{C}$-Unification). Let $\mathcal{CUT}$ be the system $\mathcal{SIM}$ augmented by the inference rules in figure 4. Just as in $\mathcal{SIM}$ leave the associativity and commutativity of $\wedge$, $=^t$, and $=^c$ implicit. We have combined the classical imitation ($\mathbf{G}$ has head h) and projection ($\mathbf{G}$ is a projection binding) transformations (see [32]) into $\mathcal{CUT}(flex/rig)$. This set of rules is used with the convention that all formulae are eagerly reduced to $\mathcal{SIM}$-normal form. In particular by eliminating the new pairs $F_{\mathtt{a}} =^t \mathbf{G}$ instantiate colored variables in $\mathcal{E}$ with the correctly colored variants of $\mathbf{G}$.

$$\frac{F_{\mathtt{a}}\overline{\mathbf{U}^n} =^t F_{\mathtt{b}}\overline{\mathbf{V}^n} \wedge \mathcal{E}}{\mathtt{a} =^c \mathtt{b} \wedge \mathbf{U}^1 =^t \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^t \mathbf{V}^n \wedge \mathcal{E}}\,\mathcal{CUT}(dec)$$

$$\frac{F_{\mathtt{a}}^{\alpha}\overline{\mathbf{U}} =^t h_{\mathtt{b}}\overline{\mathbf{V}} \wedge \mathcal{E} \quad \mathbf{G} \in \mathcal{AB}_{\alpha}^{h}(\Sigma)}{F_{\mathtt{a}} =^t \mathbf{G} \wedge F_{\mathtt{a}}\overline{\mathbf{U}} =^t h_{\mathtt{b}}\overline{\mathbf{V}} \wedge \mathcal{E}}\,\mathcal{CUT}(flex/rig)$$

$$\frac{F_{\mathtt{a}}^{\alpha}\overline{\mathbf{U}} =^t G_{\mathtt{b}}\overline{\mathbf{V}} \wedge \mathcal{E} \quad \mathbf{G} \in \mathcal{AB}_{\alpha}^{h}(\Sigma)}{F_{\mathtt{a}} =^t \mathbf{G} \wedge F_{\mathtt{a}}\overline{\mathbf{U}} =^t G_{\mathtt{b}}\overline{\mathbf{V}} \wedge \mathcal{E}}\,\mathcal{CUT}(guess)$$

Figure 4. General Colored Unification

The soundness of these rules is a direct consequence of the (colored) soundness of $\mathcal{SIM}$ and the soundness of classical HOU on the erasures. Therefore, we directly have the following theorem.

THEOREM 2. (Soundness of $\mathcal{CUT}$). *If $\mathcal{E} \vdash_{\mathcal{CUT}} \mathcal{E}'$ such that $\mathcal{E}'$ is in $\mathcal{C}$-solved form, then the substitution $\sigma_{\mathcal{E}'}|_{\mathbf{free}(\mathcal{E})} \in \mathbf{U}(\mathcal{E})$.*

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 [16], our set of transformations cannot be terminating in general. We will prove, that $\mathcal{CUT}$ is a complete $\mathcal{C}$-unification procedure, i.e. 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 a subset $\mathcal{CUT}_{\theta}$ of inference rules that approximate the solution θ (for details and proofs see[32, 24]). Even though $\mathcal{CUT}$ must be non-terminating in general (otherwise HOU would be decidable) we have the following semi-termination result.

THEOREM 3. *If $\mathcal{E}$ is a unification problem with unifier θ, then*

- *$\mathcal{CUT}_{\theta}$ is terminating.*
- *$\mathcal{CUT}_{\theta}$ conserves the subset of unifiers that are compatible with θ.*

– *if no transformation rule from $\mathcal{CUT}_\theta$ is applicable to $\mathcal{E}$, then $\mathcal{E}$ is in $\mathcal{C}$-solved form.*

In particular, a unification problem $\mathcal{E}$ has an unifier θ, iff there is a sequence of $\mathcal{CUT}_\theta$-transformations that terminates with a solved form $\mathcal{E}'$. Since $\mathcal{CUT}_\theta$ conserves the set of unifiers that approximate θ, and $\mathcal{E}_{\mathcal{E}'}$ is a most general unifier of $\mathcal{E}'$, $\mathcal{E}_{\mathcal{E}'}$ must be more general than θ. Since the colored unification transformations are also classical ones, we directly have semi-termination for the colored case. This leads to the following completeness result for higher-order colored unification.

THEOREM 4. (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 $\sigma'_{\mathcal{E}} \leq_{\beta\eta} \theta[\mathcal{E}]$.*

If we combine the soundness results from theorem 2 with the completeness result from theorem 4, we can characterize the set of solutions found by the algorithm $\mathcal{CUT}$ by the following corollary.

COROLLARY 1. *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 $\mathcal{E}$.

3.3. Pre-$\mathcal{C}$-Unification

As for unification in the simply typed λ calculus, the rule $\mathcal{CUT}(guess)$ gives rise to a combinatory 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 λ-calculus a flex-flex pair may have no solution, e.g. if the top-level variables of both terms are annotated by different colors. Consider the following examples:

EXAMPLE 4. *Let $F, G \in \mathcal{V}$, then the unification problem $F_{\mathtt{d}} a_{\mathtt{d}} =^t G_{\mathtt{c}} a_{\mathtt{c}}$ has no unifier. On the other hand $F_{\mathtt{d}} a_{\mathtt{c}} =^t G_{\mathtt{c}} a_{\mathtt{c}}$ has an unifier $[\lambda Z \,.\, Z/F_{\mathtt{d}}], [\lambda Z \,.\, Z/G_{\mathtt{c}}]$.*

The reason for this is the fact that projections, i.e. terms of the form $\lambda\overline{X^k}.X^i$, carry no color information and these are valid instances of colored variables like $F_{\mathtt{d}}$ or $G_{\mathtt{c}}$. Hence, in order to solve flex-flex pairs like the second one in example 4 we have to map one of the top-level variables to a projection formula. This gives rise to the following definition:

DEFINITION 12. (Flexible Chain). *Let $\mathcal{E}$ be a $\mathcal{C}$-unification problem, then a subset $\mathcal{E}' = \mathbf{A}^1 =^t \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^t \mathbf{B}^n$ 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 $2 \leq i \leq n$. We call $\mathbf{head}(\mathbf{A}^1) = F_{\mathtt{c}}$ and $\mathbf{head}(\mathbf{B}^n) = G_{\mathtt{d}}$ the* **left**– *and* **right ends** *of $\mathcal{E}'$.*

If $\mathtt{c}, \mathtt{d} \in \mathcal{C}$ and $\mathtt{c} \neq \mathtt{d}$ then we call $\mathcal{E}'$ a **reducible chain**, *otherwise a* **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 bind 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 3. *Let $\mathcal{E} = \mathcal{E}' \wedge \mathcal{E}_r$, 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 $1 \leq i \leq n$, such that $\sigma(\mathbf{head}(\mathbf{A}^i))$ or $\sigma(\mathbf{head}(\mathbf{B}^i))$ is a projection formula.*

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

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

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

DEFINITION 13. (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 in $\mathcal{E}$. 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 14. (Color Restriction). *Let $\mathcal{E}$ be a safe system, then the* **color restriction** *$cr(X_{\mathtt{a}}, \mathcal{E})$ of a colored variable $X_{\mathtt{a}}$ with respect to $\mathcal{E}$ is defined by*

- *$cr(X_{\mathtt{a}}, \mathcal{E}) = \mathtt{d}$ if $\mathtt{a} \in \mathcal{X}$ and there is flexible chain $\mathcal{E}'$ in $\mathcal{E}$ with left head $X_{\mathtt{a}}$ and right head $Y_{\mathtt{d}}$ for some $\mathtt{d} \in \mathcal{C}$.*
- *$cr(X_{\mathtt{a}}, \mathcal{E}) = \mathtt{a}$ 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 $\mathtt{c}$ and $\mathtt{c}'$ for a colored variable atom $X_{\mathtt{a}}$. Merging both sets we would obtain a reducible chain in $\mathcal{E}$, which contradicts our assumption that $\mathcal{E}$ is safe. Note that for any flex/flex pair $F_{\mathtt{a}}\overline{\mathbf{U}} =^t G_{\mathtt{b}}\overline{\mathbf{V}}$ in $\mathcal{E}$ either

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

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

EXAMPLE 5. *Both unification problems $F_{\mathtt{d}}a_{\mathtt{d}} =^t G_{\mathtt{c}}a_{\mathtt{c}}$ and $F_{\mathtt{d}}a_{\mathtt{c}} =^t G_{\mathtt{c}}a_{\mathtt{c}}$ from example 4 are reducible flexible chains, so any unifier has to be a projection. Indeed for the second one, the projection bindings $[\lambda Z\,.\,Z/F_{\mathtt{d}}], [\lambda Z\,.\,Z/G_{\mathtt{c}}]$ succeed, whereas they clash for the first problem.*

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

DEFINITION 15. (Trivial Unifier). *Let $\mathcal{E}$ be a pre-$\mathcal{C}$-solved $\mathcal{C}$-unification problem, such that $\mathcal{E}' := F_{\mathtt{a}}^{\overline{\alpha^n} \to \beta}\overline{\mathbf{U}^n} =^t G_{\mathtt{b}}^{\overline{\gamma^m} \to \beta}\overline{\mathbf{V}^m}$ is a pre-$\mathcal{C}$-solved pair in $\mathcal{E}$ and $\mathcal{H} := \{H^\beta | \beta \in \mathcal{T}\}$ be a reserved set of typed variables with $\mathcal{H} \cap \mathbf{free}(\mathcal{E}) = \emptyset$. Furthermore let*

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

If $cr(F_{\mathtt{a}}, \mathcal{E}) \neq cr(G_{\mathtt{b}}, \mathcal{E})$, then both are color variables and $\zeta_{\mathcal{E}'}$ is augmented by the color substitution $[cr(G_{\mathtt{b}}, \mathcal{E})/cr(F_{\mathtt{a}}, \mathcal{E})]$. 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. *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 definition 15, then $\zeta_{\mathcal{E}'}$ is a $\mathcal{C}$-unifier for $\mathcal{E}'$, since

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

and either $cr(F_\mathtt{a},\mathcal{E}) = cr(G_\mathtt{b},\mathcal{E})$ 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 definition 4: We have two cases

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

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

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

DEFINITION 16. ($\mathcal{CPT}$: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 $\mathcal{CUT}(dec)$ and $\mathcal{CUT}(flex/rig)$ by requiring that they may not be performed on $\mathcal{C}$-pre-solved pairs and replacing $\mathcal{CUT}(guess)$ by the inference rule in figure 5

$$\frac{F^\alpha_\mathtt{a}\overline{\mathbf{U}} =^t G_\mathtt{b}\overline{\mathbf{V}} \wedge \mathcal{E} \quad \mathbf{G} \in \mathcal{GB}^j_\alpha(\Sigma)}{F_\mathtt{a} =^t \mathbf{G} \wedge F_\mathtt{a}\overline{\mathbf{U}} =^t G_\mathtt{b}\overline{\mathbf{V}} \wedge \mathcal{E}}\,\mathcal{CPT}(flex/flex)$$

$F_\mathtt{a}\overline{\mathbf{U}} =^t G_\mathtt{b}\overline{\mathbf{V}}$ is a pair of a reducible chain.

Figure 5. Higher-Order Colored Pre-Unification

With the definitions above we obtain a completeness result for $\mathcal{CPT}$ similar to theorem 4.

THEOREM 5. *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 3. The only difference is that we use lemma 3 to account for the restricted flex/flex-case. □

Note that in contrast to classical higher-order pre-unification we cannot drop the $\mathcal{CUT}(guess)$ and $\mathcal{CUT}(dec)$ rules altogether, but the given restriction is severe enough to make pre-$\mathcal{C}$-unification tractable. In particular the restriction alleviates the need for unspecified imitations in $\mathcal{CUT}(guess)$, which makes full unification infinitely branching. Obviously, safety is a decidable property and colour restrictions are effectively computable: unification problems are finite, and there can be at most exponentially many paths in one. Thus a simple generate-and test procedure will do. Computationally, this exponential step will not slow down higher-order unification, since the β-normalization step is already non-elementary.

3.4. Higher-Order Patterns

There are certain syntactic fragments of the simply typed λ calculus, where the higher-order unification problem has better properties than in the general case. We will concentrate on *higher-order patterns* [26, 27], 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 of rippling have already been studied by Kraan et al. in the context of program synthesis in [25], 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 Kraan's work.

For the colored λ-calculus, the definition of higher-order patterns is very similar to the uncolored case; in fact it coincides with the traditional one for proper formulae.

DEFINITION 17. (Higher-Order Pattern). We call a formula $\mathbf{A} \in wff(\Sigma)$ a **higher-order pattern**, iff any occurrence of a free variable F^α in $\mathbf{A}$ must be in a subformula $\mathbf{B}$ of $\mathbf{A}$ of the form $FZ^{\varphi(1)} \dots Z^{\varphi(n)}$, where $Z^i \in \mathcal{LV}$ 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 α. 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 $\mathcal{C}$-substitution σ a **pattern substitution**, iff for all $X_{\mathtt{a}}$ in $\mathbf{Dom}(\sigma)$, $\sigma(X_{\mathtt{a}})$ is a higher-order pattern. Note that the class of higher-order patterns is closed under pattern substitutions.

EXAMPLE 6. *Let f, a be constants and F, G be variables and UVW be local variables of appropriate type, then $\lambda UVW.F_{\mathtt{a}}WU$, and $F_{\mathtt{a}}WU$, and $\lambda U.f_{\mathtt{A}}(\lambda V.F_{\mathtt{b}}UV)a_{\mathtt{a}}(G_{\mathtt{a}}U)$ are higher-order patterns, while Fa, $\lambda Z.F_{\mathtt{c}}Za_{\mathtt{B}}$, $\lambda Z.F_{\mathtt{a}}ZZ$, and $\lambda UV.F_{\mathtt{c}}U(VU)$ 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.*

This syntactic fragment allows to specialize the unification rules from definition 11.

DEFINITION 18. (Transformations for Pattern Unification). The inference rules for $\mathcal{UP}at$ are those of $\mathcal{CPT}$, together with the additional rules for safe flex/flex pairs shown in figures 6 and 7. The first one handles the case, where the heads are identical and the second one, where they are distinct.

$$\frac{F_{\mathtt{a}}^{\overline{\alpha_k}\to\beta}\overline{X^{\varphi(k)}} =^t F_{\mathtt{b}}\overline{Z^{\psi(k)}} \wedge \mathcal{E}}{\begin{array}{l} cr(F_{\mathtt{a}},\mathcal{E}) =^c cr(F_{\mathtt{b}},\mathcal{E}) \wedge \mathcal{E} \wedge F_{\mathtt{a}}\overline{Z^{\varphi(k)}} =^t F_{\mathtt{b}}\overline{Z^{\psi(k)}} \\ \wedge F_{\mathtt{a}} =^t \lambda\overline{W^k_{\alpha_{\varphi(k)}}}.H_{\mathtt{a}}\overline{W^{\rho(l)}} \wedge F_{\mathtt{b}} =^t \lambda\overline{W^k_{\alpha_{\varphi(k)}}}.H_{\mathtt{b}}\overline{W^{\rho(l)}} \end{array}}\mathcal{UP}at(same)$$

- φ and ψ are partial permutations from k to n
- 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.
- H is a new variable of type $\overline{\alpha_{\rho(l)}} \to \beta$

Figure 6. Pattern Unification with Identical Heads

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 erasure of which can

be represented by a more general uncolored higher-order pattern[16]). Consider for instance the pair

$$\lambda UVWZ.F_{\mathtt{a}}UVWZ =^t \lambda UVWZ.F_{\mathtt{b}}VUWZ$$

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

$$\begin{array}{lcl} \sigma_3 & := & [\lambda UVWZ.W/F_{\mathtt{a}}], [\lambda UVWZ.W/F_{\mathtt{b}}] \\ \sigma_4 & := & [\lambda UVWZ.Z/F_{\mathtt{a}}], [\lambda UVWZ.Z/F_{\mathtt{b}}] \end{array}$$

$$\frac{F_{\mathtt{a}}^{\overline{\alpha_k}\to\beta}\overline{Z^{\varphi(k)}} =^t G_{\mathtt{b}}^{\overline{\alpha_l}\to\beta}\overline{Z^{\psi(l)}} \wedge \mathcal{E}}{\begin{array}{l}\mathtt{a} =^c \mathtt{b} \wedge F_{\mathtt{a}}\overline{Z^{\varphi(k)}} =^t G_{\mathtt{b}}\overline{Z^{\psi(l)}} \wedge \mathcal{E} \\ \wedge F_{\mathtt{a}} =^t \lambda\overline{W^k_{\alpha_{\varphi(k)}}}.H_{\mathtt{a}}\overline{W^{\varphi'(m)}} \wedge G_{\mathtt{b}} =^t \lambda\overline{W^l_{\alpha_{\psi(l)}}}.H_{\mathtt{b}}\overline{W^{\psi'(m)}}\end{array}}\ \mathcal{UP}at(diff)$$

- φ, ψ be partial permutations from k (l) to n.
- φ' and ψ' are partial permutations from m into k, l, such that $\varphi'(m) = i$ and $\psi'(m) = j$, iff $\varphi(i) = \psi(j)$.
- H is a new variable of type $\overline{\alpha_{\psi'(m)}} \to \beta$

Figure 7. Higher-Order Colored Pattern Unification with Distinct Heads

The tractable nature of pattern unification hinges on the observation that the solving of flex/rigid pairs is deterministic, that is, all but the imitation or one projection immediately lead to failure. Thus for pattern unification we only can directly inherit the decomposition and flex/rigid rules from general colored higher-order unification and only have to concern ourselves with the flex/flex situations. Clearly, all the discussion about flexible chains also applies to higher-order patterns, so we keep the flex/flex rule for reducible flexible chains. This leaves us with the case of safe flex/flex pairs, where (as we have seen above) color clashes are not a problem. Therefore, we can directly adapt the well-known rules for higher-order pattern unification: If we have a pair $F_{\mathtt{a}}\overline{U^n} =^t F_{\mathtt{b}}\overline{V^n}$, then $F_{\mathtt{a}}$ is bound to to $\lambda\overline{Z^n}.H_{\mathtt{a}}\overline{W_k}$ and $F_{\mathtt{b}}$ to

[16] This observation shows that a generate-and-test procedure for colored pattern unification is infeasible, since this would have generated the uncolored solution and rejected it, erroneously predicting the absence of colored solutions.

$\lambda\overline{Z^n}.H_{\mathtt{b}}\overline{W_k}$, where the $\overline{W^k}$ are those bound variables, where $U^i = V^i$.[17] Unlike in the uncolored case, an application of this rule does not immediately solve the pair (the colors `a` and `b` need not be identical), but it transforms it into a form, in which decomposition can do the rest (this will always succeed, iff the pair the rule acts upon is safe).

For the remaining case of safe flex/flex pairs with differing head variables, we use a similar argumentation (directly modeled after the uncolored case) and rule. From this argumentation (the flex/rigid and the safe flex/flex cases are deterministic and the reducible flex/flex cases only involve projections), we can directly derive that colored pattern unification is decidable[18] and finitary i.e. pattern unification problems have at most finitely many most general unifiers.

THEOREM 6. (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.

4. Knowledge Representation

In this section we will work on the *semantics* of colors as they are used to encode additional knowledge about formulae into annotations of symbol occurrences. The colored λ-calculus provides the necessary devices to maintain this knowledge during the deduction by introducing colored substitutions and colored unifiers. It remains the question how to encode the given domain knowledge into colors and vice versa how to decode again the inherited knowledge from the annotations distributed over the entire term or formula.

In the examples in sections 1.1 and 1.2 we have used annotations (colors) of symbols to guide the deduction process in such a way that

[17] For any unifier σ we have $\sigma(F_{\mathtt{a}}) = \lambda\overline{Z^n}.\mathbf{A}$. Now, $\mathbf{A}$ can only have occurrences of Z^i, such that $U^i = V^i$: If we assume that $\mathbf{A}$ contains an occurrence of Z_i (say at position p) with $U^i \neq V^i$, then $\sigma(F)\overline{U^n} =_{\beta\eta} [\overline{U^n/Z^n}]\mathbf{A}$ and $\sigma(F)\overline{U^n} =_{\beta\eta} [\overline{U^n/Z^n}]\mathbf{A}$, so these differ at position p, which contradicts the assumption that σ unifies $\mathcal{E}$.

[18] The termination and confluence arguments can be directly modeled after the standard case.

each intermediate result of deduction has to satisfy specific restrictions. Rippling uses the semantic information that we must apply the induction hypothesis to rewrite the induction conclusion. Additionally it assumes that the hypothesis is always homomorphically embedded into the intermediate results of manipulating the conclusion. Each symbol occurrence inside the hypothesis corresponds uniquely to a symbol occurrence inside the (manipulated) conclusion. We may encode this knowledge by dyeing symbol occurrences which are inside the range of the mapping white while all others grey. If we need a more granular information about corresponding symbol occurrences in hypothesis and conclusion we may even attach unique colors to the corresponding occurrences instead of coloring them all uniquely white. Thus we split up the set of color constants $\mathcal{C}$ into a subset $\mathcal{D}$ of color constants which contribute to the skeleton, like for instance "white", and other colors $\mathcal{C} \setminus \mathcal{D}$, like "grey", which indicate wave-fronts.

Encoding the mapping into annotations we have to supply an appropriate function $\Omega_{\mathcal{D}}$ which will synthesize the information encoded into annotations. In particular we are interested in all symbol occurrences of the manipulated conclusion which are related to occurrences of the hypothesis. We can easily determine these occurrences by the color of their annotation. Since we also like to extract the subterm-relations in which these symbols occur, $\Omega_{\mathcal{D}}$ has also to incorporate knowledge about the subterm relations between symbols annotated with colors of $\mathcal{D}$. Thus, instead of computing a set of unrelated symbol occurrences, $\Omega_{\mathcal{D}}$ will construct new (annotated) formulae according to the original term structure by using the symbols annotated with colors of $\mathcal{D}$.

We introduce the notion of a skeleton in two steps. First, we characterize the extraction and gluing part within the definition of an *initial skeleton* $\widetilde{\Omega}_{\mathcal{D}}$. This skeleton will still contain some redundancies which will be removed by applying some reduction rules yielding the final skeleton $\Omega_{\mathcal{D}}$.

Since $\Omega_{\mathcal{D}}$ has to deliver the subterm-relations between the symbol occurrences under consideration, the skeleton is built up along the formula construction of annotated formulae. Thus, the skeleton inherits the structure of the original formula and is basically defined as a homomorphic extension of a mapping on symbol occurrences. The skeleton of an application is a set of applications constructed of the skeletons of the arguments, also lambda-expressions are translated in a homomorphic way to a set of lambda-expressions using the skeletons of the redex.

Since we are interested in symbol occurrences annotated with colors of $\mathcal{D}$, $\widetilde{\Omega}_{\mathcal{D}}$ will map such symbols $h_{\mathtt{d}}$ into the singleton set $\{h_{\mathtt{d}}\}$. If a symbol h is annotated with a color $\mathtt{c}$ which is not member of $\mathcal{D}$ then

we actually like to discard this symbol occurrence. If h has base type α then $\widetilde{\Omega}_{\mathcal{D}}$ will return a singleton set containing a dummy symbol f_ω to preserve well-typedness. For h of non-base type, the $\widetilde{\Omega}_{\mathcal{D}}$ has to provide appropriate λ-expressions such that computing $\widetilde{\Omega}_{\mathcal{D}}(h_{\mathtt{c}}\mathbf{AB})$ will return the union of the skeletons of $\mathbf{A}$ and $\mathbf{B}$. This can be done by mapping $h_{\mathtt{c}}$ to the set of all the possible projections to its arguments.

We are left with the case of bound variables which have no annotations at all. In this case $\widetilde{\Omega}_{\mathcal{D}}$ again returns a singleton set containing the bound variable.

We summarize the notion of a initial skeleton into the following definition:

DEFINITION 19. (Initial Skeleton). Let $\mathbf{A} \in pwff_\alpha(\Sigma)$ be a well-formed formula and $\mathcal{D} \subset \mathcal{C}$ be the set of **skeleton colors**. Then the **initial skeleton** $\widetilde{\Omega}_{\mathcal{D}}$ is defined as follows:

- $\widetilde{\Omega}_{\mathcal{D}}(X) = \{X\}$ if $X \in \mathcal{LV}$,
- $\widetilde{\Omega}_{\mathcal{D}}(h_{\mathtt{d}}) = \{h_{\mathtt{d}}\}$ if $\mathtt{d} \in \mathcal{D} \cup \mathcal{X}$,
- $\widetilde{\Omega}_{\mathcal{D}}(h_{\mathtt{d}}) = \{f_\omega\}$ if $\mathtt{d} \in (\mathcal{C} \setminus \mathcal{D})$, and the type of h is base,
- $\widetilde{\Omega}_{\mathcal{D}}(h_{\mathtt{d}}) = \{(\lambda\overline{Z_n}.f_\omega^{\alpha_1\to\beta}Z_1), \ldots, (\lambda\overline{Z_n}.f_\omega^{\alpha_n\to\beta}Z_n)\}$ if $\mathtt{d} \in (\mathcal{C} \setminus \mathcal{D})$, where $h$is of type $\overline{\alpha_n} \to \beta$ (β base type).
- $\widetilde{\Omega}_{\mathcal{D}}(\mathbf{BC}) = \{(\mathbf{B}'\mathbf{C}')|\mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B}) \text{ and } \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C})\}$, and
- $\widetilde{\Omega}_{\mathcal{D}}(\lambda Z.\mathbf{B}) = \{\lambda Z.\mathbf{B}'|\mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}$.

We had to invent type-conversion function to obtain typed terms as members of the skeleton. Since in some terms, occurrences of these type-conversion functions are redundant, we use the following ω-reduction rules to remove redundancies in the initial skeleton.

DEFINITION 20. (ω-Reduction). We say that a well-formed formula $\mathbf{B}$ is obtained by a well-formed formula $\mathbf{A}$ by a one-step **ω-reduction** ($\mathbf{A} \longrightarrow_\omega \mathbf{B}$), if it is obtained by applying one of the rules given in Fig. 8 to a well-formed part of $\mathbf{A}$ As usual we define the transitive closure of the reduction relation $\longrightarrow_\omega$ with $\longrightarrow_\omega^*$ and use $\longrightarrow_{\beta\eta\omega}$ for the union of the reduction relations $\longrightarrow_\beta$, $\longrightarrow_\eta$, and $\longrightarrow_\omega$.

In order to compare two skeletons we like to disregard differences caused by $\beta\eta$-equality or ω-reduction. Hence, we combine $\beta\eta$ and ω-reductions to obtain a $\beta\eta\omega$-reduction.

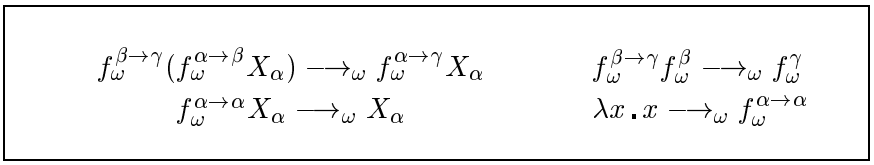

Figure 8. ω-Reduction Rules

THEOREM 7. (Strong Normalization). *Every sequence of $\beta\eta\omega$-reductions terminates and leads to a unique $\beta\eta\omega$-normal form.*

Proof. The Theorem is proven in the appendix (section A). □

Thus, $\beta\eta\omega$-reduction terminates for all λ-terms $\mathbf{A}$ and results in a unique normal form $\mathbf{A}\downarrow_{\beta\eta\omega}$. Based on the definitions of the initial skeleton and the $\beta\eta\omega$-reduction we define now the skeleton of a higher-order term as follows:

DEFINITION 21. (Skeleton). Let $\mathbf{A} \in pwff_\alpha(\Sigma)$ be a well-formed formula and $\mathcal{D} \subset \mathcal{C}$ be the set of **skeleton colors**. Then the **skeleton** $\Omega_\mathcal{D}(\mathbf{A})$ of $\mathbf{A}$ is defined as the $\beta\eta\omega$-normal form $\widetilde{\Omega}_\mathcal{D}(\mathbf{A})\downarrow_{\beta\eta\omega}$ of the initial skeleton. Notice that due to the confluence of $\beta\eta\omega$-reduction we can also intertwine this reduction and the computation of the initial skeleton which results in a one-pass computation of the skeleton.

The skeleton is independant of the representation of a term wrt. $\beta\eta$-equality, i.e. we do not have to normalize a term before computing its skeleton.

LEMMA 5. *For all* $\mathbf{A}, \mathbf{B} \in pwff_\alpha(\Sigma)$:

$$\mathbf{A} =_{\beta\eta} \mathbf{B} \textit{ implies } \Omega_\mathcal{D}(\mathbf{A}) = \Omega_\mathcal{D}(\mathbf{B})$$

The proof can be found in the appendix (section B). As a consequence we have to make compromises to the granularity in which we can annotate terms. Since we do not restrict the application of the η-rule (compared to the non-annotated λ-calculus) we cannot store any domain knowledge (annotations) to local variables. Otherwise the unrestricted application of the η-rule would destroy this knowledge.[19]

The skeleton function denotes an abstraction on colored terms. Rippling makes use of this abstraction by restricting possible deductions

[19] The introduction of type-conversion functions guarantee the $\alpha\beta\eta$-compatibility. Especially there is a subtle relation between mapping non-skeleton symbols to f_ω (thus introducing some kind of "partial" skeleton) and β-reduction as the following example illustrates: $\Omega_{\{\mathtt{a}\}}(\lambda UV\,.\,U)(\mathbf{A}_\mathtt{a}, \mathbf{B}_\mathtt{b}) = (\lambda UV\,.\,U)(\mathbf{A}_\mathtt{a}, f_\omega) =_\beta \{\mathbf{A}_\mathtt{a}\} = \Omega_{\{\mathtt{a}\}}(\mathbf{A}_\mathtt{a}) =_\beta \Omega_{\{\mathtt{a}\}}(\lambda UV\,.\,U)(\mathbf{A}_\mathtt{a}, \mathbf{B}_\mathtt{a})$

to formulas or terms with identical skeletons. Restricting the proof in the abstract space we are now able to implement a generate-and-test approach which first computes a deduction step, secondly calculates its abstraction (skeleton) and finally backtracks this step if the abstraction does not satisfy the given restrictions. But in order to plan a proof in the abstract space we like to *predict* whether a specific deduction step will satisfy the restrictions on its abstraction *without* computing the result on a ground level. Thus, the question arises whether we are able to calculate the abstraction (skeleton) of the result of a deduction step only by considering the abstractions (skeletons) of all involved formulas?

As rippling is mostly done in a rewriting environment[20], we will discuss the issues of reasoning in the abstract space in terms of a term rewriting calculus. In principle, term rewriting allows one to deduce a term $[\sigma(\mathbf{B})/\pi]\mathbf{C}$ from $\mathbf{C}$ iff there is an equation $\mathbf{A} = \mathbf{B}$ and a substitution σ such that $\sigma(\mathbf{A}) = \mathbf{C}|_\pi$. In order to reason entirely in the abstract space we have to be able to compute the skeleton of $[\sigma(\mathbf{B})/\pi]\mathbf{C}$ with the help of the skeletons of $\mathbf{C}$, $\mathbf{A}$ and $\mathbf{B}$. In case of first order logic we know that $\Omega_{\mathcal{D}}(\mathbf{A}) = \Omega_{\mathcal{D}}(\mathbf{B})$ implies $\Omega_{\mathcal{D}}(\sigma(\mathbf{A})) = \Omega_{\mathcal{D}}(\sigma(\mathbf{B}))$ (Substitution-Compatibility) and that $\Omega_{\mathcal{D}}(\mathbf{C}|_\pi) = \Omega_{\mathcal{D}}(\sigma(\mathbf{B}))$ implies $\Omega_{\mathcal{D}}(\mathbf{C}) = \Omega_{\mathcal{D}}([\sigma(\mathbf{B})/\pi]\mathbf{C}$ (Subterm-Compatibility). Once the abstractions of both sides of a rewrite rule are equal, its use to rewrite a term will not change the abstraction of the modified term.

We will discuss now the higher-order case: The property of subterm-compatibility is in principle an immediate consequence of the definition of a skeleton being a homomorphic extension of a mapping on symbol occurrences. Thus the following lemma holds; its proof is given in the appendix (section B).

LEMMA 6. *For all* $\mathbf{A}, \mathbf{B} \in \mathit{pwff}_\alpha(\Sigma)$

$$\Omega_{\mathcal{D}}(\mathbf{A}) = \Omega_{\mathcal{D}}(\mathbf{B}|_\pi) \textit{ implies } \Omega_{\mathcal{D}}(\mathbf{B}) = \Omega_{\mathcal{D}}([\mathbf{A}/\pi]\mathbf{B}).$$

The corresponding lemma about substitution-compatibility can not be lifted to the higher-order case. The reason is that an instantiation may enable the use of the β-rule which then, may delete parts of the skeleton.

EXAMPLE 7. *Consider two terms* $F_{\mathtt{d}}a_{\mathtt{c}}$ *and* $f_{\mathtt{d}}a_{\mathtt{c}}$. *Let* $\mathtt{c} \in \mathcal{D}$ *while* $\mathtt{d} \notin \mathcal{D}$ *such that both terms coincide in their skeletons* $a_{\mathtt{c}}$. *Instantiating* $F_{\mathtt{d}}$ *by* $\lambda X.b_{\mathtt{d}}$ *will (after* β*-reduction) result in terms* $b_{\mathtt{d}}$ *and* $f_{\mathtt{d}}a_{\mathtt{c}}$ *which do obviously not coincide in their skeleton.*

[20] We will denote the subterm $\mathbf{B}$ in $\mathbf{A}$ at position π with $\mathbf{A}|_\pi$ and the result of replacing this occurrence of $\mathbf{B}$ in $\mathbf{A}$ with $\mathbf{C}$ with $[\mathbf{C}/\pi]\mathbf{A}$.

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 too restrictive for practical reasons. What are the practical implications of this lack of substitution-compatibility? In order to guarantee that $\Omega_{\mathcal{D}}(\mathbf{C}) = \Omega_{\mathcal{D}}([\sigma(\mathbf{B})/\pi]\mathbf{C})$ holds we have now to test the missing property: each time an annotated rewrite rule $\mathbf{A} = \mathbf{B}$ (satisfying $\Omega_{\mathcal{D}}(\mathbf{A}) = \Omega_{\mathcal{D}}(\mathbf{B})$) is used to modify a term $\mathbf{C}$ we have to check whether $\Omega_{\mathcal{D}}(\sigma(\mathbf{A})) = \Omega_{\mathcal{D}}(\sigma(B))$ holds. Since this test can be made very efficiently, the practical implications of the missing property are rather small. We can still speculate about a proof on the abstract space although we have to keep in mind that in some cases the refinement will not be possible because of the missing substitution-property.

5. Related Work

Smaill and Green [31] developed the notion of *higher-order embeddings*. An embedding $\subsetneq$ is a relation on terms and $s \subsetneq t$ — speaking $\mathbf{A}$ *is embedded* in $\mathbf{B}$ — denotes intuitively that $\mathbf{A}$ is a skeleton of $\mathbf{B}$. As a base case each atomic expression $\mathbf{B}$ is embedded into itself: $t \subsetneq t$. Also a term $\mathbf{A}$ is embedded into an application $(\mathbf{B}_1\mathbf{B}_2)$ if it is embedded into one of its arguments or $\mathbf{A}$ is itself an application $(\mathbf{A}_1\mathbf{A}_2)$ and each $\mathbf{A}_i$ is embedded in $\mathbf{B}_i$. $\mathbf{A}$ is embedded into an abstraction $(\lambda U\,.\,\mathbf{B})$ if it is embedded into all instantiations $(\lambda U\,.\,\mathbf{B})\mathbf{C}$ for all $\mathbf{C}$ or $\mathbf{A}$ is itself an abstraction $(\lambda U\,.\,\mathbf{A}')$ and $(\lambda U\,.\,\mathbf{A}')\mathbf{C}$ is embedded into $(\lambda U\,.\,\mathbf{B})\mathbf{C}$ for all $\mathbf{C}$.

The notion of embeddings enables a generate-and-test procedure[21] based on standard higher order matching/unification which performs an (arbitrary) deduction steps and tests whether a specific term is embedded into the result of this step. Our approach to attach additional information at each symbol allows one to maintain the information about embedding during the deduction process since skeletons are stable with respect to subterm-replacement. This information is also necessary to restrict the number of possible solutions (e.g during higher-order unification) as soon as possible.

On one hand the generate-and-test approach requires only the existence of an algorithm which tests whether the skeleton is in some sense *embedded* (cf. [31]) in a formula which corresponds to a check whether the indicated pattern is matched by the formula. But on the

[21] For principal difficulties of this approach cf. footnote 16.

other hand, then we have no information how parts of the skeleton will be inherited during the deduction and therefore we have also no information how to select appropriate lemmata. Even worse, higher-order unification usually results in tremendous number of unifiers and we only can significantly reduce that number if we incorporate the knowledge about the invariance of the skeleton into the unification procedure. Thus, there is a strong need to incorporate a notion of skeleton into the deduction machinery.

Comparing this notion of embedding and our definition of skeletons one observe several conceptual differences. In case of applications the definition of embeddings does not preserve the intended subterm relation on (first-order) terms. For example, consider a first order term $g(h(\mathbf{A},\mathbf{C}),\mathbf{B})$ then $h(\mathbf{A},\mathbf{B}) \subsetneq g(h(\mathbf{A},\mathbf{C}),\mathbf{B})$ holds. This confusion of arguments of h and g may cause severe problems when defining termination orderings on rippling with the help of embeddings. In our setting the skeleton of $g(h(\mathbf{A},\mathbf{C}),\mathbf{B})$ is the set $\{h(\mathbf{A},f_\omega),\mathbf{B}\}$ and the use of the syntactic type-conversion functions prevents the mix-up of arguments. Hence, the intended subterm-relation is preserved.

6. Conclusion

Motivated by the first-order rippling/coloring method we have developed a colored higher-order logic and presented unification, pre-unification and pattern unification algorithms that we have proven to be correct and complete. In contrast to other semantic annotation techniques like sorts, the coloring technique allows one to add annotations to symbol *occurrences* in λ-terms. Thus it is possible to encode arbitrary structural information into colors and to maintain this information implicitly by the calculus during a deduction.

From an abstract point of view we can see that the colored λ-calculus and labeled deduction systems [10] share basic intuitions. Both use annotations to restrict the applicability of inference rules and provide a mechanism for maintaining the annotations during the inference. However, while LDS attach labels to formulae, HOCU annotates symbol occurrences with colors.

It seems plausible that colored logics can be embedded into suitable LDS if we assume that the labels have the same algebraic structure as the formulae they are attached to. Moreover, any LDS that deals with equality will probably need to maintain labels in such a term-structured form, since equality operates on subterm occurrences, which have to be represented in some way. In this case the colored λ-calculus allows one to deal with labels and formulae in a uniform and efficient way taking

advantage of the common structure of both. We leave a formal analysis of this relation of approaches to further research.

We have presented two applications of the colored λ-calculus. First, it provides a formal basis for the mechanization of higher-order reasoning with equality along the lines of [35, 20, 22] which develop heuristics that guide the difference reduction process in first-order equality calculi. Since rippling is an instance of this general difference reduction methodology, the calculus presented in this paper is a basis for an implementation of rippling in a higher-order setting which is required e.g. in case of middle-out reasoning [17, 23]. Secondly, it is also a logical basis for an interface for linguistic extra-semantical information in the construction of natural-language semantics. The algorithms presented in this paper and the linguistic analyses from [11, 13, 12, 15] have been implemented in the CHOLI system [8].

At least the linguistic applications suggest the need for more expressive color languages. In [15] we have used feature structures as colors to model the interaction of linguistic constraints. It turns out that the unification methods presented in this paper are sufficient to treat such extensions in a uniform way.

A further extension would be to alleviate the restriction that colors only annotate free symbol occurrences by allowing color annotations to subterm occurrences or to bound variables. This would allow to extend the scope of the calculus towards structural phenomena that is induced by the full term structure.

References

1. Hendrik P. Barendregt. *The Lambda-Calculus: Its Syntax and Semantics.* North-Holland, 1980.
2. D. Basin and T. Walsh. A calculus for and termination of rippling. *Special Issue of the Journal of Automated Reasoning*, 16(1-2):147–180, 1996.
3. Val Breazu-Tannen. Polymorphic rewriting conserves algebraic strong normalization and confluence. In *Proceedings of the ICALP*, pages 137–150, 1989.
4. A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, , and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993.
5. 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.
6. A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.
7. M. Dalrymple, S. Shieber, , and F. Pereira. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14:399–452, 1991.

8. Ralph Debusmann, Markus Egg, Claire Gardent, Alexander Koller, Karsten Konrad, Joachim Niehren, Guido Schaefer, Stephan Thater, Verena Winter, and Feiyu Xu. A natural language system for semantic construction and evaluation. CLAUS Report 102, University of the Saarland, Saarbrücken, 1998.
9. Gilles Dowek. Third order matching is decidable. *Annals of pure and applied mathematics*, 69:135–155, 1994.
10. D. Gabbay. *Labelled Deductive Systems*. Oxford Logic Guides, No 33. Oxford University Press, 1996.
11. C. Gardent and M. Kohlhase. Higher-order coloured unification and natural language semantics. In A. Zampolli, editor, *Proceedings of COLING'96*, 1996.
12. C. Gardent and M. Kohlhase. Computing parallism in discourse. In *Proceedings 15th International Joint Conference on Artificial Intelligence (IJCAI)*, Nagoya, Japan, 1997. Morgan Kaufman Publ.
13. Claire Gardent. Sloppy identity. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, pages 188–207. Springer, 1997.
14. Claire Gardent, Michael Kohlhase, and Karsten Konrad. Higher–order coloured unification: a linguistic application. CLAUS Report 97, University of the Saarland, Saarbrücken, 1997.
15. Claire Gardent, Michael Kohlhase, and Karsten Konrad. Higher–order coloured unification: a linguistic application. *Techniques Sciences Informatiques*, pages 1–28, 1999.
16. Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.
17. J. Hesketh. *Using Middle-Out Reasoning to Guide Induction*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, 1981.
18. J. Hindley and J. Seldin. *Introduction to Combinators and Lambda Calculus*. Cambridge University Press, 1986.
19. 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.
20. Dieter Hutter. Using rippling for equational reasoning. In S. Hölldobler, editor, *Proceedings 20th German Annual Conference on Artificial Intelligence KI-96*, pages 121–134, Dresden, Germany, 1996. Springer-Verlag, LNAI 1137.
21. Dieter Hutter. Colouring terms to control equational reasoning. *Journal of Automated Reasoning*, 18:399–442, 1997. Kluwer-Publishers.
22. Dieter Hutter. Hierarchical proof planning using abstractions. In D. Dankel II, editor, *Proceedings 10th Annual Florida AI Research Symposium, FLAIRS'97, Track: Using AI methods to control automated deduction*, pages 181–185, Daytona Beach, USA, 1997. M. Fishman.
23. A. Ireland and A. Bundy. Productive use of failure in inductive proof. *Special Issue of the Journal of Automated Reasoning*, 16(1-2):79–111, 1996.
24. Michael Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*. PhD thesis, Universität des Saarlandes, 1994.
25. I. Kraan, D. Basin, , and A. Bundy. Middle-out reasoning for synthesis and induction. *Journal of Automated Reasoning*, 16(1-2):113–145, 1996.
26. Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, September 1991.
27. Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14:321–358, 1992.

28. R. Montague. The proper treatment of quantification in ordinary english. In R. Montague, editor, *Formal Philosophy. Selected Papers*. Yale University Press, New Haven, 1974.
29. V. Padovani. *Filtrage d'order supérieur*. Thése de doctorat, Université Paris VII, 1996.
30. Christian Prehofer. Decidable higher-order unification problems. In Alan Bundy, editor, *Proceedings of the 12th Conference on Automated Deduction*, number 814 in LNAI, pages 635–649, Nancy, France, 1994. Springer Verlag.
31. A. Smaill and I. Green. Higher-order annotated terms for proof search. In *Proceedings of the International Conference on Theorem Proving in Higher Order Logics (TPHOLs'96)*, 1996.
32. Wayne Snyder. *A Proof Theory for General Unification*. Progress in Computer Science and Applied Logic. Birkhäuser, 1991.
33. R. Statman. Logical relations and the typed lambda calculus. *Information and Computation*, 65, 1985.
34. W. Tait. Intensional interpretation of functionals of finite type I. *Information and Computation*, 32:198–212, 1967.
35. T. Walsh, A. Nunes, , and A. Bundy. The use of proof plans to sum series. In D. Kapur, editor, *Proceedings of the 11th Conference on Automated Deduction*, pages 325–339, Saratoga Spings, NY, USA, 1992. Springer Verlag, LNCS 607.

Appendix

A. Strong Normalization 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. For the termination proof, we use the well-known logical-relations method due to Tait [34] and Statman [33]. A simpler approach using a permutation argument for $\beta\eta$- and ω-reduction will not work, since in some cases an η-step has to be simulated by two ω-reduction steps (the reader is refered to the proof of lemma 11 for details). Neither can we use the general combination result by Breazu-Tannen & Gallier [3] that proves strong normalization of $\beta\eta\mathcal{R}$ for strongly normalizing algebraic $\mathcal{R}$, since ω is not algebraic.

For the confluence proof we only need to show that $\beta\eta\omega$-reduction is locally confluent, since this is sufficient for confluence for well-founded reductions [1].

A.1. Termination

The logical relation proof of termination has three steps: We define a logical relation $\mathcal{L}$ that coincides with with the desired termination property $\mathcal{S}$ at base types and has good closure properties. Then we show that $\mathcal{L} \subseteq \mathcal{S}$ (Lemma 8) and that moreover $\mathcal{L}_\alpha(\mathbf{A})$ for all $\mathbf{A} \in pwff_\alpha(\Sigma)$

(Lemma 10). Together these two results yield the termination result for $\beta\eta\omega$-reduction (Theorem 8).

DEFINITION 22. (Logical Relation). A set $\mathcal{L} \subset pwff(\Sigma)$ of proper formulae is called a **logical relation**[22], iff for all types $\alpha = \gamma \to \delta$ and all $\mathbf{A} \in pwff_\alpha(\Sigma)$ we have $\mathcal{L}_\alpha(\mathbf{A})$, iff $\mathcal{L}_\delta(\mathbf{AC})$ for all $\mathbf{C} \in pwff_\gamma(\Sigma)$ with $\mathcal{L}_\gamma(\mathbf{C})$. Clearly, logical relations are totally determined by their values on base types. For any other $\mathcal{S} \subseteq pwff(\Sigma)$, we call the (unique) logical relation $\mathcal{L}$ that coincides with $\mathcal{S}$ on base types the **logical relation induced by** $\mathcal{S}$.

DEFINITION 23. (Admissible). Let $\mathcal{L}$ be a logical relation, then a head reduction

$$(\lambda Z_\gamma \mathbf{.} \mathbf{A}_\alpha)\mathbf{C} \longrightarrow^h_\beta [\mathbf{C}/Z]\mathbf{A}$$

is called **admissible**, iff $\mathcal{L}_\gamma(\mathbf{C})$. A logical relation $\mathcal{L}$ is said to be **admissible** if it is closed under admissible head expansions. (Formally: Let α be a type, and let $\mathbf{A}_\alpha, \mathbf{B}_\alpha$ be formulae with $\mathbf{A} \longrightarrow^h_\beta \mathbf{B}$, then a logical relation $\mathcal{L}$ is called **admissible** if $\mathcal{L}_\alpha(\mathbf{B})$ implies that $\mathcal{L}_\alpha(\mathbf{A})$.)

DEFINITION 24. (Terminating at $\mathbf{A}$).
We say that $\beta\eta\omega$-reduction is **terminating at A** *(we write $\mathcal{S}_\alpha(\mathbf{A})$, iff any sequence of $\beta\eta\omega$-reductions starting with $\mathbf{A}$ terminates. In the following we will use $\mathcal{L}$ for the logical relation induced by $\mathcal{S}$.*

LEMMA 7. *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 $\mathbf{C}$. □

LEMMA 8. ($\mathcal{L} \subseteq \mathcal{S}$). *Let h be a constant or free variable of type $\alpha = \overline{\alpha^n} \to \beta$, such that β is a base type then*

- *$\mathcal{S}_{\alpha^i}(\mathbf{A}^i)$ implies $\mathcal{L}_\beta(h\overline{\mathbf{A}^n})$, and*
- *$\mathcal{L}_\alpha(\mathbf{A})$ 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

[22] The name comes from the intuition that $\mathcal{L}$ is a relation between formulae and types.

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

$$h\overline{\mathbf{A}^n} = (\underbrace{f_\omega(f_\omega(\ldots(f_\omega}_{m}\mathbf{B})\ldots))\mathbf{A}^2\ldots\mathbf{A}^n)$$

where the head of $\mathbf{B}$ is not f_ω. We show the assertion by an induction over the number m, using the previous case for $m = 0$. 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 m and the third rule cannot apply at the top-level of $h\overline{\mathbf{A}^j}$ at all. Thus we have completed the proof of the first assertion in the base 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 lemma 7. □

LEMMA 9. *$\mathcal{L}$ is admissible.*

Proof. We have to show that $\mathcal{L}_\alpha([\mathbf{B}/Z]\mathbf{A})$ implies $\mathcal{L}_\alpha((\lambda Z_\beta\mathbf{.}\,\mathbf{A})\mathbf{B})$. 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 $\mathcal{S}_\delta((\lambda Z\mathbf{.}\,\mathbf{A})\mathbf{B}\overline{\mathbf{C}})$ ($\beta\eta\omega$-reduction is terminating at $(\lambda Z\mathbf{.}\,\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$), since δ is a base type.

So let us assume that $\mathcal{L}_\alpha([\mathbf{B}/Z]\mathbf{A})$, then by definition of $\mathcal{L}$ we have $\mathcal{L}_\delta([\mathbf{B}/Z]\mathbf{A}\overline{\mathbf{C}})$ and thus $\mathcal{S}_\delta([\mathbf{B}/Z]\mathbf{A}\overline{\mathbf{C}})$. In particular $\mathcal{S}_\alpha([\mathbf{B}/Z]\mathbf{A})$ and $\mathcal{S}_{\gamma^i}(\mathbf{C}^i)$ by lemma 7. Furthermore the head reduction is admissible and therefore $\mathcal{L}_\beta(\mathbf{B})$ and thus $\mathcal{S}_\beta(\mathbf{B})$ by lemma 8. Finally, $\beta\eta\omega$-reduction must be terminating at $\mathbf{A}$, since an infinite reduction from $\mathbf{A}$ would imply one from $[\mathbf{B}/Z]\mathbf{A}$ ($\beta\eta\omega$-reduction is invariant under instantiation). Thus there cannot be an infinite sequence of reductions from $(\lambda Z\mathbf{.}\,\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ that only contracts redexes from $[\mathbf{B}/Z]\mathbf{A}$ and the $\mathbf{C}^i$. Thus such a reduction sequence from $(\lambda Z\mathbf{.}\,\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ has the form

$$\begin{array}{lll}(\lambda Z\mathbf{.}\,\mathbf{A})\mathbf{B}\overline{\mathbf{C}} & \longrightarrow^*_{\beta\eta\omega} & (\lambda Z\mathbf{.}\,\mathbf{A}')\mathbf{B}'\overline{\mathbf{C}'} \\ & \longrightarrow^h_{\beta\eta\omega} & [\mathbf{B}'/Z]\mathbf{A}'\overline{\mathbf{C}'} \\ & \longrightarrow^*_{\beta\eta\omega} & \ldots\end{array}$$

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

$$[\mathbf{B}/Z]\mathbf{A}\overline{\mathbf{C}} \longrightarrow^*_{\beta\eta\omega} [\mathbf{B}'/Z]\mathbf{A}'\overline{\mathbf{C}'} \longrightarrow^*_{\beta\eta\omega} \ldots \qquad \square$$

We will now use the admissibility of $\mathcal{L}$ to ascertain that $\mathcal{L}_\alpha = \mathit{pwff}_\alpha(\Sigma)$. To make the proof go through, we have to prove the following stronger assertion.

LEMMA 10. *If θ is a substitution, such that $\mathcal{L}_\alpha(\theta(X^\alpha))$ for all $X \in \mathbf{Dom}(\theta^t)$, then $\mathcal{L}_\alpha(\theta(\mathbf{A}))$ for all $\mathbf{A} \in \mathit{pwff}_\alpha(\Sigma)$.*

Proof. We prove the assertion by induction on the structure of $\mathbf{A}$. If $\mathbf{A}$ is a constant, or a variable not in $\mathbf{Dom}(\theta)$, then we obtain the assertion by the first case of Lemma 8 ($n = 0$). If $\mathbf{A} = X^\alpha$, then $\mathcal{L}_\alpha(\mathbf{A})$ by assumption.

If $\mathbf{A} = \mathbf{B}_{\gamma\to\alpha}\mathbf{C}_\gamma$, then by inductive hypothesis we have $\mathcal{L}_{\gamma\to\alpha}(\theta(\mathbf{B}))$ and $\mathcal{L}_\gamma(\theta(\mathbf{C}))$ and therefore $\mathcal{L}_\alpha(\theta(\mathbf{BC}))$ since $\mathcal{L}$ is logical.

Finally, if $\mathbf{A} = \lambda X_\gamma.\mathbf{B}$, then we will show that $\mathcal{L}_\delta(\theta(\mathbf{A})\mathbf{C})$ for all $\mathbf{C} \in \mathit{pwff}_\gamma(\Sigma)$ with $\mathcal{L}_\gamma(\mathbf{C})$. Without loss of generality, we can assume that $X \notin \mathbf{Dom}(\theta)$. Now let $\theta' := \theta, [\mathbf{C}/X]$. Clearly, θ' meets the condition in the induction hypothesis, so we have $\mathcal{L}_\delta(\theta'(\mathbf{B}))$. Furthermore, $\theta(\mathbf{A})\mathbf{C} = (\lambda X_\gamma.\theta(\mathbf{B}))\mathbf{C}$ reduces to $\theta'(\mathbf{B})$ in an admissible head reduction and therefore we have $\mathcal{L}_\delta((\lambda X_\gamma.\mathbf{B})\mathbf{C}))$ by admissibility of $\mathcal{L}$ (Lemma 9). $\square$

Collecting the results above, we arrive at the desired termination result.

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

Proof. Let $\mathbf{A} \in \mathit{pwff}_\alpha(\Sigma)$ be an arbitrary formula, by lemma 10 we have $\mathcal{L}_\alpha(\mathbf{A})$ (using the empty substitution θ) and thus $\mathcal{S}_\alpha(\mathbf{A})$ by lemma 8. $\square$

A.2. Confluence

Since $\beta\eta\omega$-reductions are finite, we only need to prove local confluence to ensure that $\beta\eta\omega$-reduction is confluent.

LEMMA 11. (Local Confluence). *If $\mathbf{A} \longrightarrow_{\beta\eta\omega} \mathbf{A}'$ and $\mathbf{A} \longrightarrow_{\beta\eta\omega} \mathbf{A}''$ then there is a term $\mathbf{B}$ 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 $\mathbf{B}$ with $\mathbf{A}' \longrightarrow_{\beta\eta\omega}^{*} \mathbf{B}$ and $\mathbf{A}'' \longrightarrow_{\beta\eta\omega}^{*} \mathbf{B}$. As in [1] we use $\mathbf{M}[t]$ to denote a term with a specific subterm t inside a so-called context $\mathbf{M}$. Let $\mathbf{A} \longrightarrow_{\beta} \mathbf{A}'$ then $\mathbf{A} = \mathbf{M}[(\lambda Z\,.\,\mathbf{C})\mathbf{C}']$ and $\mathbf{A}' = \mathbf{M}[[\mathbf{C}'/X]\mathbf{C}]$. In case a ω-rule has been applied inside $\mathbf{C}$ then obviously the same rule is also applicable in $[\mathbf{C}'/X]\mathbf{C}$. Analogously an application of a ω-rule on $\mathbf{C}'$ can be simulated by $n \geq 0$ applications of the same ω-rule on each occurrence of $\mathbf{C}'$ in $[\mathbf{C}'/X]\mathbf{C}$. On the other hand suppose the ω-rule has been applied inside the context $\mathbf{M}$ then the same ω-rule is also applicable in $\mathbf{M}[[\mathbf{C}'/X]\mathbf{C}]$. Analogous considerations can be made in case of the η-rule.

Hence, we may restrict ourselves to the following conflicts:

- let $\mathbf{A} = \mathbf{M}[\lambda Z\,.\,(f_{\omega}^{\alpha\to\alpha} Z)]$ and $\mathbf{A}' = \mathbf{M}[f_{\omega}^{\alpha\to\alpha}]$ and $\mathbf{A}'' = \mathbf{M}[\lambda Z\,.\,Z]$. With $\mathbf{B} = \mathbf{A}'\ \mathbf{A}'' \longrightarrow_{\omega} \mathbf{B}$.

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

- let $\mathbf{A} = \mathbf{M}[(\lambda Z\,.\,Z)\mathbf{D}]$ and $\mathbf{A}' = \mathbf{M}[\mathbf{D}]$ and $\mathbf{A}'' = \mathbf{M}[(f_{\omega}^{\alpha\to\alpha}\mathbf{B})]$. Let $\mathbf{B} = \mathbf{D}$ then $\mathbf{A}'' \longrightarrow_{\omega} \mathbf{B}$.

- let $\mathbf{A} = \mathbf{M}[f_{\omega}^{(\alpha\to\alpha)\to(\alpha\to\alpha)}(\lambda Z\,.\,Z)]$, $\mathbf{A}' = \mathbf{M}[f_{\omega}^{(\alpha\to\alpha)\to(\alpha\to\alpha)} f_{\omega}^{\alpha\to\alpha}]$ and $\mathbf{A}'' = \mathbf{M}[\lambda Z\,.\,Z]$. Let $\mathbf{B} = \mathbf{M}[f_{\omega}^{\alpha\to\alpha}]$ then $\mathbf{A}' \longrightarrow_{\omega} \mathbf{B}$ and $\mathbf{A}'' \longrightarrow_{\omega} \mathbf{B}$. □

Given the termination and the local confluence of $\beta\eta\omega$-reduction, we can deduce the Church-Rosser property:

THEOREM 9. (Church-Rosser Theorem). *If* $\mathbf{A} \longrightarrow_{\beta\eta\omega}^{*} \mathbf{A}'$ *and* $\mathbf{A} \longrightarrow_{\beta\eta\omega}^{*} \mathbf{A}''$ *then there exits some term* $\mathbf{B}$ *such that* $\mathbf{A}' \longrightarrow_{\beta\eta\omega}^{*} \mathbf{B}$ *and* $\mathbf{A}'' \longrightarrow_{\beta\eta\omega}^{*} \mathbf{B}$.

Proof. By [1] termination and local confluence entail confluence.

B. Properties of the Skeleton

While the subterm compatibility of the *initial* skeleton is trivially given by its definition, the proof of the subterm compatibility of the skeleton is done in two steps. First we state the relation between the skeleton of an application and the skeletons of its arguments:

LEMMA 12. *For all* $\mathbf{A}, \mathbf{B} \in pwff_\alpha(\Sigma)$ *we have:*

$$\begin{array}{rcl} \Omega_{\mathcal{D}}(\mathbf{AB}) & = & \{(\mathbf{A'B'})\downarrow_{\beta\eta\omega} \mid \mathbf{A'} \in \Omega_{\mathcal{D}}(\mathbf{A}), \mathbf{B'} \in \Omega_{\mathcal{D}}(\mathbf{B})\} \text{ and} \\ \Omega_{\mathcal{D}}(\lambda Z\mathbf{.A}) & = & \{(\lambda Z\mathbf{.A'})\downarrow_{\beta\eta\omega} \mid \mathbf{A'} \in \Omega_{\mathcal{D}}(\mathbf{A})\} \end{array}$$

Proof. The proof depends basically on the confluence of $\beta\eta\omega$-reductions:

$$\begin{array}{rl} & \Omega_{\mathcal{D}}(\mathbf{AB}) = \{(\mathbf{A'B'})\downarrow_{\beta\eta\omega} \mid \mathbf{A'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}), \mathbf{B'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\} \\ = & \{(\mathbf{A'}\downarrow_{\beta\eta\omega} \mathbf{B'}\downarrow_{\beta\eta\omega})\downarrow_{\beta\eta\omega} \mid \mathbf{A'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}), \mathbf{B'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\} \\ = & \{(\mathbf{A'B'})\mid \mathbf{A'} \in \Omega_{\mathcal{D}}(\mathbf{A}), \mathbf{B'} \in \Omega_{\mathcal{D}}(\mathbf{B})\} \end{array}$$

The second statement is proven analogously:

$$\begin{array}{rl} & \Omega_{\mathcal{D}}(\lambda Z\mathbf{.A}) = \{(\lambda Z\mathbf{.A'})\downarrow_{\beta\eta\omega} \mid \mathbf{A'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A})\} \\ = & \{(\lambda Z\mathbf{.A'}\downarrow_{\beta\eta\omega})\downarrow_{\beta\eta\omega} \mid \mathbf{A'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A})\} \\ = & \{(\lambda Z\mathbf{.A'})\downarrow_{\beta\eta\omega} \mid \mathbf{A'} \in \Omega_{\mathcal{D}}(\mathbf{A})\} \end{array}$$

□

As a corollary of lemma 12 we obtain the subterm-compatibility of the skeleton:

COROLLARY 2. *For all* $\mathbf{A}, \mathbf{B}, \in pwff_\alpha(\Sigma)$:

$$\Omega_{\mathcal{D}}(\mathbf{A}|_\pi) = \Omega_{\mathcal{D}}(\mathbf{B}) \text{ implies } \Omega_{\mathcal{D}}(\mathbf{A}) = \Omega_{\mathcal{D}}([\mathbf{B}/\pi]\mathbf{A})$$

Now, we show that the skeleton is invariant with respect to $\beta\eta$-reduction. In a first step, to prove this property for the initial skeleton, we introduce the following lemma:

LEMMA 13. *For all* $\mathbf{A}, \mathbf{B} \in pwff_\alpha(\Sigma)$ *and* $@ \in \mathcal{V} \cup \mathcal{LV}$ *we have*

$$\widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{A}) = \{([\mathbf{B'}/@]\mathbf{A'})\mid \mathbf{A'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}), \mathbf{B'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}$$

Proof. We prove this lemma by induction on the structure of $\mathbf{A}$:

Let $\mathbf{A}$ be a symbol distinct from @ or $\mathbf{A} = @ \in \mathcal{LV}$, then

$$\widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{A}) = \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}) =_\beta \{([\mathbf{B'}/@]\mathbf{A'})\mid \mathbf{A'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}), \mathbf{B'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}$$

Let $\mathbf{A} = @ = X^\alpha_{\mathtt{a}}$, then $\widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}) = \{X\}$ or $\widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}) = f^{\alpha\beta}_\omega$ depending on whether $\mathtt{a} \in \mathcal{D}$ or not, thus we have

$$\widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{A}) = \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B}) = \{[\mathbf{B'}/@]\mathbf{A'}\mid \mathbf{A'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}), \mathbf{B'} \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})$$

Let $\mathbf{A} = \lambda W\,.\,\mathbf{C}$. With

$$\widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{C}) = \{[\mathbf{B}'/@]\mathbf{C}'|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}$$

as an induction hypothesis we deduce:

$$\begin{aligned}
&\widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{A}) = \widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@](\lambda W\,.\,\mathbf{C})) = \widetilde{\Omega}_{\mathcal{D}}(\lambda W\,.\,[\mathbf{B}/@]\mathbf{C})\\
=\ & \{(\lambda W\,.\,\mathbf{C}')|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{C})\}\\
=\ & \{(\lambda W\,.\,[\mathbf{B}'/@]\mathbf{C}')|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}\\
=\ & \{([\mathbf{B}'/@](\lambda W\,.\,\mathbf{C}'))|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}\\
=\ & \{([\mathbf{B}'/@]\mathbf{A}')|\ \mathbf{A}' \in \widetilde{\Omega}_{\mathcal{D}}(\lambda W\,.\,\mathbf{C}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}\\
=\ & \{([\mathbf{B}'/@]\mathbf{A}')|\ \mathbf{A}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}
\end{aligned}$$

Let $\mathbf{A} = (\mathbf{C}\mathbf{D})$. With

$$\begin{aligned}
\widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{C}) &= \{([\mathbf{B}'/@]\mathbf{C}')|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}\\
\widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{D}) &= \{([\mathbf{B}'/@]\mathbf{D}')|\ \mathbf{D}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{D}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}
\end{aligned}$$

as induction hypotheses we deduce:

$$\begin{aligned}
&\widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{A}) = \widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@](\mathbf{C}\mathbf{D})) = \widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{C}[\mathbf{B}/@]\mathbf{D})\\
=\ & \{(\mathbf{C}'\mathbf{D}')|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{C}), \mathbf{D}' \in \widetilde{\Omega}_{\mathcal{D}}([\mathbf{B}/@]\mathbf{D})\}\\
=\ & \{(\mathbf{C}'\mathbf{D}')|\ \mathbf{C}' \in [\mathbf{B}'/@]\widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in [\mathbf{B}'/@]\widetilde{\Omega}_{\mathcal{D}}(\mathbf{D}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}\\
=\ & \{[\mathbf{B}'/@](\mathbf{C}'\mathbf{D}')|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{D}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}\\
=\ & \{[\mathbf{B}'/@]\mathbf{A}'|\ \mathbf{A}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}\mathbf{D}), \mathbf{B}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\}
\end{aligned}$$

□

As a consequence we obtain the invariance of the initial skeleton wrt. $\beta\eta$-reduction:

LEMMA 14. *For all* $\mathbf{A}, \mathbf{B} \in \mathit{pwff}_{\alpha}(\Sigma)$:

$$\mathbf{A} =_{\beta\eta} \mathbf{B} \textit{ implies } \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}) =_{\beta\eta} \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})$$

Proof. We use the subterm-compatibility of $\widetilde{\Omega}_{\mathcal{D}}$ to prove this property by induction on the number of applications of $\beta\eta$-conversions necessary to equalize $\mathbf{A}$ and $\mathbf{B}$. By lemma 2 we only have to consider top-level reductions, thus we are left with the following cases:

$\longrightarrow_{\eta}$:

$$\begin{aligned}
&\widetilde{\Omega}_{\mathcal{D}}(\lambda Z\,.\,\mathbf{C}Z) = \{\lambda Z\,.\,\mathbf{C}'|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}Z)\}\\
=\ & \{(\lambda Z\,.\,\mathbf{C}''Z)|\ \mathbf{C}'' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C})\} =_{\eta} \{\mathbf{C}''|\ \mathbf{C}'' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C})\} = \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C})
\end{aligned}$$

$\longrightarrow_\beta$:

$$\begin{aligned} & \widetilde{\Omega}_{\mathcal{D}}((\lambda Z\,.\,\mathbf{C})\mathbf{D}) = \{(\mathbf{C}'\mathbf{D}')|\ \mathbf{C}' \in \widetilde{\Omega}_{\mathcal{D}}(\lambda Z\,.\,\mathbf{C}), \mathbf{D}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{D})\} \\ = \quad & \{(\lambda Z\,.\,\mathbf{C}'')\mathbf{D}'|\ \mathbf{C}'' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{D})\} \\ = \quad & \{([\mathbf{D}'/Z]\mathbf{C}'')|\ \mathbf{C}'' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{C}), \mathbf{D}' \in \widetilde{\Omega}_{\mathcal{D}}(\mathbf{D})\} =_\beta \widetilde{\Omega}_{\mathcal{D}}([\mathbf{D}/Z]\mathbf{C}) \end{aligned}$$

□

As an immediate consequence of the lemmata stated above, we obtain that also the skeleton is invariant wrt. $\beta\eta$-reduction:

COROLLARY 3. *For all* $\mathbf{A}, \mathbf{B} \in \mathit{pwff}_\alpha(\Sigma)$:

$$\mathbf{A} =_{\beta\eta} \mathbf{B} \textit{ implies } \Omega_{\mathcal{D}}(\mathbf{A}) = \Omega_{\mathcal{D}}(\mathbf{B})$$

Proof. Let $\mathbf{A} \longrightarrow_{\beta\eta} \mathbf{B}$ then lemma 14 guarantees $\widetilde{\Omega}_{\mathcal{D}}(\mathbf{A}) \longrightarrow_{\beta\eta} \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})$. Thus $\Omega_{\mathcal{D}}(\mathbf{A}) = \widetilde{\Omega}_{\mathcal{D}}(\mathbf{A})\downarrow_{\beta\eta\omega} = \widetilde{\Omega}_{\mathcal{D}}(\mathbf{B})\downarrow_{\beta\eta\omega} = \Omega_{\mathcal{D}}(\mathbf{B})$ holds. □