A Calculus for Rippling*

David A. Basin
Max-Planck-Institut für Informatik
Saarbrücken, Germany
Email: basin@mpi-sb.mpg.de

Toby Walsh
INRIA-Lorraine, 615, rue du Jardin Botanique,
54602 Villers-les-Nancy, France
Email: walsh@loria.fr

Abstract

Rippling is a special type of rewriting which uses annotations to guide the derivation towards a particular goal. It has many desirable properties. For example, it terminates yet allows rules like associativity to be used in both directions; it is also highly goal-directed applying only those rewrite rules needed to remove the differences between the goal and some hypothesis. Although it has been suggested that rippling can be directly implemented via first-order term rewriting on annotated terms, we show that this is not possible. We describe a simple calculus we have implemented for rippling and prove that it has the desired properties. This calculus allows us to combine rippling with conventional term rewriting. Such a combination offers the flexibility and uniformity of conventional rewriting with the highly goal-directed nature of rippling.

1 Introduction

Rippling is a type of rewriting developed at Edinburgh [5, 3] and independently by [8]. Although rippling was originally designed for inductive theorem proving, it has recently been applied with success to domains outwith inductive theorem proving [1, 9]. Rippling is rewriting tightly restricted by special kinds of annotation. Consider, for example, a proof of the associativity of multiplication using structural induction. In the step-case, the induction hypothesis is, $(x \times y) \times z = x \times (y \times z)$ and the induction conclusion is,

$$(\boxed{s(\underline{x})} \times y) \times z = \boxed{s(\underline{x})} \times (y \times z).$$

The annotations in the induction conclusion mark the differences with the induction hypothesis. Deleting everything in the box that is not underlined gives the *skeleton*; this is preserved during rewriting. By comparison, simply removing annotations from terms gives the *erasure*. The boxed but not underlined terms are *wave-fronts*; these are removed by rippling. The underlined parts are *wave-holes*; they represent terms in the wave-front we would like to leave unchanged. To remove the wave-fronts, we use annotated

*We thank Leo Bachmair, Alan Bundy, and Michael Rusinowitch. The first author was funded by the German Ministry for Research and Technology (BMFT) under grant ITS 9102. The second author was supported by a Human Capital and Mobility Postdoctoral Fellowship from the European Union. The current address for the second author is IRST, Location Panté di Povo, I38100 Trento, Italy.

rewrite rules called *wave-rules* which move wave-fronts in some well founded manner (usually towards the top of the term where cancellation can occur). In this example, we need the wave-rules,

$$\boxed{s(\underline{U})} \times V \quad \Rightarrow \quad \boxed{\underline{(U \times V)} + V} \tag{1}$$

$$\boxed{\underline{U} + V} \times W \quad \Rightarrow \quad \boxed{\underline{U \times W} + V \times W} \tag{2}$$

Rippling using (1) yields (3) and then with (2) gives (4).

$$(\boxed{\underline{x \times y} + y}) \times z \quad = \quad \boxed{\underline{(x \times (y \times z))} + y \times z} \tag{3}$$

$$\boxed{\underline{((x \times y) \times z)} + y \times z} \quad = \quad \boxed{\underline{(x \times (y \times z))} + y \times z} \tag{4}$$

As the wavefronts are now at the top of each term, we have successfully rippled-out both sides of the equality. We can complete the proof by simplifying with the induction hypothesis.

While the ideas behind rippling are intuitive, implementation is more complex than one might expect. In this paper we show where some of the difficulties lie and why first-order rewriting alone cannot directly implement rippling. We propose a new calculus that redefines the normal notions of substitution and matching and prove that it has the desired properties. To save space we restrict ourselves in several ways. First, we only consider so called "outward oriented" wave-rules (see [3]) and second we restrict ourselves to rippling in equational theories. Both restrictions are easily lifted.

2 Properties

Rippling may be viewed as first-order rewriting with the addition of annotation to restrict applicability of rewrite rules. In the CLAM system [4], rippling is implemented as first-order rewriting in a meta-level theory in which the signature of the original theory is extended with two new unary function symbols *wf* and *wh* (representing wave-fronts and wave-holes). In a well annotated term, every wave-front has at least one proper subterm that is a wave-hole. Terms in wave-holes may be further annotated.

Definition 1 (wats) *well annotated terms (or* wats*) are the smallest set such that,*

1. *t is a* wat *for all unannotated terms;*
2. *$wf(f(t_1,...,t_n))$ is a* wat *iff $\exists i.t_i = wh(s_i)$ and $\forall i.t_i = wh(s_i)$, s_i is a* wat *and $\forall i.t_i \neq wh(s_i)$, t_i is an unannotated term (*unat*);*
3. *$f(t_1,...t_n)$ is a* wat *where $f \neq wf$ and $f \neq wh$ iff $\forall i.t_i$ is a* wat.

To aid the reader, we will always write annotated terms using boxes and holes, though this is *just* meant to be syntactic sugar for *wf* and *wh*.

We also introduce two functions which return the skeleton and the erasure of a well annotated term.

Definition 2 (skeleton) *the function $skel : wats \rightarrow \mathcal{P}(wats)$ is defined by,*

1. *$skel(x) = \{x\}$ for all variables x;*
2. *$skel(f(t_1,...,t_n)) = \{f(s_1,...,s_n) | \forall i.s_i \in skel(t_i)\}$ where $f \neq wf$;*
3. *$skel(wf(f(t_1,...,t_n))) = \{s | \exists i.t_i = wh(t'_i) \wedge s \in skel(t'_i)\}$.*

Definition 3 (erasure) *the function $erase : wats \rightarrow unats$ is defined by,*

1. *$erase(x) = x$ for all variables x;*
2. *$erase(f(t_1,...,t_n)) = f(s_1,...,s_n)$ where $f \neq wf$ and $s_i = erase(t_i)$;*

3. $erase(wf(f(t_1,...,t_n))) = f(s_1,...,s_n)$ *where if* $t_i = wh(t_i')$ *then* $s_i = erase(t_i')$ *else* $s_i = erase(t_i)$.

Proofs constructed in this annotated meta-theory are translated into proofs in the original theory by erasing annotations using *erase*. Unfortunately this simple idea of rippling as first-order rewriting is not adequate. To show why, we define four properties desired of rippling.

well-formedness: rippling produces only well annotated terms (*wats*);

skeleton preservation: rippling preserves the skeleton of the term being rewritten;

correctness: rippling produces terms whose erasures are equal in the underlying equational theory;

termination: rippling terminates.

Unfortunately, it is not possible to achieve these aims by direct first-order rewriting. Consider a wave-rule formed for the recursive definition of multiplication,

$$\boxed{s(\underline{x})} \times y \Rightarrow \boxed{y + \underline{x \times y}}.$$

If we implement rippling with first-order rewriting this rule generates terms which are not well annotated. For example, $\boxed{s(\underline{a})} \times \boxed{s(\underline{b})}$ is a *wat* but rewrites to $\boxed{\boxed{s(\underline{b})} + \underline{a \times \boxed{s(\underline{b})}}}$ which is not a *wat* since the first argument of plus contains a wave-front (a box) directly inside another without an intermediate wave-hole.

Termination also fails using first-order rewriting. Consider, for example, the equation, $h(f(x, s(y))) = s(h(f(s(x), y)))$. This is a wave-rule in the forward direction (see [2]) annotated as

$$h(\boxed{f(x, s(\underline{y}))}) \quad \Rightarrow \quad \boxed{\underline{s(h(\boxed{f(s(x), \underline{y})}))}}.$$

This rule should not lead to non-termination; the rough intuition is that the skeleton of both terms is $h(y)$ and the wave-rule transforms a wave-front on the left-hand side that is two nested function symbols deep to one on the right-hand side at the same position that is only one function symbol deep and another wave-front at a higher position in the skeleton. This equation also constitutes a wave-rule in the reverse direction with different annotation.

$$\boxed{\underline{s(h(\boxed{f(s(\underline{x}), y)}))}} \quad \Rightarrow \quad h(\boxed{f(\underline{x}, s(y))})$$

Again this rule should not lead to non-termination; the intuition here is that the outermost wave-front is rippled "off the top" and disappears. However, these wave-rules together lead to cycling, as the following derivation illustrates:

$$h(\boxed{f(\underline{a}, s(\underline{a}))}) \;\mapsto\; \boxed{\underline{s(h(\boxed{f(s(\underline{a}), \underline{a})}))}} \;\mapsto\; h(\boxed{f(\underline{a}, s(\underline{a}))}) \;\mapsto\; \ldots .$$

Note that unlike the multiplication example, all the terms involved in this derivation are well annotated and share the same skeleton. With two equations, we can also construct looping derivations.

The problems of improperly annotated terms and non-termination arise when an annotated term t replaces an unannotated term in a wave-front. We therefore define a calculus for rippling based on a new notion of term replacement for annotated terms that erases annotation where appropriate; in this case, it erases the annotations on t. This new notion of term replacement naturally gives a new notion of substitution, and thus matching. By means of these simple modifications, we get a calculus for rewriting annotated terms which is guaranteed to preserve well-formedness of annotation, skeletons, and correctness wrt the underlying theory. Moreover, for unannotated terms and rewrite rules, this calculus performs conventional rewriting.

3 A Calculus for Rippling

We begin with ground rewriting and extend to first-order rewriting in a straightforward way. We distinguish between two kinds of variables: those in rewrite rules and those in terms. The later kind, "term variables" will be treated as constants here.[1] Hence we shall always consider the rewritten term as ground and by non-ground rewriting we mean equations between terms that may contain (non-term) variables.

We define a new function for subterm replacement $s[l] \mapsto s[r]$ which is identical to usual subterm replacement except if l occurs solely within a wave-front, the annotations on r are erased. For example, replacing a in $\boxed{f(a, \underline{b})}$ by $\boxed{g(\underline{a})}$ gives $\boxed{f(g(a), \underline{b})}$, but replacing b by $\boxed{g(\underline{b})}$ gives $\boxed{f(a, \underline{\underline{g(\underline{b})}})}$. We will perform all term replacement in substitution, and rewriting using this function. We will also restrict rewriting by only using *proper* equations which preserve skeletons.

Definition 4 (proper equation) *$l =_T r$ is a proper equation iff l and r are* wats, *$skel(l) = skel(r)$ and $erase(l) =_T erase(r)$.*

All the results given in this paper go through if we weaken the definition of proper equation to $skel(l) \supseteq skel(r)$ except rippling preserves now only a subset of the skeletons. We define *ground annotated rewriting* as rewriting using proper equations between ground terms. If s is a *wat* and $l =_T r$ a proper equation between ground *wats* then $s[l] \mapsto_G s[r]$ denotes the ground annotated rewriting of a subterm l of s.

We now show that ground annotated rewriting preserves the well-formedness of annotated terms, skeletons and erasure wrt the theory.

Theorem 1 *if s is a* wat, *$l =_T r$ a proper equation and $s[l] \mapsto_G s[r]$, then*

1. *$s[r]$ is a* wat,
2. *$skel(s[l]) = skel(s[r])$,*
3. *$erase(s[l]) =_T erase(s[r])$.*

Proof: By structural induction on the *wat* s.

Case 1. s is a (term) variable. The three properties hold trivially.

Case 2. $s = wf(f(s_1, ...s_n))$. Now l is a subterm of s. If it is s itself then the three properties hold trivially. On the other hand, suppose l is a subterm of some s_i. Either $s_i = wh(t_i)$ or $s_i \neq wh(t_i)$. In the first case, l is a subterm of the *wat* t_i as l is a *wat* and cannot itself be headed by *wh*. By the induction hypothesis, $t_i[r]$ is a *wat*. Thus, $s[r]$ is a *wat*. Also, by the induction hypothesis $skel(t_i[l]) = skel(t_i[r])$. As all other subterms $wh(s_j)$ are unchanged, the union of these skeletons is unchanged. Hence $skel(s[l]) = skel(s[r])$. Finally, by the induction hypothesis, $erase(t_i[l]) =_T erase(t_i[r])$. Again, as all the other subterms are unchanged, their erasures stay the same. Thus, $erase(s[l]) =_T erase(s[r])$. In the second case, $s_i \neq wh(t_i)$. From the definiton of *wat*, s_i must be an unannotated term. Thus, when we substitute r for l we will erase annotations on r. Hence, $s_i[r]$ is unannotated, and $s[r]$ is a *wat*. By the definition of skeleton, $skel(s[l]) = skel(s) = skel(s[r])$. Since s_i is unannotated and $l =_T r$ is a proper equation, $erase(s_i[l]) =_T erase(s_i[r])$. Hence, $erase(s[l]) = erase(s[r])$.

Case 3. Suppose $s = f(s_1, ...s_n)$ and f is not *wh* or *wf*. Now l is a subterm of s. If it is s itself then the three properties trivially hold. On the other hand, suppose l is a subterm of some s_i, i.e., $s_i[l]$. By the induction hypothesis, $s_i[r]$ is a *wat*. Hence $s[r]$ is a *wat*. By the induction hypothesis, $skel(s_i[l]) = skel(s_i[r])$. As $skel(s_j) = skel(s_j)$, we have by the definition of the skeleton function that $skel(s[l]) = skel(s[r])$. By the induction hypothesis, $erase(s_i[l]) =_T erase(s_i[r])$. As the other s_j have the same erasures, by the definition of erase, $erase(s[l]) =_T erase(s[j])$. □

By induction on number of rewrite steps, the reflexive transitive closure of $\mapsto_G$ also preserves well-formedness, skeletons and erasure wrt the theory.

[1] E.g., See [6] Section II.

4 Annotated Matching

We now consider rewriting with variables. Since substitution depends on subterm replacement, our new definition of subterm replacement gives rise to a new kind of substitution; in particular, during substitution terms replacing variables in wave-fronts are erased of annotation. For example, applying $\sigma = \{x/\boxed{s(\underline{a})}\}$ to $t = \boxed{f(x,\underline{x})}$ gives $\boxed{f(s(a),\boxed{s(\underline{a})})}$. We say that σ is a *well-annotated substitution* (*was*) iff for every $x_i/t_i \in \sigma$, t_i is a *wat*. Unlike regular substitution, this new substitution function preserves well-formedness, skeletons and erasure.

Theorem 2 *if t is a* wat *and σ a* was *then*

1. $\sigma(t)$ *is a* wat,
2. $skel(\sigma(t)) = (skel(\sigma))(skel(t))$,
3. $erase(\sigma(t)) = (erase(\sigma))(erase(t))$.

Proof: By structural induction on the *wat* t.

Case 1. t is a (term) variable x. The three properties trivially hold.

Case 2. Suppose $t = wf(f(t_1,...t_n))$. Either $t_i = wh(s_i)$ or $t_i \neq wh(s_i)$. In the first case, using the induction hypothesis and the fact that s_i is a *wat*, $\sigma(s_i)$ must also be a *wat*. In the second case, t_i must be unannotated. As unannotated terms are a subset of *wats*, calling upon the induction hypothesis, $\sigma(t_i)$ is a *wat*. Thus, $erase(\sigma(t_i))$ is unannotated. Hence, by the definition of annotated substitution, $\sigma(t)$ is a *wat*. By the defintion of skeleton, $skel(\sigma(t)) = \{s|s \in skel(\sigma(s_i))\}$. By the induction hypothesis this is $\{s|s \in (skel(\sigma))(skel(s_i))\}$. But this is equal to $(skel(\sigma))(\{s|s \in skel(s_i)\})$ which itself equals $(skel(\sigma))(skel(t))$. Hence, $skel(\sigma(t)) = (skel(\sigma))(skel(t))$. To prove the third property, we perform a case split on whether $t_i = wh(s_i)$ or $t_i \neq wh(s_i)$. In the first case, by the induction hypothesis, $erase(\sigma(s_i)) = (erase(\sigma))(erase(s_i))$. In the second case, t_i is unannotated and, by the induction hypothesis, $erase(\sigma(t_i)) = (erase(\sigma))(erase(t_i))$. By structural induction over *unats*, $erase(erase(\sigma(t_i))) = erase(\sigma(t_i))$. Thus, by the definition of erasure and annotated substitution, $erase(\sigma(t)) = (erase(\sigma))(erase(t))$.

Case 3. Suppose $t = f(t_1,...t_n)$ and f is not *wh* or *wf*. Now $\sigma(t) = f(\sigma(t_1),...\sigma(t_n))$. By the induction hypothesis, $\sigma(t_i)$ are *wats*. Thus $\sigma(t)$ is also a *wat*. By the definition of skeleton, $skel(f(\sigma(t_1),...\sigma(t_n)))= f(skel(\sigma(t_1)),...skel(\sigma(t_n)))$. This equals $f((skel(\sigma))(skel(t_1)),...(skel(\sigma))(skel(t_n)))$ by the induction hypothesis. By the definition of substitution, this is $(skel(\sigma))(f(skel(t_1),...skel(t_n)))$. Hence, $skel(\sigma(t))$ $= (skel(\sigma))(skel(t))$. By the definition of erasure, $erase(f(\sigma(t_1),...\sigma(t_n)))= f(erase(\sigma(t_1)),...erase(\sigma(t_n)))$. By the induction hypothesis this equals $f((erase(\sigma))(erase(t_1)),...(erase(\sigma))(erase(t_n)))$. By the definition of substitution, this is $(erase(\sigma))(f(erase(t_1),...erase(t_n)))$. Hence, $erase(\sigma(t)) = (erase(\sigma))(erase(t))$. □

To perform rewriting, we need a notion of annotated matching corresponding to this new notion of annotated substitution.

Definition 5 (annotated match) *if s and t are* wats, *then σ is an* annotated match *of s with t iff σ is a* was *and $\sigma(s) = t$.*

Observe that even if we restrict σ to variables in s annotated matching is not unitary. For example, $\{x/s(0)\}$ and $\{x/\boxed{s(\underline{0})}\}$ are both annotated matches of $\boxed{f(x,\underline{0})}$ with $\boxed{f(s(0),\underline{0})}$. Matches differ only in the amount of annotation which appear on substitutions for variables the occur in wave-fronts but not in skeletons. We can, however, define a notion of minimality of annotation so that annotated matching is unique. In the following, $annotations(s)$ is the set of addresses of waveholes in the wat s.

Definition 6 (minimality) *if σ_1 and σ_2 are* was *then $\sigma_1 \preceq \sigma_2$ iff for all $x/t \in \sigma_1$ there exists $x/s \in \sigma_2$ with $erase(t) = erase(s)$ and $annotations(t) \subseteq annotations(s)$.*

Definition 7 (minimal match) *if s and t are wats, then σ is a minimal match of s with t iff $Dom(\sigma) = vars(s)$, σ is an annotated match of s with t and there does not exists any annotated match $\sigma' \neq \sigma$ with $\sigma' \preceq \sigma$.*

Theorem 3 (uniqueness) *if σ is a minimal match of s with t then σ is unique.*

Proof: By contradiction. Let σ_1 and σ_2 be different minimal matches of s with t. Since $Dom(\sigma_1) = vars(s) = Dom(\sigma_2)$ and $\sigma_1 \neq \sigma_2$ there must be at least one variable x in $vars(s)$ on which σ_1 and σ_2 disagree. Let $x/s_1 \in \sigma_1$ and $x/s_2 \in \sigma_2$ where $s_1 \neq s_2$. By the well formedness of annotated substitution, $erase(s_1) = erase(s_2)$. Assume x occurs in the skeleton of s. By well formedness again, $skel(s_1) = skel(s_2)$. But this means that $s_1 = s_2$. Hence x can only occur in the wavefront but not in the skeleton. As σ_1 is minimally annotated, s_1 must therefore be an *unat*. Similarly, s_2 must be an *unat*. Thus $s_1 = s_2$. □We now give a complete and correct annotated matching procedure. The idea is to rename apart variables in the wavefront from their occurrences in the skeleton and to combine the substitutions at the end of regular matching provided the renamed vars have the appropriate erasure. To do this, we define two new functions, *convert* and *combine*.

Definition 8 (convert)

1. *$convert(x) = x$ for all variables x;*
2. *$convert(f(t_1, ..., t_n)) = f(convert(t_1), ...convert(t_n))$ where $f \neq wf$;*
3. *$convert(wf(f(t_1, ..., t_i))) = wf(f(s_1, ...s_n))$ where if $t_i = wh(t_i')$ then $s_i = wh(convert(t_i'))$ otherwise $s_i = new\text{-}vars(t_i)$.*

The function *new-vars* takes a term and replaces all variables in it by fresh ones which are guaranteed to be unique from all other variables. Different occurrences of the same variable are replaced by the *same* variable. If the set of variable renamings performed is given by ρ and we have a substitution τ returned from ordinary matching (using the renamed term) then we can combine these into a single substitution composed of two parts: σ_{skel} contains the substitutions for variables in the skeleton for the annotated match, whilst σ_{wf} contains the (unannotated) substitutions for the variables that occur just in wavefront.

Definition 9 (combine) *$combine(\rho, \tau) = \sigma_{skel} \cup \sigma_{wf}$ where,*

- $\sigma_{skel} = \{x/s \mid x/s \in \tau \ \wedge \ (y/x \in \rho \ \rightarrow \ y/erase(s) \in \tau)\}$
- $\sigma_{wf} = \{x/s \mid s \text{ is an unat} \ \wedge \ x \notin Dom(\tau) \ \wedge \ y/x \in \rho \ \wedge \ y/s \in \tau\}$

To perform annotated matching, we rename variables in the wavefront using *convert*, perform regular matching and then combine the resulting substitution with the renamings using the function *combine*. We define this process with the (partial) function *amatch*.

Definition 10 (amatch) *$amatch(s, t) = \sigma$ if $r = convert(s)$, $\rho(r) = s$, $\tau(r) = t$, $\sigma = combine(\rho, \tau)$, and $Dom(\sigma) = vars(s)$.*

Note that the annotated match can fail because $Dom(\sigma) \subset vars(s)$ indicating that the substitutions for renamed variables were incompatible. In such a situation, annotated matching fails.

Theorem 4 (correctness and completeness) *if s and t are* wats *then $amatch(s, t) = \sigma$ iff $\sigma(s) = t$ and σ is a minimal match.*

Proof: ($\Rightarrow$) Since $amatch(s, t) = \sigma$, we have $r = convert(s)$, $\rho(r) = s$, $\tau(r) = t$, $\sigma = combine(\rho, \tau)$, and $Dom(\sigma) = vars(s)$. Now σ is a *was* by construction. Consider $\sigma(s)$ and $\tau(r)$. By construction, these terms are identical except where variables occur in s. Consider an occurrence of a variable in the

skeleton of s. By the construction of σ, the same annotated substitution from σ_{skel} is applied to both. Consider an occurrence of a variable in the wavefront of s that does not appear in the skeleton. By the construction of σ, the same unannotated substitution from σ_{wf} is also applied to both. Finally, consider an occurrence of a variable in the wavefront of s that also appears in the skeleton. By the construction of σ, the same unannotated substitution from σ_{skel} is also applied to both. Thus $\sigma(s)$ is identical to $\tau(r)$. But $\tau(r) = t$. Hence $\sigma(s) = t$. In addition, by construction, σ is a minimal match for variables which appear just in the wavefront.

($\Leftarrow$) Let $\sigma(s) = t$. Let $r = convert(s)$, $\rho(r) = s$, and $\tau = \sigma \cup \tau_{wf}$ where $\tau_{wf} = \{x/erase(s) \mid x/y \in \rho \wedge y/s \in \sigma\}$. Consider, $\tau(r)$ and $\sigma(s)$. By construction, these terms are identical except where variables occur in s. Consider an occurrence of a variable in the skeleton of s. By the construction of τ, the same annotated substitution from σ is applied to both. Consider an occurrence of a variable in the wavefront of s. By the construction of τ_{wf}, the same unannotated substitution from is also applied to both. Thus $\tau(r)$ and $\sigma(s)$ are identical. Also, by construction, since σ is a minimal match then $\sigma = combine(\rho, \tau)$. Trivially, $Dom(\sigma) = vars(s)$. Hence $amatch(s, t) = \sigma$. □

Annotated matching can be performed, like regular matching, in linear time.

Theorem 5 (complexity) *annotated matching can be performed in $O(|t|)$ time where $|t|$ is the size of the term being matched against.*

Proof: Consider the annotated match of s with t. We will assume that annotated terms are represented by dags. First we rename the variables in the wavefront of s. This takes $O(|s|)$ time. At the position of each renaming, we add a new type of arc (called *erase*) linking the renamed variable with its original name. This arc represents the constraint on the erasure of substitution. In addition, it also flags that this is a renamed variables. (If a variable occurs only in the wavefront and not in the skeleton then we do not need to rename it). Next, we compare s with t from the root of both dags to the leaves of the pattern. This takes $O(|s|)$ time. Finally, we check that the substitutions which must be performed at the leaves for variables in wavefronts have unannotated substitutions and have an appropriate erasure. These tests can be performed together. We explore down t from the position where a renamed variable occurred in the pattern checking that no annotations occur. At the same time, by following the *erase* arc, we can explore down the skeleton of t skipping over wavefronts to ensure that we have the same erasure. At worst, this exploration will take $O(|t|)$ time since we are bounded by the size of t. Combining the three steps, we see that annotated matching can be performed in $O(|s|)+O(|t|)$ time. However, since there cannot be an annotated match for s with t unless $|s| \leq |t|$, this can be simplified to $O(|t|)$. □

5 First-Order Rippling

We can now show that rippling with proper equations is correct and specify sufficient conditions for termination. We do this by lifting the results of ground annotated rewriting to the first-order case. The previous definition of proper equation carries over without change. We use it to define non-ground rewriting. If s is a *wat* and $l =_T r$ a proper equation with $Vars(r) \subseteq Vars(l)$, then we write $s[\sigma(l)] \mapsto s[\sigma(r)]$ to indicate the annotated rewriting of a subterm of s with which l annotated matches giving σ. Correctness parallels the ground case. The proof relies on the fact that σ is a *was* and we can thus reduce the problem to the correctness of ground-rewriting.

Theorem 6 *if s is a* wat*, $l =_T r$ a proper equation, and $s[\sigma(l)] \mapsto s[\sigma(r)]$, then*

1. *$s[\sigma(r)]$ is a* wat*,*
2. *$skel(s[\sigma(l)]) = skel(s[\sigma(r)])$,*
3. *$erase(s[\sigma(l)]) =_T erase(s[\sigma(r)])$.*

Proof: Annotated matching guarantees that σ is a *was*. By Theorem 2, $\sigma(l)$ and $\sigma(r)$ are *wats* with the same skeleton and erasures. Now, as $\sigma(l)$ is syntactical identical to a subterm of s it is ground (recall discussion about treating "term variables" as constants). Furthermore $\sigma(r)$ is also ground as to be a rewrite rule we require $Vars(r) \subseteq Vars(l)$. Hence the rewriting of $s[\sigma(l)] \mapsto s[\sigma(r)]$ is equivalent to that performed by the ground rewrite rule $\sigma(l) \to \sigma(r)$ which is proper as $l =_T r$ and Theorem 2 tells us the three properties are maintained for any σ. Thus, by Theorem 1, the three properties hold. □

Let $\overset{*}{\mapsto}$ be the reflexive transitive closure of $\mapsto$. By induction on number of rewrite steps, it follows from Theorem 6 that annotated rewriting is correct. That is, if we erase all annotations, we can perform the same (object-level) rewriting. Annotations merely guide rewriting in a skeleton preserving way.

Theorem 7 (correctness) *if s is a* wat *and $s \overset{*}{\mapsto} t$ then*

1. *t is a* wat,
2. *$skel(s) = skel(t)$,*
3. *$erase(s) =_T erase(t)$.*

6 Termination

We can now turn towards questions of termination. A simple way to ensure termination of ground annotated rewriting is to rewrite only with wave rules which are measure decreasing under some well founded order which is monotonic with respect to *wats*.

Definition 11 (monotonicity wrt *wats*) *an order $>$ is monotonic wrt* wats *iff for all* wats *l and r such that $l > r$, $s[l] \mapsto_G s[r]$ implies $s[l] > s[r]$.*

Note that this is weaker than the usual definition of monotonicity since we do not consider all possible subterms of s, only those that are *wats*. The orders in [2, 3], for example, are monotonic wrt *wats*.

Theorem 8 (ground termination) *for a well founded order $>$ monotonic wrt* wats*, ground annotated rewriting using rewrite rules $l \Rightarrow r$ for which $l > r$ is terminating.*

Proof: An infinite rewrite sequence $t_1 \Rightarrow t_2 \Rightarrow \ldots$ gives rise to an infinite sequence of terms $t_1 > t_2 >$ which contradicts the well foundedness of $>$. □

Note that the preservation of skeletons is important for termination since proposed termination orders for rippling (*eg.* those in [2, 3]) depend upon the skeleton remaining constant (or bounded in size).

Finally we lift our requirements on termination to non-ground rewriting. To do this, we need a restricted form of stability.

Definition 12 (stable wrt *wats*) *an order $>$ is stable wrt* wats *iff for all* wats *s, t and any* was *σ, $s > t$ implies $\sigma(s) > \sigma(t)$.*

The orders in [2, 3] based on the width (but not size) of the wavefronts are stable wrt *wats*. It follows from our final theorem that rippling with the wave-rules given there is terminating.

Theorem 9 (termination) *for a well founded-order $>$ monotonic and stable wrt* wats*, annotated rewriting using proper rewrite rules $l =_T r$ for which $l > r$ is terminating.*

Proof: We again reduce the problem to the ground case. If $s \mapsto t$ using $l =_T r$ then there is a ground instance $\sigma(l) =_T \sigma(r)$ and as $>$ is stable wrt *wats*, $\sigma(l) > \sigma(r)$. By the termination of ground annotated rewriting, we have termination in the general case. □

7 Conclusions and Related Work

The implementation of rippling in CLAM uses first-order rewriting restricted by some precondition checking. It is considerably more restricted than what we propose here, yet leads to ill-formed annotation and potentially non-terminating deduction. The INKA system [8] uses a formalized calculus for rippling but it is considerably more complex and specialized. In addition, the termination of rippling has not been considered within this framework.

Our results show that conventional term rewriting systems can be used to perform rippling via a simple modification to the routines for subterm replacement and matching. Moreover, the same routines can be used for annotated and conventional unannotated term rewriting. We have implemented this calculus and we will describe some applications. Many possible directions can now be explored: for example, rippling in an implicit induction setting, Knuth-Bendix completion of annotated terms, simplification of skeletons and wave-fronts using term rewriting. We speculate that such combinations will offer the best of both worlds, the flexibility and uniformity of conventional rewriting procedures combined with the highly goal-directed nature of rippling.

References

[1] D. Basin and T. Walsh. Difference unification. In *Proceedings of the 13th IJCAI*, pages 116–122, 1993.

[2] D. Basin and T. Walsh. Termination Orderings for Rippling. To appear in A. Bundy, editor, *Proceedings of CADE-12*, 1994.

[3] 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.

[4] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In M.E. Stickel, editor, *Proceedings of CADE-10*, Springer-Verlag, 1990.

[5] A. Bundy, F. van Harmelen, A. Smaill, and A. Ireland. Extensions to the rippling-out tactic for guiding inductive proofs. In M.E. Stickel, editor, *Proceedings of CADE-10*, pages 132–146. Springer-Verlag, 1990.

[6] N. Dershowitz. Termination of rewriting. In J.-P. Jouannaud, editor, *Rewriting Techniques and Applications*, pages 69–116. Academic Press, 1987.

[7] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B. North-Holland, 1990.

[8] D. Hutter. Guiding inductive proofs. In M.E. Stickel, editor, *Proceedings of CADE-10*, pages 147–161. Springer-Verlag, 1990.

[9] T. Walsh, A. Nunes, and A. Bundy. The use of proof plans to sum series. In D. Kapur, editor, *Proceedings of CADE-11*, pages 325–339. Springer Verlag, 1992.