# A Calculus for and Termination of Rippling

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

and

Toby Walsh †
*Mechanized Reasoning Group, DIST, Genova and IRST, Trento, Italy*
*email: toby@irst.it*

**Abstract.** Rippling is a type of rewriting developed for inductive theorem proving which uses annotations to direct search. Rippling has many desirable properties: for example, it is highly goal directed, usually involves little search, and always terminates. In this paper we give a new and more general formalization of rippling. We introduce a simple calculus for rewriting annotated terms, close in spirit to first-order rewriting, and prove that it has the formal properties desired of rippling. Next we develop criteria for proving the termination of such annotated rewriting, and introduce orders on annotated terms that lead to termination. In addition, we show how to make rippling more flexible by adapting the termination orders to the problem domain. Our work has practical as well as theoretical advantages: it has led to a very simple implementation of rippling that has been integrated in the Edinburgh CLAM system.

**Key words:** Mathematical Induction, Inductive Theorem Proving, Term Rewriting

## 1. Introduction

Rippling is a form of rewriting developed by Bundy *et al* [6, 8] which uses annotations to restrict rewriting and to guide the derivation towards a particular goal. Rippling applies naturally in inductive theorem proving where the induction conclusion typically differs from the induction hypothesis by the addition of some constructors or destructors. These differences are marked by annotations. Rippling uses annotated rewrite rules, called *wave-rules*, to move these marked differences; when successful, the differences are either removed completely or moved to positions like the top of the term that enable the use of the induction hypothesis.

Rippling has many attractive properties. It is highly goal directed, manipulating just the differences between the induction conclusion and hypothesis while leaving their common structure preserved; this is in contrast to rewriting based on normalization, which is used in other inductive theorem provers such as NQTHM [4]. Rippling also involves little search since annotations severely restrict rewriting.

* Funded by the German Ministry for Research and Technology under grant ITS 9102.

† Supported by a Human Capital and Mobility Research Fellowship from the European Commission. Both authors thank members of the Edinburgh Mathematical Reasoning Group, as well as Alan Bundy, Leo Bachmair, Dieter Hutter, and Michael Rusinowitch for their comments on previous drafts. Additional support also received from the MInd grant EC-US 019-76094.

Indeed, the use of annotation places such strong restrictions on the search space that it is often possible to analyze failed rippling proofs and to suggest missing lemmas or generalizations [15]

In this paper we give a new account of rippling and its properties that is both substantially simpler and more general than previous accounts. Conceptually, the starting point of our work is the formal presentation of rippling given by Bundy *et al.* in [6]. There rippling is presented as first-order rewriting restricted by some simple preconditions that ensure that annotations in the subterm being rewritten match annotations in the applied wave-rule. We show that this is inadequate. Taken literally, it leads to an implementation of rippling too restrictive to carry out the derivations given in [6] (see §8). When the restrictions are loosened (as they were in the implementation of the CLAM system [7]) other problems arise like improperly annotated terms. We give a simple calculus for rippling which does not suffer from these problems.

We also simplify, improve, and generalize the specification of wave-rules and their associated termination orderings. Wave-rules have previously been presented using complex schematic definitions that combine the properties of structure preservation and the reduction of a well-founded measure (see §8). Our definition of wave-rules separates these two concerns and their demonstration. We present measures that, despite their simplicity, admit strictly more wave-rules than the considerably more complex specification given in [6].

Another contribution of our work is to provide new termination orderings which extend the power of rippling. Although rippling was designed primarily to prove inductive theorems, it has recently been applied to other problem domains. For example, it has been used to sum series [16], to prove limit theorems [17], and to perform normalization [1]. In rippling, as in conventional rewriting, the termination ordering can be made domain dependent. We illustrate this idea by two new orderings.

A practical contribution of our work is that it greatly simplifies implementation; our calculus leads to an implementation of rippling in the spirit of standard first-order rewriting. Moreover, systems implementing rippling require a procedure which annotates rewrite rules (here called a *wave-rule parser*) and our work eases the construction of such a routine. We report on our implementation of these routines which has been integrated into the CLAM system.

The remainder of this paper is organized as follows. §2 provides a background on rippling, in particular on annotation and wave-rules. §3 formalizes properties of rippling and shows why first-order rewriting cannot directly satisfy these properties. §4 presents a new calculus that does satisfy these properties. Next, §5 introduces reduction orders under which rippling will terminate. §6 describes how our calculus can be implemented and how unannotated rewrite rules can be annotated and oriented to give wave-rules. §7 presents two new orders which extend the power of rippling. Finally, in §8 we survey related work.

## 2. Annotation and Wave-rules

In this section, we give a brief overview of rippling and introduce notation and terminology. [6] should be consulted for additional motivation and examples.

Rippling arose out of an analysis of inductive proofs, and of the heuristics embedded within the NQTHM theorem prover [4]. As a simple example, suppose we wish to prove $P(x)$ for all natural numbers, $x$. We assume $P(n)$ and attempt to show $P(s(n))$. The hypothesis and the conclusion are identical except for the successor function $s(.)$ applied to the induction variable $n$. Rippling marks this difference by the annotation, $P(\boxed{s(\underline{n})}^{\uparrow})$. The annotation in the induction conclusion, given by the box, arrow and underlining, marks the differences with the induction hypothesis. Deleting the arrow and everything in the box that is not underlined gives the *skeleton*, $P(n)$; this is identical to the induction hypothesis, and is preserved during rippling. By comparison, simply removing annotations gives the *erasure*, $P(s(n))$. The boxed but not underlined term parts are *wave-fronts*; these are moved and transformed (and possibly deleted) by rippling. The underlined parts are *wave-holes*; they represent terms in the wave-front that we wish to leave unchanged. Wave-fronts are marked with arrows indicating if rippling should try to move the wave-front up through the skeleton term tree or down towards the leaves. Wave-fronts with an up arrow are called *outward* directed, whilst those with a down arrow are *inward* directed.

A wave-front can be viewed as a context, that is, it is a term with one, or more, proper subterms deleted. Schematically, an outward directed wave-front is of the form $\boxed{\xi(\underline{\mu_1}, \ldots, \underline{\mu_n})}^{\uparrow}$ where $n > 0$ and the $\mu_i$ may be similarly annotated; when $n = 1$ we call the wave-front *simply annotated* and when $n > 1$ we call it *multi-hole annotated*. A term is said to be simply annotated when all its wave-fronts are simply annotated, and is multi-hole annotated otherwise.

To formalize rippling, we extend the signature of the original theory with three new unary function symbols: *wfout*, *wfin* and *wh* (representing outward and inward directed wave-fronts and wave-holes respectively). The requirement that a context has at least one hole and that terms in these holes may be further annotated can be formally captured by defining the set of well-annotated terms with respect to a set of unannotated (first-order) terms *unats*.

DEFINITION 1. *Well-annotated terms* (or *wats*) are the smallest set such that,

1. $t$ is a *wat* for all unannotated terms;
2. $wfout(f(t_1, \ldots, t_n))$ is a *wat* iff for at least some $i$, $t_i = wh(s_i)$ and for each $i$ where $t_i = wh(s_i)$, $s_i$ is a *wat* and for each $i$ where $t_i \neq wh(s_i)$, $t_i$ is an unannotated term;
3. $wfin(f(t_1, \ldots, t_n))$ is a *wat* under the same conditions in case (2).
4. $f(t_1, \ldots, t_n)$ is a *wat* where $f$ is not *wfout*, *wfin*, or *wh* iff each $t_i$ is a *wat*.

For example, $wfout(s(wh(x))) \times wfout(s(wh(y)))$ represents the well-annotated term $\boxed{s(\underline{x})}^{\uparrow} \times \boxed{s(\underline{y})}^{\uparrow}$. To simplify the presentation of annotated terms, we shall still use boxes, arrows and holes. However, this is *just* syntactic sugar for *wfout*, *wfin* and *wh*. Note that in our formalization of annotation, wave-holes occur as immediate subterms of the function symbol in the wave-front. As additional syntactic sugar, we may merge adjacent wave-fronts and wave-holes when displaying annotated terms. For example, we will display the term $\boxed{s(\underline{\boxed{s(\underline{x})}^{\uparrow}})}^{\uparrow}$ as the annotated term $\boxed{s(s(\underline{x}))}^{\uparrow}$. Insisting that wave-fronts are "maximally split" both simplifies the presentation of rippling that follows and leads to a simpler implementation since wave-fronts do not need to be dynamically split or merged during rippling, as in [6].

We define the skeleton of an annotated term to be the set of unannotated terms formed by deleting function symbols and variables within wave-fronts that are not within wave-holes.

DEFINITION 2. The *skeleton function* $skel : wats \to \mathcal{P}(unats)$ is defined by,

1. $skel(x) = \{x\}$ for all variables $x$;
2. $skel(\boxed{f(t_1, \ldots, t_n)}^{\uparrow}) = \{s \mid \exists i.t_i = \underline{t'_i} \wedge s \in skel(t'_i)\}$;
3. $skel(\boxed{f(t_1, \ldots, t_n)}^{\downarrow}) = \{s \mid \exists i.t_i = \underline{t'_i} \wedge s \in skel(t'_i)\}$;
4. $skel(f(t_1, \ldots, t_n)) = \{f(s_1, \ldots, s_n) \mid \forall i.s_i \in skel(t_i)\}$.

For example, the skeleton of the annotated term

$$s(\boxed{\underline{\boxed{s(\underline{a})}^{\uparrow}} + (\underline{b \times c})}^{\uparrow}) \tag{1}$$

is the set $\{s(a), s(b \times c)\}$. Note that the skeleton of a simply annotated term is a singleton set; in this case, we refer to the member as *the skeleton* of the term.

By erasing annotation, we construct the corresponding unannotated term.

DEFINITION 3. The *erasure function* $erase : wats \to unats$ is defined by,

1. $erase(x) = x$ for all variables $x$;
2. $erase(\boxed{f(t_1, \ldots, t_n)}^{\uparrow}) = f(s_1, \ldots, s_n)$ where if $t_i = \underline{t'_i}$ then $s_i = erase(t'_i)$ else $s_i = t_i$;
3. $erase(\boxed{f(t_1, \ldots, t_n)}^{\downarrow}) = f(s_1, \ldots, s_n)$ where if $t_i = \underline{t'_i}$ then $s_i = erase(t'_i)$ else $s_i = t_i$;
4. $erase(f(t_1, \ldots, t_n)) = f(s_1, \ldots, s_n)$ where $s_i = erase(t_i)$.

For example, the erasure of (1) is $s(s(a) + (b \times c))$.

## 2.1. Wave-rules and Rippling

*Wave-rules* are formally defined in §4.4. Informally, they are rewrite rules between annotated terms that are skeleton preserving and measure decreasing under an appropriate ordering on annotated terms. This definition is simpler and more general to the one given in [6] where these two requirements were combined in the syntactic specification of a wave-rule.[1]

*Skeleton preservation* in the simply-annotated case means that both the LHS (left-hand side) and RHS (right-hand side) of the wave-rule have an identical skeleton. In the multi-hole case we demand that *some* of the skeletons on the LHS are preserved on the RHS and no new skeletons are introduced, i.e. $skel(LHS) \supseteq skel(RHS)$. *Measure reduction* and orderings for annotated terms will be described in considerable detail later. The intuition is that the position and orientation of wave-fronts define a measure. Rippling makes well-founded progress by moving annotations to decrease this measure; for example, moving outwards directed wave-fronts upwards in the rewritten term.

To illustrate the different types of wave-rules that our definition captures and to help motivate the definition of termination orders on annotated terms, we give some examples of wave-rules used by rippling. To begin with, most recursive function definitions and all primitive recursive function definitions can be annotated as wave-rules. For example, the recursive definition of times gives the wave-rule

$$\boxed{s(\underline{U})}^{\uparrow} \times V \;\rightarrow\; \boxed{(\underline{U \times V}) + V}^{\uparrow}. \tag{2}$$

Lemmas also can be often annotated as wave-rules, for example, algebraic laws like associativity, distributivity, and cancellation ($<>$ is infix append on lists).

$$(\boxed{\underline{U} + V}^{\uparrow}) \times W \;\rightarrow\; \boxed{\underline{U \times W} + V \times W}^{\uparrow} \tag{3}$$

$$\boxed{(U <> \underline{V})}^{\uparrow} <> W \;\rightarrow\; \boxed{U <> (\underline{V <> W})}^{\uparrow} \tag{4}$$

$$U <> (\boxed{\underline{V} <> W}^{\uparrow}) \;\rightarrow\; \boxed{\underline{(U <> V)} <> W}^{\uparrow} \tag{5}$$

$$\boxed{\underline{U} + \underline{V}}^{\uparrow} = \boxed{\underline{W} + \underline{Z}}^{\uparrow} \;\rightarrow\; \boxed{\underline{U = W} \wedge \underline{V = Z}}^{\uparrow} \tag{6}$$

(4) and (5) demonstrate that rules like associativity can be wave-rules in both directions; the precondition on rippling that annotations in wave-rules match those in the rewritten term prevent looping. (6) is an example of a wave-rule with multi-hole annotation. Note that $\rightarrow$ indicates rewriting, not implication; in fact, implication in (6) holds only in the reverse direction. This is sensible since in inductive theorem

[1] This generalization is, however, briefly discussed in their further work section.

proving, rippling reasons backwards from the induction conclusion to the induction hypothesis and rewrite rules based on implication as opposed to equivalence may be used; see [6] for further discussion on how such rewrite rules may be used in forward directed theorem proving.

Wave-rules can also ripple wave-fronts inwards, or change the orientation of wave-fronts from out to in (but not in to out).

$$\boxed{\underline{(U <> V)} <> W}^{\downarrow} \rightarrow U <> (\boxed{\underline{V} <> W}^{\downarrow}) \tag{7}$$

$$(\boxed{\underline{U} <> V}^{\uparrow}) <> W \rightarrow U <> (\boxed{V <> \underline{W}}^{\downarrow}) \tag{8}$$

A simple example of a proof guided by rippling is the proof of the associativity of multiplication

$$(x \times y) \times z = x \times (y \times z)\,. \tag{9}$$

The proof uses structural induction on $x$. In the step-case, (9) is the induction hypothesis and the induction conclusion is

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

The wave-fronts in the induction conclusion mark the differences with the induction hypothesis. Rippling on both sides of the induction conclusion using (2) yields

$$(\boxed{\underline{x \times y} + y}^{\uparrow}) \times z = \boxed{\underline{(x \times (y \times z))} + y \times z}^{\uparrow}\,.$$

Rippling with (3) on the left-hand side then gives

$$\boxed{\underline{(x \times y) \times z} + y \times z}^{\uparrow} = \boxed{\underline{x \times (y \times z)} + y \times z}^{\uparrow}\,.$$

As expected, the skeleton in each step of our proof is the induction hypothesis. At the end of this rewriting, the wave-fronts are at the root of each term. We can therefore complete the proof by simplifying with the induction hypothesis.

Rippling outward directed wave-fronts up towards the root of terms, like in the previous example, is called *rippling-out*. Wave-rules can also ripple wave-fronts downwards towards the position of universally quantified variables in the induction hypothesis. Such positions are called *sinks* because wave-fronts can be "absorbed" there: when we appeal to the induction hypothesis, universally quantified variables will be matched with the content of the sinks; an example of this is provided in §7.1. Rippling downward directed wave-fronts towards the leaves of the term tree using rules like (7) is called *rippling-in*.

## 3. Properties of Rippling

We now formalize the properties desired of rippling which motivate our calculus.

**Well-formedness:** if $s$ is a *wat*, and $s$ ripples to $t$, then $t$ is also a *wat*;

**Skeleton preservation:** if $s$ ripples to $t$ then $skel(t) \subseteq skel(s)$;

**Correctness:** if $s$ ripples to $t$ then $erase(s)$ rewrites to $erase(t)$ in the original (unannotated) theory;

**Termination:** rippling terminates.

The first property means that terms manipulated by rippling stay well-formed. Hence we can always compute their skeleton and erasure. Skeleton preservation ensures that rippling directs the derivation towards (at least) one of the induction hypotheses. Correctness guarantees that we can perform the corresponding derivation in the underlying object-level theory; annotation merely guides search. Finally, termination is important for practical considerations; it means we can try other possibilities (e.g., other inductions) when derivations fails.

Unfortunately, rippling implemented directly by first-order rewriting in an annotated theory fails to achieve these properties. We demonstrate this negative result by example. Consider using first-order rewriting to apply the wave-rule corresponding to the recursive definition of multiplication given in (2) to the term $\boxed{s(\underline{x})}^{\uparrow} \times \boxed{s(\underline{y})}^{\uparrow}$. This is a *wat* but rewrites to $\boxed{\underline{x \times \boxed{s(\underline{y})}^{\uparrow}} + \boxed{s(\underline{y})}^{\uparrow}}^{\uparrow}$ which is not a *wat* since the second argument of plus contains a wave-front (a box) directly inside another without an intermediate wave-hole. The reader may find it easier to carry out the rewrite step and verify this by removing the syntactic sugar and representing the boxes and holes explicitly with *wfout* and *wh*.

Termination also fails using first-order rewriting directly. Consider, for example, the equation $h(f(U, s(V))) = s(h(f(s(U), V)))$. Under the width measure given in §5.2, this is a wave-rule in the left-to-right direction annotated as

$$h(\boxed{f(U, s(\underline{V}))}^{\uparrow}) \;\rightarrow\; \boxed{s(\underline{h(\boxed{f(s(U), \underline{V})}^{\uparrow})})}^{\uparrow}.$$

It is also a wave-rule in the reverse direction with the annotation

$$\boxed{s(\underline{h(\boxed{f(s(\underline{U}), V)}^{\uparrow})})}^{\uparrow} \;\rightarrow\; h(\boxed{f(\underline{U}, s(V))}^{\uparrow}).$$

However, these wave-rules together lead to cycling as follows:

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

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

Both the problems of improperly annotated terms and non-termination arise when an annotated term replaces an unannotated term in a wave-front. This observation motivates our definition of a calculus for rippling. We introduce a new notion of term replacement that erases annotation when replacing terms in a wave-front. The redefinition of term replacement naturally gives rise to a new notion of substitution, and thus matching. By means of these simple modifications, we develop a calculus for rewriting annotated terms which is guaranteed to preserve well-formedness of annotation, skeletons, and correctness with respect to the underlying theory. In addition, this calculus allows us to design simple termination measures which are stable and monotonic. For unannotated terms and rewrite rules, our calculus performs conventional rewriting.

## 4. A Calculus for Annotated Rewriting

*Notational Convention:* To simplify notation in proofs, in this section we assume that the arguments of an annotated term like $\boxed{f(t_1,\ldots,t_n)}^{\uparrow}$ may be partitioned so that the first $j$ arguments are headed by wave-holes, i.e., of the form $t_i = \underline{t'_i}$ for $i \in \{1,\ldots,j\}$, and the last $n-j$ are unannotated. Hence the term may be written as $\boxed{f(\underline{t_1},\ldots,\underline{t_j},t_{j+1},\ldots,t_n)}^{\uparrow}$, or even $\boxed{f(\underline{t_1},\ldots,t_n)}^{\uparrow}$. This is without loss of generality as the proofs below do not depend on the order of wave-holes.

### 4.1. Ground Rewriting

We first consider rewriting using ground rewrite rules. As is typical (e.g., see [11] Section II), we distinguish between two kinds of variables: those in rewrite rules and those in terms. We treat the later kind, "term variables", as constants.

We begin by redefining subterm replacement. Let $s[l]$ represent a *wat* with a distinguished subterm $l$ that is also a *wat*. Let $rep(s[l],r)$ denote subterm replacement of $r$ for the distinguished occurrence of $l$ in the term $s$; $rep$ is defined identically to the usual subterm replacement of $l$ by $r$ in $s$ except that if $l$ occurs within a wave-front any annotations on $r$ are erased before replacement. For example, replacing $a$ in $\boxed{\underline{b}+s(a)}^{\uparrow}$ by $\boxed{s(\underline{a})}^{\uparrow}$ gives $\boxed{\underline{b}+s(s(a))}^{\uparrow}$, but replacing $b$ by $\boxed{s(\underline{b})}^{\uparrow}$ gives $\boxed{\underline{\boxed{s(\underline{b})}^{\uparrow}}+s(a)}^{\uparrow}$. From now on, we will perform all term replacement (including that occurring during substitution) using this function.

In rewriting terms, we will use just *proper* rewrite rules.

DEFINITION 4. $l \to r$ is a *proper rewrite rule* when $erase(l) \to erase(r)$ is a rewrite rule, $l$ and $r$ are *wats*, and $skel(r) \subseteq skel(l)$.

The last two requirements are needed for well-formedness and skeleton preservation. Note also that the requirement that $erase(l) \rightarrow erase(r)$ is a rewrite rule means that $Vars(r) \subseteq Vars(l)$.

Ground rewriting consists of rewriting using proper rewrite rules that contain no (non-term) variables. Let $R$ be a set of rewrite rules that are (for now) ground. If $s[l]$ is a *wat* then ground rewriting with a rule $l \rightarrow r$ in $R$ yields $rep(s[l], r)$; we use $s[l] \mapsto_R s[r]$ to denote such *ground rippling*. In what follows we will assume a particular rewrite rule set $R$, and drop subscripted references, e.g., writing simply $s[l] \mapsto s[r]$.

Ground rippling preserves the well-formedness of annotated terms, preserves skeletons, and corresponds to an annotated version of rewriting in the underlying (unannotated) theory.

THEOREM 1. *if $s$ is a* wat*, $l \rightarrow r$ a proper rewrite rule between ground* wats *$l$ and $r$ and $s[l] \mapsto s[r]$, then*

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

*Proof. (sketch)* The proof follows by structural induction on $s$. The only non-trivial case is when $s$ is headed by a wave-front, e.g., $s = \boxed{f(\underline{s_1}, \ldots, \underline{s_j}, s_{j+1}, \ldots, s_n)}^{\uparrow}$, and $l$ is *strict* subterm of one of the $s_i$ (the case for inward directed wave-fronts is analogous). There are two cases depending on if $i \leq j$. In the first case, $s_i$ is a *wat*; by the induction hypothesis, $s_i[r]$ is a *wat*. Thus, $s[r]$ is a *wat*. Also, by the induction hypothesis $skel(s_i[r]) \subseteq skel(s_i[l])$. As no other subterm is changed, the union of their skeletons is unchanged. Hence $skel(s[r]) \subseteq skel(s[l])$. Finally, by the induction hypothesis, $erase(s_i[l]) \rightarrow erase(s_i[r])$. Again, as all the other subterms are unchanged, their erasures stay the same. Thus, $erase(s[l]) \rightarrow erase(s[r])$. In the second case, $s_i$ is an unannotated term within the wave-front. 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*. From the definition of the skeleton it follows that term replacement in wave-fronts has no effect on the skeleton, so $skel(s[l]) = skel(s) = skel(s[r])$. Finally $s_i$ is unannotated and $l \rightarrow r$ is a proper equation, $erase(s_i[l]) \rightarrow erase(s_i[r])$ and thus $erase(s[l]) \rightarrow erase(s[r])$. ∎

By Theorem 1 it follows by induction on the number of rewrite steps that the reflexive transitive closure of $\mapsto$ on ground *wats* also preserves well-formedness, skeletons and correctness with respect to the theory.

As a simple example, let $\boxed{h(\underline{a})}^{\uparrow} \rightarrow a$ be a proper rewrite rule. We can apply this rule only to the first subterm of the *wat*

$$\boxed{f(\underline{\boxed{h(\underline{a})}^{\uparrow}}, h(a))}^{\uparrow} \tag{10}$$

and this results in $\boxed{f(\underline{a}, h(a))}^{\uparrow}$. Alternatively, we could apply the proper rewrite rule $a \rightarrow \boxed{h(\underline{a})}^{\uparrow}$ to both occurrences of $a$ in (10) resulting in

$$\boxed{f(\underline{\boxed{h(\underline{\boxed{h(\underline{a})}^{\uparrow}})}^{\uparrow}}, h(h(a)))}^{\uparrow} . \tag{11}$$

Note that annotation was erased when substituting $\boxed{h(\underline{a})}^{\uparrow}$ for the second occurrence of $a$. We can apply this rewrite rule again to both occurrences of $a$ in (11). Whilst rippling with proper rewrite rules is structure preserving (the skeleton of the rewritten term is always the same as the skeleton of the original term), this example shows that it is not necessarily terminating; for termination we need further restrictions which will be introduced later.

### 4.2. Annotated Matching

When rewrite rules contain (non-term) variables, they must be applied using matching. Since substitution depends on subterm replacement, our new definition of subterm replacement gives rise to a new kind of substitution; during substitution, terms replacing variables in wave-fronts are erased of annotation. This in turn gives rise to a new kind of matching. We represent a *well-annotated substitution* (or *was*) by a set of pairs, $X/t$ in which $X$ is a variable and $t$ is a *wat*, and $X$ occurs only once in the set. The *domain* of a substitution $\sigma$, written $Dom(\sigma)$ is the set $\{X \mid X/t \in \sigma\}$. Application of a well-annotated substitution $\sigma$ to a well-annotated term $t$, written $\sigma(t)$, is as normal except term replacement is performed with our new subterm replacement function. Finally, if $\sigma$ is a *was*, we define the $erase(\sigma)$, and $skel(\sigma)$ as the result of applying these functions to each element in the range of the substitution, i.e.,

$$erase(\sigma) = \{X/t' \mid X/t \in \sigma \wedge t' = erase(t)\} .$$

The following can be easily verified by induction over the structure of *wats*.

LEMMA 1. *For $\sigma$ a well-annotated substitution we have*

$$\sigma(\boxed{f(\underline{t_1}, \ldots, t_n)}^{\uparrow}) = \boxed{f(\underline{\sigma(t_1)}, \ldots, erase(\sigma)(t_n))}^{\uparrow} = \boxed{f(\underline{\sigma(t_1)}, \ldots, erase(\sigma(t_n)))}^{\uparrow}$$

*and*

$$\sigma(\boxed{f(\underline{t_1}, \ldots, t_n)}^{\downarrow}) = \boxed{f(\underline{\sigma(t_1)}, \ldots, erase(\sigma)(t_n))}^{\downarrow} = \boxed{f(\underline{\sigma(t_1)}, \ldots, erase(\sigma(t_n)))}^{\downarrow}.$$

Annotated substitution preserves well-formedness, skeletons and erasure in the following sense:

THEOREM 2. *if $t$ is a* wat *and $\sigma$ is a well-annotated substitution then*

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

*Proof. (sketch)* The proof follows by structural induction on $t$. As before, the only interesting case is when $t = \boxed{f(\underline{t_1}, \ldots, \underline{t_j}, t_{j+1}, \ldots, t_n)}^{\uparrow}$ (the inwards case is analogous). (1) follows as

$$\sigma(t) = \boxed{f(\underline{\sigma(t_1)}, \ldots, \underline{\sigma(t_j)}, erase(\sigma(t_{j+1})), \ldots, erase(\sigma(t_n)))}^{\uparrow}$$

and subterms in wave-fronts, i.e., $\sigma(t_i)$ for $i \in \{1, \ldots, j\}$ are *wats* by the induction hypothesis and the remaining subterms are unannotated and therefore also *wats*. (2) follows as

$$\begin{aligned}
skel(\sigma(t)) &= skel(\boxed{f(\underline{\sigma(t_1)}, \ldots, \underline{\sigma(t_j)}, erase(\sigma(t_{j+1})), \ldots, erase(\sigma(t_n)))}^{\uparrow}) \\
&= \cup_{i=1}^{j} \{s \mid s \in skel(\sigma(t_i))\} \\
&= \cup_{i=1}^{j} \{s \mid s \in skel(\sigma)(skel(t_1))\} \\
&= (skel(\sigma))skel(\boxed{f(\underline{t_1}, \ldots, \underline{t_j}, t_{j+1}, \ldots, t_n)}^{\uparrow})
\end{aligned}$$

Finally (3) follows as

$$\begin{aligned}
erase(\sigma(t)) &= erase(\boxed{f(\underline{\sigma(t_1)}, \ldots, erase(\sigma(t_n)))}^{\uparrow}) \\
&= f(erase(\sigma(t_1)), \ldots, erase(erase(\sigma(t_n)))) \\
&= f(erase(\sigma(t_1)), \ldots, erase(\sigma(t_n))) \\
&= f((erase(\sigma))(erase(t_1)), \ldots, (erase(\sigma))(erase(t_n))) \\
&= (erase(\sigma))(f(erase(t_1), \ldots, erase(t_n))) \\
&= (erase(\sigma))(erase(t))
\end{aligned}$$

■

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

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

DELETE :
$S \cup \{t = t : Pos\} \;\Rightarrow\; S$

DECOMPOSE :
$S \cup \{f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) : Pos\} \;\Rightarrow\; S \cup \{s_i = t_i : Pos \mid 1 \leq i \leq n\}$
$S \cup \{\boxed{f(\underline{s_1}, \ldots, \underline{s_j}, s_{j+1}, \ldots, s_n)}^{\uparrow} = \boxed{f(\underline{t_1}, \ldots, \underline{t_j}, t_{j+1}, \ldots, t_n)}^{\uparrow} : sk\} \;\Rightarrow$
$S \cup \{s_1 = t_1 : sk, \ldots, s_j = t_j : sk, s_{j+1} = t_{j+1} : wf, \ldots, s_n = t_n : wf\}$
$S \cup \{\boxed{f(\underline{s_1}, \ldots, \underline{s_j}, s_{j+1}, \ldots, s_n)}^{\downarrow} = \boxed{f(\underline{t_1}, \ldots, \underline{t_j}, t_{j+1}, \ldots, t_n)}^{\downarrow} : sk\} \;\Rightarrow$
$S \cup \{s_1 = t_1 : sk, \ldots, s_j = t_j : sk, s_{j+1} = t_{j+1} : wf, \ldots, s_n = t_n : wf\}$

Fig. 1. Transformation rules for $amatch(s, t)$.

Observe that even restricting $\sigma$ to variables in $s$, annotated matching is not unitary. For example, $\{X/s(0)\}$ and $\{X/\boxed{s(\underline{0})}^{\uparrow}\}$ are both annotated matches of $\boxed{X \times \underline{0}}^{\uparrow}$ with $\boxed{s(0) \times \underline{0}}^{\uparrow}$. It can be seen, however, that matches differ only in the amount of annotation which appear on substitutions for variables that occur in wave-fronts but not in skeletons. Based on this observation, we can define a notion of minimality of annotation so that annotated matching is unique. If $\sigma$ and $\tau$ are well annotated substitutions for $s$ and $t$ then we write $\sigma \prec \tau$ iff there exists $X/t_1 \in \sigma$ and $X/t_2 \in \tau$ with $t_1 = erase(t_2)$ and all other pairs in $\sigma$ and $\tau$ are identical. We write $\prec^+$ for the transitive (but not reflexive) closure of $\prec$.

DEFINITION 6. if $s$ and $t$ are wats, then $\sigma$ is a *minimal match* of $s$ with $t$ iff $\sigma$ is an annotated match of $s$ with $t$ and there does not exist any annotated match $\tau$ with $\tau \prec^+ \sigma$.

It follows from this definition that if we have a minimal match, we cannot remove any annotation and have the result remain a match. We now give an algorithm for computing minimal matches. We will use this afterwards to show that minimal matches are unique.

Our procedure, $amatch(s, t)$, is based on the transformation rules given in Figure 1. Because term replacement, and hence substitution is dependent on context (i.e. whether or not the term to be replaced is in a wave-front), our rules manipulate equations labeled with context information (*wf* for "in the wave-front" and *sk* for "in the skeleton"). As notational shorthand, *Pos* is a meta-variable that matches either *wf* or *sk*.

Starting with the set containing the match problem $\{s = t : sk\}$, we apply these transformation rules exhaustively. An equation set is *normalized* if no more transformation rules can be applied. A normalized equation set, $S$ is *compatible* iff 1)

every equation is a variable assignment $X = s : Pos$ where $s$ is a $wat$, 2) for each variable $X$ there exists at most one equation of the form $X = s : sk$, and 3) if $X = s : sk \in S$ and $X = t : wf \in S$ then $erase(s) = t$. If a normalized equation set is not compatible, matching has failed; otherwise matching has succeeded and we return the answer substitution

$$\{X/s \mid X = s : sk \in S \text{ or } (X = s : wf \in S \text{ and } X = t : sk \notin S)\}.$$

As an example, if we match $\boxed{\underline{X} + X}^{\uparrow}$ with $\boxed{\underline{\underline{s(\underline{a})}^{\uparrow}} + s(a))}^{\uparrow}$ then our initial matching problem is

$$\{\boxed{\underline{X} + X)}^{\uparrow} = \boxed{\underline{\underline{s(\underline{a})}^{\uparrow}} + s(a))}^{\uparrow} : sk\}$$

and applying DECOMPOSE yields $\{X = \boxed{s(\underline{a})}^{\uparrow} : sk, X = s(a) : wf\}$. This normalized equation set is compatible and yields the answer $\{X/\boxed{s(\underline{a})}^{\uparrow}\}$. Note that regular matching would fail on this example.

It is easy to see that the application of these rules in any order terminates in time linear in the size of the smaller of $s$ and $t$. Moreover, although there are choice points concerning which rule is applied, these choices are $AND$ choices: all equations must be reduced and each reduction is independent. It follows that when there is an answer substitution, it is unique. Below, when we show correctness, we demonstrate that the algorithm returns all minimal answers, so it follows that there is a unique such answer.

Note that the DELETE and the first DECOMPOSE rule implement regular matching. For unannotated terms, the compatibility check reduces to the requirement that the normalized equation set only contains variable assignments, and each variable has a single substitution. Annotated matching therefore subsumes regular matching. As with regular matching, we can also add two failure rules for greater efficiency: CONFLICT which causes annotated matching to fail when the outermost function symbols disagree; and INCOMPATIBLE which causes annotated matching to fail immediately if the set of equations of the form $X = s : Pos$ is not compatible. These additional failure rules are not, however, needed for the completeness or correctness of annotated matching.

THEOREM 3. *if $s$ and $t$ are* wats *then,*

1. *$amatch(s, t) = \sigma$ iff $\sigma$ is the minimal match of $s$ and $t$;*
2. *$amatch(s, t)$ fails iff $s$ and $t$ do not have an annotated match.*

*Proof. (sketch)* We prove a stronger result: a set of labeled equations $S$ can be transformed to a compatible set of equations from which we extract $\sigma$ iff for all $s = t : sk \in S$, $\sigma(s) = t$, and for all $s = t : wf \in S$, $(erase(\sigma))(s) = t$. The minimality of the answer substitution extracted holds by construction.

($\Rightarrow$) We use induction on the length of the transformation. If $S$ is already a normalized compatible set of equations, then the result follows directly from the way the answer substitution is computed. Alternatively, we must apply a transformation rule. The interesting case is when DECOMPOSE is applied, say to $\boxed{f(\underline{s_1},\ldots,s_n)}^{\uparrow} = \boxed{f(\underline{t_1},\ldots,t_n)}^{\uparrow} : sk$, giving the set of equations $S'$. By the induction hypothesis, for all $s = t : sk \in S'$, $\sigma(s) = \sigma(t)$, and for all $s = t : wf \in S'$, $(erase(\sigma))(s) = (erase(\sigma))(t)$. By the definition of annotated substitution, we have $\sigma(\boxed{f(\underline{s_1},\ldots,s_n)}^{\uparrow}) = \boxed{f(\underline{t_1},\ldots,t_n)}^{\uparrow}$.

($\Leftarrow$) We use induction on the height of the largest LHS of an equation. If this is atomic then, possibly after applications of DELETE, the equation set will be compatible. If it is not atomic, then the equation with the largest LHS will be of the form $f(s_1,\ldots,s_n) = t : Pos$, $\boxed{f(\underline{s_1},\ldots,s_n)}^{\uparrow} = t : sk$, or $\boxed{f(\underline{s_1},\ldots,s_n)}^{\downarrow} = t : sk$. In all three cases, we can apply DECOMPOSE and appeal to the induction hypothesis.

Note that the normalized equation set is not compatible iff either two occurrences of a variable in the skeleton need a different substitution, or a variable in the wave-front needs a substitution which is not the erasure of the substitution needed by an occurrence in the skeleton, or there is a conflict in function symbols or annotation preventing application of DECOMPOSE. But this occurs iff $s$ and $t$ do have an annotated match. ∎

### 4.3. First-Order Annotated Rewriting

We now consider rippling with proper rewrite rules that may contain variables. First, we define non-ground rippling. Let $s[t]$ be a *wat* with a distinguished subterm $t$ and $l \to r$ be a proper rewrite rule. Further, let $\sigma = amatch(l,t)$. Then $s[t]$ rewrites to $s[\sigma(r)]$, which we write as $s[\sigma(l)] \mapsto s[\sigma(r)]$. Correctness parallels the ground case. The proof relies on the fact that $\sigma$ is a well-annotated substitution and we can thus reduce the problem to the correctness of ground rippling.

THEOREM 4. *if $s$ is a* wat*, $l \to r$ a proper rewrite rule, and $s[\sigma(l)] \mapsto s[\sigma(r)]$, then*

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

*Proof.* Annotated matching guarantees that $\sigma$ is a *was*. By Theorem 2, $\sigma(l)$ and $\sigma(r)$ are *wats*, $skel(\sigma(r)) \subseteq skel(\sigma(l))$ and $erase(\sigma(l)) \to erase(\sigma(r))$. Now, since $\sigma(l)$ is syntactically identical to a subterm of $s$, it is ground (no non-term variables occur in $s$). Furthermore $\sigma(r)$ is also ground because $l \to r$ is a rewrite rule and thus $Vars(r) \subseteq Vars(l)$. Hence the rewriting of $s[\sigma(l)] \mapsto s[\sigma(r)]$ is equivalent

to rewriting $s[\sigma(l)]$ with the ground proper rewrite rule $\sigma(l) \to \sigma(r)$. Thus, by Theorem 1, the three properties hold. ∎

Let $\mapsto^*$ be the reflexive transitive closure of $\mapsto$. By induction on the number of rewrite steps, it follows from Theorem 4 that rippling is correct and preserves skeletons. That is, if we erase annotations, we can perform the same (object-level) rewriting; annotations merely guide rewriting in a skeleton preserving way.

THEOREM 5. *if $s$ is a* wat *and $s \mapsto^* t$ then*

1. *$t$ is a* wat,
2. *$skel(t) \subseteq skel(s)$,*
3. *$erase(s) \to^* erase(t)$.*

### 4.4. Termination

We now have a calculus which satisfies the first three properties required of rippling. Now we address the fourth: termination. As is conventional, we guarantee termination by placing an additional restriction on (proper) rewrite rules: they must reduce the measure of the rewritten term in a well-founded ordering.

We first define a weak kind of monotonicity that ensures monotonicity with respect to replacement of *wats* by *wats*. This is adequate to show termination when our rewrite rules are between *wats*.

DEFINITION 7. An order $>$ is *monotonic with respect to* wats iff for all *wats* $s$ and for all ground *wats* $l$ and $r$, if $l > r$, then $s[l] > s[r]$.

We now consider termination restricted to the ground case.

LEMMA 2. *Let $R$ be a set of (not necessarily proper) rewrite rules between ground* wats. *If $>$ is well-founded and monotonic with respect to* wats, *and for all $l \to r \in R$ we have $l > r$ then rippling with $R$ is terminating.*

*Proof.* By monotonicity, an infinite rewrite sequence $t_1 \mapsto t_2 \mapsto \ldots$ gives an infinite sequence of *wats* $t_1 > t_2 > \ldots$, contradicting the well-foundedness of $>$. ∎

For non-ground rippling we consider a restricted form of stability.

DEFINITION 8. An order $>$ is *stable* with respect to *wats* iff for all well-annotated substitutions $\sigma$ and *wats* $s$ and $t$ where $s > t$, we have that $\sigma(s) > \sigma(t)$. An order on annotated terms that is well-founded, monotonic, and stable with respect to *wats* is an *annotation reduction order*.

Note that annotation reduction orders are strictly weaker than normal reduction orders which are monotonic and stable over all terms in the signature, as opposed to just well-annotated ones. We will call our orders simply *reduction orders* when no confusion can arise.

Based on these definitions, we can at last formally define a *wave-rule*.

DEFINITION 9. Let $>$ be an annotation reduction order. Then a proper rewrite rule $l \to r$ is a *wave-rule with respect to* $>$ iff $l > r$.

THEOREM 6. *for an annotation reduction order $>$ and $R$ a set of wave-rules with respect to $>$, rippling using wave-rules in $R$ is terminating.*

*Proof.* We again reduce the problem to the ground case. If $s \mapsto t$ using $l \to r$ then this is equivalent to rippling with a rewrite rule $\sigma(l) \to \sigma(r)$ between ground *wats*. Since $>$ is stable with respect to *wats*, $\sigma(l) > \sigma(r)$. By the termination of ground rippling, we have termination in the general case. ∎

Note that our proof is similar to the one given in [11] (Corollary to Theorem 5) as we need not show that the ordering is a reduction ordering, but rather only monotonic and stable with respect to possible instances of the rewrite rules.

## 5. Annotation Orders

To prove the termination of rippling using Theorem 6, we need to define a suitable order on annotated terms. We begin with simply annotated terms, those whose wave-fronts have a single wave-hole. We then generalize to orders for terms with multi-hole annotation. The orders we define are similar, though simpler, to that given by Bundy *et al.* in [6]. We can order all the wave-rules given in [6] and admit wave-rules not possible in their setting (see §8).

### 5.1. Single Wave-holes

We consider annotated terms as decorated trees where the tree is the skeleton and the wave-fronts are boxes decorating the nodes. See, for example, the first tree in Fig. 2 which represents the term $\boxed{s(\underline{U})}^{\uparrow} \geq \boxed{s(\underline{V})}^{\uparrow}$. Our orders are based on assigning measures to annotation in these trees. We define orders by progressively simplifying these annotated trees to capture the notion of progress during rippling that we wish to measure.

To begin with, since rippling is skeleton preserving, we need not account for the contents of the skeleton in our orderings. That is, we can abstract away function symbols in the skeleton, for example, mapping each to a variadic function constant "*". This gives, for example, the second tree in Fig. 2.

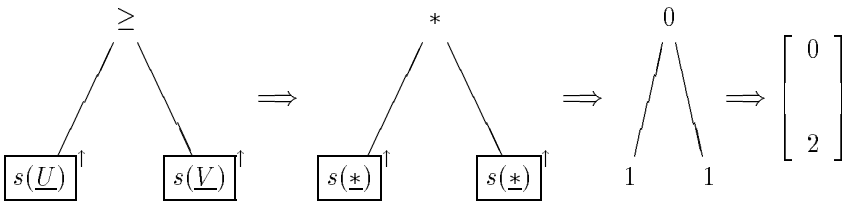


Fig. 2. Defining a measure on annotated terms.

A further abstraction is to ignore the names of function symbols within wave-fronts and assign a numeric weight to wave-fronts. For example, we may tally up the values associated with each function symbol as in a Knuth-Bendix ordering. Two of the simplest kinds of weights that we may assign to wave-fronts measure their *width* and their *size*. Width is the number of nested function symbols between the root of the wave-front and the wave-hole. Size is the number of function symbols and constants in a wave-front. In what follows we will restrict our attention to the width measure. This gives, for example, the third tree in Fig. 2. Of course, there are problem domains where we want our measure to reflect more of the structure of wave-fronts. §7.1 contains an example of this where the actual contents of the wave-front are compared using a conventional term ordering.

Finally, a very simple notion of progress during rippling is that wave-fronts move up or down through the skeleton tree. Under this view, the tree structure may be ignored: it is not important which branch a wave-front is on, only its depth in the skeleton tree. Hence, we can apply an abstraction that maps the tree onto a list, level by level. For instance, we can use the sum of the weights at a given depth. Applying this abstraction gives the final list in Fig. 2. Note that depths are relative to the skeleton as opposed to depth in the erased term; measuring depth relative to a fixed skeleton is one of the key ideas in the measures proposed here.

To formalize the above ideas, we introduce the following definitions. As is standard, a *position* is simply a path address represented by a string and is defined as follows: the set of positions in the term $t$ is $Pos(t)$ where,

$$Pos(f(s_1, \ldots, s_n)) \;=\; \{\Lambda\} \cup \{i.p \mid 1 \leq i \leq n \;\wedge\; p \in Pos(s_i)\}$$

$\Lambda$ represents the empty string and "." is the string concatenation operator. The subterm of a term $t$ at position $p$ is $t/p$ where:

$$t/\Lambda \;=\; t$$
$$f(s_1, \ldots, s_n)/i.p \;=\; s_i/p$$

If $s$ is a subterm of $t$ at position $p$, its *depth* is the length of $p$. The *height* of $t$, written $|t|$, is the maximal depth of any subterm in $t$.

Because we are interested in measures based on weight relative to the skeleton, during the remainder of this paper, positions, depth, and height, will *always* be

relative to the skeleton of simply annotated terms. That is, we picture such terms as in the first tree in Figure 2. The positions in the term tree are only those in the skeleton; annotation and function symbols in wave-fronts are treated as markings of function symbols in the skeleton. For example, the term in Figure 2 is $\boxed{s(\underline{U})}^{\uparrow} \geq \boxed{s(\underline{V})}^{\uparrow}$ which has skeleton $U \geq V$. The height of this term is 1 since the deepest subterms, $U$ and $V$, have positions 1 and 2 respectively. Another example is $\boxed{f(s(\underline{f(a,s(b))}),c)}^{\uparrow}$ with the skeleton $f(a,s(b))$. The deepest subterm is $b$ at position 2.1 and hence the height of the annotated term is 2.

For an annotated term $t$, the *out-weight of a position* $p$ is the sum of the weights of the (possibly nested) outwards oriented wave-fronts at $p$. The *in-weight* is defined analogously except for inward directed wave-fronts. We now define a measure on terms corresponding to the final list in Fig. 2 based on weights of annotation relative to their depths.

DEFINITION 10. The *out-measure*, $\mathcal{MO}(t)$ of an annotated term $t$ is a list of length $|t|+1$ whose $i$-th element is the sum of out-weights for all term positions in $t$ at depth $i$. The *in-measure*, $\mathcal{MI}(t)$ is a list whose $i$-th element is the sum of in-weights for all term positions in $t$ at depth $i$. The *measure* of an annotated term, $\mathcal{M}(t)$ is the pair of out and in-measures, $\langle \mathcal{MO}(t), \mathcal{MI}(t) \rangle$.

Consider, for example, the following palindrome function ("::" is infix cons)

$$palin(\boxed{H :: \underline{T}}^{\uparrow}, Acc) \rightarrow \boxed{H :: \underline{palin(T, \boxed{H :: \underline{Acc)}}^{\downarrow})}}^{\uparrow} \tag{12}$$

The skeleton of both sides is $palin(T, Acc)$. The out-measure of the left-hand side is [0,1] and that of the right-hand side is [1,0]. The in-measures are [0,0] and [0,1].

We now define a well-founded ordering on these measures which reflects the progress that we want rippling to make. Consider, a simple wave-rule like (2),

$$\boxed{s(\underline{U})}^{\uparrow} \times V \rightarrow \boxed{(\underline{U \times V}) + V}^{\uparrow}.$$

The LHS out-measure is $[0,1]$, and the RHS is $[1,0]$. Rippling with this rule makes progress because it moves one wave-front upwards towards the root of the term. In general, rippling progresses if one out-oriented wave-front moves up or disappears, while nothing deeper moves downwards. If the out-measure of a term before rippling is $[l_0, \ldots, l_k]$ and after $[r_0, \ldots, r_k]$ then there must be some depth $d$ where $l_d > r_d$ and for all $i > d$ we have $l_i = r_i$. This is simply the lexicographic order on the reverse of the two lists where components are compared using $>$ on the natural numbers. Progress for in-oriented wave-fronts is similar and reflects that these wave-fronts should move towards leaves; that is just the lexicographic order on the in-measures. Of course, both outward and inward oriented wave-fronts may

occur in the same rule, e.g., (12). Similar to [6], we define a composite ordering on the out and in-measures. We order the out-measure before the in-measure since this enables us to ripple wave-fronts out and either to reach the root of the term, or at some point to turn wave-fronts down and to ripple in towards the leaves.

DEFINITION 11. $t \succ s$ iff $\mathcal{M}(t) > \mathcal{M}(s)$ and $skel(s) = skel(t)$. Here $>$ represents the lexicographic order on pairs with $>_{revlex}$ (the reversed lexicographic order on lists) used to compare the out-measure and $>_{lex}$ (the lexicographic order on lists) to compare the in-measure.

This definition is sensible as the restriction that $skel(s) = skel(t)$ means that the measure lists are the same length and may be compared. Although a skeleton independent measure would be desirable there is a deeper reason for this restriction: our order would not be stable without it. As a simple example, consider the terms $s = \boxed{X + \underline{s(s(Y))}}^{\uparrow}$ and $t = \boxed{s(\underline{X}) + Y)}^{\uparrow}$. If we ignore the skeleton restriction and just compare annotation measures then $s \succ t$. However, under the substitution $\sigma = \{X/\boxed{s(s(\underline{a}))}^{\uparrow}\}$ we have $\sigma(t) \succ \sigma(s)$. We will shortly show the stability of our more restricted ordering.

Given the well-foundedness of $>$ on the natural numbers and that lexicographic combinations of well-founded orders are well-founded we can conclude:

THEOREM 7. *The composite ordering is well-founded.*

## 5.2. Multiple Wave-holes

We now generalize our order to multi-hole annotation; that is, multiple wave-holes in a single wave-front. Wave-rules involving such terms are called *multi-wave-rules* in [6]. We have already seen an example of this in (6). The binomial equation is another example.

$$binom(\boxed{s(\underline{X})}^{\uparrow}, \boxed{s(\underline{Y})}^{\uparrow}) = \boxed{\underline{binom(X, \boxed{s(\underline{Y})}^{\uparrow})} + \underline{binom(X, Y)}}^{\uparrow} \qquad (13)$$

Both sides have the same skeleton, namely $\{binom(X, Y)\}$. In general, however, the skeletons of the right-hand side of a multi-wave-rule need only be a subset of the skeletons of the left-hand side.

We define orders for terms with multi-hole annotation in a uniform way from the previous single hole ordering by reducing terms with multi-hole annotation to sets of simply annotated terms and extending the single hole ordering to these sets. This reduction is accomplished by considering ways that multi-hole annotation can be *weakened* to simple annotation by "absorbing" wave-holes. Weakening a multi-wave term like (13) erases some of the wave-holes (underlining) though always

leaving at least one wave-hole. By erasing a wave-hole $\underline{t_i}$ we mean removing the underline annotation and erasing any further annotation in $t_i$. A wave-front is *maximally weak* when it has exactly one wave-hole. A term is *maximally weak* when all its wave-fronts are maximally weak. Maximally weak terms are simply annotated and we can apply the previously defined single hole measure to them.

Returning to the binomial example, (13) has precisely two weakenings.

$$binom(\boxed{s(\underline{X})}^{\uparrow}, \boxed{s(\underline{Y})}^{\uparrow}) = \boxed{\underline{binom(X, \boxed{s(\underline{Y})}^{\uparrow})} + binom(X, Y)}^{\uparrow} \quad (14)$$

$$binom(\boxed{s(\underline{X})}^{\uparrow}, \boxed{s(\underline{Y})}^{\uparrow}) = \boxed{binom(X, s(Y)) + \underline{binom(X, Y)}}^{\uparrow} \quad (15)$$

Both are maximally weak as each wave-front has a single hole. As another example, the left-hand side of (6) has four maximal weakenings (and four non-maximal weakenings) whilst the right-hand side has two weakenings, both maximal.

Let *weakenings*$(s)$ be the set of maximal weakenings of $s$. It is easily computed by constructing the closure of all weakenings of $s$ and returning the set of simply annotated results. As elements of these sets are simply annotated, we can apply the single hole measure to them. A natural order to define on such sets is therefore the multi-set extension of the order used to compare simply annotated terms. A multi-set extension of an ordering is defined as follows [11].

DEFINITION 12. A *multi-set ordering* $>>$ is induced from a given ordering $>$ whereby $M >> N$ iff $N$ can be obtained from $M$ by replacing one or more elements in $M$ by any finite number of elements each of which is smaller (under $>$) than one of the replaced elements.

We extend the single hole ordering to multi-hole annotated terms as follows.

DEFINITION 13. $l \succ^{\star} r$ iff *weakenings*$(l) \succ\succ$ *weakenings*$(r)$ where $\succ\succ$ is the multi-set extension of the single hole order $\succ$.

This order is well-defined as maximal weakenings are simply annotated and can be compared using the single hole order. Note that if $l$ and $r$ are simply annotated then their weakenings are $\{l\}$ and $\{r\}$, and $l \succ^{\star} r$ and $l \succ r$ are equivalent. We will drop the superscript on $\succ^{\star}$ when context makes our intention clear.

As an example consider (13). The LHS weakenings are

$$\{binom(\boxed{s(\underline{X})}^{\uparrow}, \boxed{s(\underline{Y})}^{\uparrow})\}.$$

The RHS weakenings are

$$\{\boxed{\underline{binom(X, \boxed{s(\underline{Y})}^{\uparrow})} + binom(X, Y)}^{\uparrow}, \boxed{binom(X + s(Y)) + \underline{binom(X, Y)}}^{\uparrow}\}.$$

The only member of the first set is $\succ$-greater than both members of the second set. This wave-rule is thus measure decreasing.

### 5.3. Termination under $\succ^\star$

Since $\succ^\star$ is defined via a multi-set extension of a well-founded order it too is well-founded.

LEMMA 3. *$\succ^\star$ is well-founded.*

We now show that $\succ^\star$ is monotonic and stable. To simplify proofs, we ignore complications caused by inwards oriented wave-fronts. Reincorporating these is conceptually simple but notationally involved since measures expand to pairs.

As measures are lists, term replacement corresponds to operations on lists. Hence we begin with relevant terminology. Let $l$ and $r$ be list of integers and $l + r$ and $l - r$ be componentwise addition and subtraction. When one list is shorter than the other, we "pad" it out by appending additional 0s to the end so that its length is the same as the longer. For $n$ a natural number, let $l{\uparrow^n}$ be the result of "right shifting" $l$ by $n$ positions by appending $l$ to the end of the list containing $n$ zeros. If the length of $l$ is $n+1$, then for any $d \in \{0, \ldots, n\}$ we define the *splice* of $r$ into $l$ at depth $d$, which we write as $l \ +_d \ r$, to be $l + (r{\uparrow^d})$. Splicing can result in a longer list; for example, if $l = [l_0, l_1, l_2, l_3]$ and $r = [r_0, r_1, r_2]$, then

$$l \ +_2 \ r = l + (r{\uparrow^2}) = [l_0, l_1, l_2, l_3] + [0, 0, r_0, r_1, r_2] = [l_0, l_1, l_2 + r_0, l_3 + r_1, r_2].$$

We will use some simple properties about splice and list arithmetic below.

LEMMA 4. *Let $l$ and $l'$ be lists of length $i+1$ and $l >_{revlex} l'$. Let $r, r_1, \ldots, r_k$ be lists of length $j+1$ then*

1. $\forall d \in \{0, \ldots, j\}. r >_{revlex} r \ +_d \ l' - l$
2. $\forall d \in \{0, \ldots, i\}. l \ +_d \ r >_{revlex} l' \ +_d \ r$
3. $\forall d_1, \ldots, d_m \in \{0, \ldots, i\}. (\ldots((l \ +_{d_1} \ r_1) \ +_{d_2} \ r_2) \ldots +_{d_m} \ r_l) >_{revlex} (\ldots((l' +_{d_1} \ r_1) \ +_{d_2} \ r_2) \ldots +_{d_m} \ r_l)$

The first lemma says we can splice in the difference between $l'$ and $l$ where $l >_{revlex} l'$ and the result will be smaller. The second says we can splice a list $r$ into $l$ and $l'$ and preserve the ordering of $l$ and $l'$. The third lemma is essentially an iterated version of the second for performing multiple splices with different lists at multiple positions. We use these results to prove theorems about stability and monotonicity as such theorems can be seen as statements about splicing measures.

LEMMA 5. *$\succ^\star$ is monotonic with respect to* wats.

*Proof. (sketch)* Let $s[l]$ be a term with a distinguished subterm $l$. Note that if $l \succ r$ then $l$ must be annotated. We argue by cases. Suppose first that $s$, $l$, and $r$ are simply annotated. Let $m_l = \mathcal{MO}(l)$ be the measure of $l$; similarly let $m_r = \mathcal{MO}(r)$ and $m_s = \mathcal{MO}(s)$. Let $d$ be the depth of $l$ in the skeleton of s. The measure of $s[r]$ is the measure of $s$ altered by splicing at depth $d$ the difference between the measures of $r$ and $l$, i.e., $m_s +_d (m_r - m_l)$. Since $l \succ r$ we can conclude, using the first part of Lemma 4, that $s[l] \succ^\star s[r]$.

Now suppose $l$ and $r$ contain multi-hole annotation and the only multi-hole annotation in $s[l]$ occurs in $l$ itself. Let the maximal weakenings of $l$ and $r$ be the sets $L = \{l_1, \ldots, l_j\}$ and $R = \{r_1, \ldots, r_k\}$ respectively. The maximal weakenings of $s[l]$ and $s[r]$ then are the sets $S_l = \{s[l_1], \ldots, s[l_j]\}$ and $S_r = \{s[r_1], \ldots, s[r_k]\}$. Now under the definition of $\succ^\star$ and multi-sets, $l \succ^\star r$ if we can replace some collection of the $l_i \in L$ by smaller elements (under $\succ$) resulting in the set $R$. But we can do the identical replacements in the context $s[.]$ hence transforming the set $S_l$ to $S_r$. Consider such a replacement, say replacing $l_1 \in L$ by $r_1, \ldots, r_p$; now $l_1 \succ r_i$ and it follows (by the previously considered case) that $s[l_1] \succ s[r_i]$ for each $i \in \{1, \ldots, p\}$. Hence the transformation of $S_l$ to $S_r$ shows that $s[l] \succ^\star s[r]$.

The final case to consider is when $s$ itself has multiple skeletons, independent of the number of skeletons of $l$. We argue as above except that rather than just comparing sets composed from $s[l_i]$ and $s[r_i]$ we have to consider weakings of $s$ as well. But any steps in weakening $s$ (not in the subterm $l$) can be made identically in both $s[l_i]$ and $s[r_i]$ and $s[l] \succ^\star s[r]$ follows. ∎

LEMMA 6. $\succ^\star$ *is stable with respect to wats.*

*Proof. (sketch)* Let $s$ and $t$ be *wats* with $s \succ^\star t$. To show that $\sigma(s) \succ^\star \sigma(t)$ it suffices to consider a substitution $\sigma$ that replaces a single variable $x$ with a wat $r$ since substitution for multiple variables can be achieved by iterating substitutions for single variables. We consider two cases: first, when $s$, $t$ are simply annotated, and second, when they may contain multi-hole wave-fronts.

Case 1: $s$ and $t$ are simply annotated. As $s \succ t$, both is terms have the same single skeleton. Note that substitutions for occurrences of $x$ in wave-fronts have no effect on our width measure (although they can change the size of a wave-front). Assume $x$ occurs $p$ times in each skeleton. If $\mathit{weakenings}(r) = \{r_1, \ldots, r_m\}$ then

$$S = \mathit{weakenings}(\sigma(s)) = \{s_1, \ldots, s_n\}$$

and

$$T = \mathit{weakenings}(\sigma(t)) = \{t_1, \ldots, t_n\}$$

where $n = p * m$. Each of these weakenings can be constructed by replacing the variables $x$ in $s$ and $t$ with maximal weakenings of $r$; each $s_i$ thus has a "partner" $t_i$ in which the occurrences of $x$ are replaced by the same weakening of $r$. Now to show that $S$ is greater than $T$ under the multi-set ordering we must give a transformation of $S$ to $T$ where each term is replaced by a finite number of smaller

(under $\succ$) terms. Our transformation is simply to replace $s_i$ by its partner $t_i$. If we order (arbitrarily) the occurrences of $x$ in the skeleton of $s$ (and therefore also $t$), $x_1, \ldots, x_p$, then if $s_i$ and $t_i$ were formed by replacing $x_j$, occurring at depth $d_j$ with a weakening of $t$ that has a measure $r_j$, then the measures of the two terms $s_i$ and $t_i$ are

$$(\ldots((s +_{d_1} r_1) +_{d_2} r_2) \ldots +_{d_p} r_p)$$

and

$$(\ldots((t +_{d_1} r_1) +_{d_2} r_2) \ldots +_{d_p} r_p)$$

respectively. But now, using part 3 of Lemma 4 we have that the former is greater under $>_{revlex}$ than the latter, hence $\sigma(l) \succ \sigma(r)$.

Case 2: all terms may contain multi-hole annotation. Let $S = \{s_1, \ldots, s_j\}$ and $T = \{t_1, \ldots, t_k\}$ be the maximal weakenings of $s$ and $t$. As $s \succ^\star t$, there is a transformation (respecting $\succ$) of $S$ to $T$. We must construct a transformation from the maximal weakenings of $\sigma(s)$ to the maximal weakenings of $\sigma(t)$. We proceed as follows. Consider a replacement of, say, $s_1$ in $S$ with some $t_1, \ldots, t_p$ that takes place in transforming $S$ to $T$. Now suppose the maximal weakenings of $r$ are $\{r_1, \ldots, r_m\}$ then $\sigma(s_1)$ and the $\sigma(t_i)$ each have $n$ maximal weakenings where $n$ is a multiple of $m$ dependent on the number of occurrences of $x$ in the skeleton of $s_1$. In particular, $\mathit{weakenings}(\sigma(s_1)) = \{s_{1,1}, \ldots, s_{1,n}\}$ and for each $t_i$, $\mathit{weakenings}(\sigma(t_i)) = \{t_{i,1}, \ldots, t_{i,n}\}$. Again we may speak of "partners": each $s_{1,j}$ has as partners $t_{i,j}$, for $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, n\}$ where the weakenings of $t_{i,j}$ come from weakening the occurrences $t$ identically to their weakenings in $s_{1,j}$. Furthermore, because for each $i \in \{1, \ldots, p\}$, $s_1 \succ t_i$, we can use case 1 to conclude that each maximal weakening of $\sigma(s_1)$ is larger than its partners. Hence replacing each $s_{1,i}$ with its partners defines an appropriate transformation from $\mathit{weakenings}(\sigma(s))$ to $\mathit{weakenings}(\sigma(t))$. ∎

As $\succ^\star$ is an annotation reduction ordering we can conclude:

THEOREM 8. *Rippling using proper rewrite rules $l \to r$ for which $l \succ^\star r$ is terminating.*

## 6. Implementing Rippling

We have completed our development of a calculus for rippling and termination orderings for annotated terms. We now consider the more practical problem of mechanizing such a calculus. In particular, given an ordering, how do we then recognize wave-rules and apply them? We have implemented the rewrite calculus described and here we indicate how the simplicity of our calculus and orderings led to a simple implementation, which now comprises part of the Edinburgh CLAM

system. To give the reader a feel for this, and the issues involved, we briefly sketch a couple of the core routines.

Much of the work in implementing rippling concerns turning unannotated rewrite rules into wave-rules; we call this *wave-rule parsing*. A wave-rule parser must, given unannotated rewrite rules, return wave-rules, that is a collection of annotated copies of the rule that are proper rewrite rules and measure decreasing. We can achieve the requirements of proper annotation and measure decreasingness separately. An *annotation phase* first annotates $l$ and $r$ with unoriented wave-fronts so their skeletons are identical; this guarantees that rippling is skeleton preserving. An *orientation phase* then orients the wave-fronts with up and down arrows so that $l \succ r$. We sum this up by the slogan

$$\mathit{WAVE\text{-}RULE} = \mathit{ANNOTATION} + \mathit{ORIENTATION}\,.$$

As an example, consider parsing a rewrite rule like

$$s(U) \times V \to (U \times V) + V\,. \tag{16}$$

We may proceed by annotating this so the two sides have identical skeletons, i.e.,

$$\boxed{s(\underline{U})} \times V \to \boxed{(\underline{U \times V}) + V}\,. \tag{17}$$

Afterwards we can orient the annotation yielding the wave-rule,

$$\boxed{s(\underline{U})}^{\uparrow} \times V \to \boxed{(\underline{U \times V}) + V}^{\uparrow}\,. \tag{18}$$

Both sides of (12) now have the same skeleton and the measure of the left-hand side is greater than that of the right-hand side.

Any implementation, however, must cope with the problem that under our definition of wave-rules, a given rewrite rule can generate exponentially many (in the size of the input rule) wave-rules. Computing and storing all possible wave-rules is expensive both in time and space and complicates efficient wave-rule lookup. For example, in the previous example, there are other possible legal parsing such as:

$$\boxed{s(\underline{U}) \times V}^{\uparrow} \to \boxed{\underline{U} \times V + V}^{\downarrow} \tag{19}$$

$$\boxed{s(U) \times \underline{V}}^{\uparrow} \to \boxed{U \times \underline{V} + V}^{\downarrow} \tag{20}$$

$$\boxed{s(U) \times \underline{V}}^{\uparrow} \to \boxed{U \times V + \underline{V}}^{\downarrow} \tag{21}$$

These additional parsings are problematic; while they are not really in the "spirit" of wave-rules as originally proposed by [6] (nor admissible under their definition), and are seldomly useful in practice, they are admissible under our more liberal definition and on occasion find use in, for example, wave-front normalization

```
rewrite(T,NT) :-                    % rewrite at some term position
  subterm(Pos,T,ST),                % find a subterm ST in T
  pick_rule(L,R),                   % pick a rule L -> R
  match_rule(L,R,ST,NR),            % can rule be annotated to match ST
  replace(Pos,T,NR).                % replace subtern ST with NR

match_rule(L,R,ST,NR) :-
  copy_an(ST,L,AL),                 % copy annotation from ST onto L
  amatch(AL,ST,Sigma),              % annotated match of AL with ST
  parse(AL,R,AR),                   % find annotations for R
  apply_subs(Sigma,AR,NR).          % apply substitution to AR

parse(AL,R,AR) :-
  pick_an(R,A),                      % annotate R
  skel_preserving(AL,A),             % skeletons equal?
  orient(AL,A,AR).                   % Orient R
```

Fig. 3. Wave-rule parser (Top Level Routines)

(we discuss this in §7.1). Rather than trying to say in advance which wave-rules could be useful in practice, our solution to this problem is to compute wave-rules dynamically, by parsing "on demand". We describe this in the following section.

### 6.1. Dynamic Wave-rule Parsing

We have implemented a *dynamic parser* that, given a data-base of unannotated rewrite rule, uses them for rippling by annotating them only as required during rewriting. That is, given a term $s[t]$ to be rewritten, we look for an unannotated rule $l \to r$ where $l$ matches the erasure of $t$. When this is the case, $l \to r$ is a candidate wave-rule. We proceed by computing annotations for $l$ that allow for an annotated match with $t$; afterwards, based on these annotations and their orientations, we compute annotations and orientations for $r$ so that $l \to r$ is a wave-rule.

Figure 3 contains the actual Prolog (we assume the reader is familiar with Prolog in the following) that implements the top level "architecture" of rippling based on dynamic parsing. We illustrate the procedure through an example. Suppose we wish to perform one step of rewriting of the term $T = \boxed{s(\underline{x})}^{\uparrow} \times \boxed{s(\underline{y})}^{\uparrow}$ using a collection of unannotated rewrite rules that includes (16), the recursive definition of multiplication. `Rewrite` picks a subterm $ST$ of $T$ and a rule $L \to R$. In our example, a solution to this (there may be others which are returned on backtracking) is where $ST$ is $T$ itself and the rule selected is (16). In dynamic parsing we need only generate annotations for the right-hand sides of wave-rules whose left-hand sides arise during the proof. This is performed in `match_rule` which starts by copying annotations from $ST$ onto $L$; this yields $AL$, an anno-

tated version of $L$. Copying annotation fails if $ST$ and $L$ have erasures which do not match. In our example, $AL = \boxed{s(\underline{U})} \times V$. We then annotated match $AL$ with $ST$ to find a suitable substitution for the rewriting step. In our example, we get the substitution $\{ U/x, V/\boxed{s(\underline{y})}^{\uparrow} \}$. After, we call `parse` to find an annotation of $R$ with the same skeleton as $AL$, and with a maximum[2] orientation, in this case $\boxed{(\underline{U \times V}) + V}^{\uparrow}$. We then apply the substitution to this annotated right-hand side, giving $\boxed{(\underline{x \times \boxed{s(\underline{y})}^{\uparrow}}) + s(y)}^{\uparrow}$. Note that substitution application uses our new notion of subterm replacement (from §4). Here, regular substitution would generate an improperly annotated term. The final step in rewriting is to replace the rewritten subterm within the context of the superterm from which it came, again using our new notion of subterm replacement.

The supporting subroutines for parsing are fairly straightforward. We used generate (`pick_an`) and test (`skel_preserving`) to generate skeleton preserving annotations of the right-hand side of rewrite rules. This takes exponential time but the efficiency can be considerably improved by interleaving generation and testing (i.e., testing during generation) or via dynamic programming. A dynamic programming approach to such *difference matching* problems is described in [1]. In our experience, naive generate and test has acceptable performance.

The routine `orient` finds an orientation of the wave-fronts on the right-hand side that yields a measure smaller than the left-hand side. This can be implemented naively by generating orientations (there are two possibilities for each wave-front) and comparing the two sides of the proposed rule under the given measure. By comparing possible orientations against each other, we can return the maximum possible right-hand side orientations. As with annotation, there are algorithms to implement orientation more efficiently. In particular when all annotation is simple (single wave-holes) it is possible to orient the right-hand side in linear time (in the size of the term). An algorithm for this is given in [2].

### 6.2. Sinks and Colours

One kind of annotation we have not discussed in our measures or parsing is *sinks*. This is deliberate as we can safely ignore sinks in both the measure and the parser. Sinks only serve to decrease the applicability of wave-rules by creating additional preconditions; that is, we only ripple inwards if there is a sink underneath the wave-front. Hence sinks decrease the search space of rippling and termination without this restriction implies termination with this restriction. The value of sinks is they restrict search without reducing the *utility* of rippling: their use guides rippling in a way that allows the induction hypothesis to be successfully applied.

[2] Maximum under our order. When there are multiple choices with the same measure we return them all on backtracking.

Another type of annotation introduced in [17] are wave-holes marked with colours. Different colours are used to distinguish different skeletons. Colours are useful in inductive proofs with multiple induction hypotheses (for example, inductions on binary trees). The motivation behind the introduction of colours is that rippling only preserves a subset of the skeletons, and colours helps prevent us ending up with the wrong subset. Since coloured rippling is a restriction of uncoloured rippling, termination follows immediately from termination in the uncoloured case. Colours thus increase the utility of rippling. Although colours are not needed for showing the termination of rippling, they actually played an implicit role in our discussion about termination. The reduction order defined in §5 compares the measures of different colours separately. Since each colour has a single skeleton, and the measure is stable for single skeletons, the resulting order is stable.

## 7. Extensions to Rippling

Our definition of wave-rules is parameterized by a reduction ordering. This gives us flexibility in exploring new orderings and hence new applications for rippling. To illustrate this potential, we give two examples for inductive and non-inductive theorem proving. The first highlights a problem occurring in induction: during rippling we may need to normalize the contents of wave-fronts when rippling gets stuck. The second explores orderings useful for algebraic problem solving.

### 7.1. Unblocking

Here we consider new reduction orderings motivated by unblocking rippling. That is, sometimes rippling fails because no wave-rule is applicable, but not all wave-fronts have been moved "out of the way" (to the root of the term or to sinks). This can occur because a lemma is needed; these missing wave-rules can sometimes be speculated automatically using techniques presented by Ireland and Bundy in [15]. Rippling can also become blocked simply because a wave-front itself needs to be rewritten so that it matches either an existing wave-rule (to allow further rippling) or a sink (to allow use of the induction hypothesis). This is best illustrated by an example taken from [6].

Consider the following theorem, where $rev$ is naive reverse, $qrev$ is tail-recursive reverse using an accumulator, $<>$ is infix append, and :: infix cons.

$$\forall L, M.\, qrev(L, M) = rev(L) <> M \tag{22}$$

We proceed by induction on $L$. The induction hypothesis is

$$qrev(l, M) = rev(l) <> M$$

where $M$ is a universally quantified variable. The induction conclusion is

$$qrev(\boxed{h :: \underline{l}}^{\uparrow}, \lfloor m \rfloor) \;=\; rev(\boxed{h :: \underline{l}}^{\uparrow}) <> \lfloor m \rfloor \tag{23}$$

where $m$ is a skolem constant which sits in a sink, annotated with "$\lfloor\ \rfloor$".

We use wave-rules taken from the recursive definition of $qrev$, and $rev$.

$$rev(\boxed{H :: \underline{T}}^{\uparrow}) \rightarrow \boxed{\underline{rev(T)} <> (H :: nil)}^{\uparrow} \tag{24}$$

$$qrev(\boxed{H :: \underline{T}}^{\uparrow}, L) \rightarrow qrev(T, \boxed{H :: \underline{L}}^{\downarrow}) \tag{25}$$

On the LHS, we ripple with (25) to give

$$qrev(l, \left\lfloor \boxed{h :: \underline{m}}^{\downarrow} \right\rfloor) = rev(\boxed{h :: \underline{l}}^{\uparrow}) <> \lfloor m \rfloor .$$

The sink stays in the same position relative to the skeleton and absorbs the wave-front rippled across by (25). On the RHS, we ripple with (24) and then (8), the associativity of $<>$, to get

$$qrev(l, \left\lfloor \boxed{h :: \underline{m}}^{\downarrow} \right\rfloor) = rev(l) <> (\left\lfloor \boxed{(h :: nil) <> \underline{m}}^{\downarrow} \right\rfloor). \tag{26}$$

Again note how the sink has absorbed the wave-front rippled across. Unfortunately, the proof is now blocked. We cannot ripple any further nor apply the induction hypothesis. The problem is that we need to simplify the wave-front on the right-hand side so that the two sinks become identical. CLAM currently uses an ad-hoc method to try to perform wave-front simplification when rippling becomes blocked. In this case (26) is rewritten using conventional rewriting to

$$qrev(l, \left\lfloor \boxed{h :: \underline{m}}^{\downarrow} \right\rfloor) = rev(l) <> (\left\lfloor \boxed{h :: \underline{m}}^{\downarrow} \right\rfloor) .$$

Simplification with the induction hypothesis can now occur.

Unblocking steps that manipulate just wave-fronts will use proper rewrite rules; for example, here we use another parsing for the recursive definition of append.

$$\boxed{(H :: T) <> \underline{L}}^{\downarrow} \rightarrow \boxed{H :: (T <> \underline{L})}^{\downarrow} \tag{27}$$

In [6] such a rule is not admitted as a wave-rule (see §8). It is also not admissible under our ordering $\succ^{\star}$ as $\succ^{\star}$ measures the *width* of wave-fronts and the right-hand side is wider than the left-hand side. However, both sides have the same size (number of function symbols and constants). If we extend our measure to account for the contents of wave-fronts then we can find a reduction ordering based on size of wave-fronts that includes the above rule.

We do this as follows. As before, we give an ordering on simply annotated terms, which can then be lifted to an order on multi-wave terms. To order simply annotated terms, we take the lexicographic order of the simple wave-rule measure proposed above (using size of the wave-front as the notion of weight) paired with

$>_{wf}$, an order on the *contents* of wave-fronts. As a simply annotated term may contain multiple wave-fronts, this second order is lifted to a measure on sets of wave-fronts by taking its multi-set extension. The first part of the lexicographic ordering ensures that anything which is normally measure decreasing remains measure decreasing and the second part allows us to orient rules that only manipulate wave-fronts. This combination can provide a reduction ordering that allows us to use rippling to move wave-fronts about the skeleton and conventional rewriting to manipulate the contents of these wave-fronts.

In our reverse example, (27) doesn't change the size of the wave-front or its position but only its form. Hence we want this to be decreasing under some ordering on the contents of wave-fronts. There are many such orderings; here we take $>_{wf}$ to be the recursive path ordering [12] on the terms in the wave-front where $<>$ has a higher precedence than :: and all other function symbols have an equivalent but lower priority. The measure of the LHS of (27) is now greater than that of the RHS as its wave-front is $(H :: T) <> *$ which is greater than $H :: (T <> *)$ in the recursive path ordering (to convert wave-fronts into well-formed terms, wave-holes are marked with the new symbol *).

Unblocking steps which simplify wave-fronts are useful in many proofs enabling both immediate application of the induction hypothesis (as in this example) and continued rippling. By defining new orders we can combine rippling with conventional term rewriting so that rules to rewrite wave-fronts are measure decreasing wave-rules accepted by the parser and applied like other wave-rules.

### 7.2. Algebraic Problem Solving

Rippling has found several novel uses besides inductive theorem proving. For example, it has been used to sum series [16], to prove limit theorems [17], and guide equational reasoning [10]. However, new domains, especially non-inductive ones, require new orderings to guide proof construction. Here we sketch an application based on the PRESS system [9].[3] To solve algebraic equations, PRESS uses a set of methods which apply rewrite rules. The three main methods are: *isolation*, *collection*, and *attraction*. Below are examples of rewrite rules used by these methods.

$$\begin{array}{ll} \text{ATTRACTION :} & \boxed{\log(\underline{U}) + \log(\underline{V})}^{\uparrow} \rightarrow \boxed{\log(\underline{U} \times \underline{V})}^{\uparrow} \\ \text{COLLECTION :} & \boxed{\underline{U} \times \underline{U}}^{\uparrow} \rightarrow \boxed{\underline{U}^2}^{\uparrow} \\ \text{ISOLATION :} & \boxed{\underline{U}^2}^{\uparrow} = V \rightarrow U = \boxed{\pm\sqrt{\underline{V}}}^{\downarrow} \end{array}$$

PRESS uses preconditions and not annotation to determine rewrite rule applicability. Attraction must bring occurrences of unknowns closer together. Collection must reduce the number of occurrences of unknowns. Finally, isolation must make

[3] The idea of reconstructing PRESS with rippling was suggested by Nick Free and Alan Bundy.

progress towards isolating unknowns on the LHS of the equation. These requirements can be captured by annotation and PRESS can thus be implemented by rippling. The above wave-rules suggest how this would work. The wave-rules in PRESS are structure preserving, where the preserved structure is the unknowns. The ordering used reflects the well-founded progress achieved by the PRESS methods. Namely, we lexicographically combine orderings on the number of wave-holes for collection, their distance (shortest path between wave-holes in term tree) for attraction, and our width measure on annotation weight for isolation.

## 8. Related Work

### 8.1. CLAM

Our starting point is rippling as developed at Edinburgh and implemented in the CLAM proof planning system. Our results improve those presented in [6] in a number of respects.

To begin with, rippling as described in [6] is not a rewriting calculus. Rather it is implemented by first-order rewriting with the strong precondition that "...each wave-front in the expression [being rewritten] is matched with a wave-front of the same kind in the rule" (Definition 5, page 222). Saying this another way, variables in wave-rules cannot be instantiated with annotated terms. This is sufficient for rippling to be structure preserving and terminating, but it is an unacceptably large restriction on the applicability of rippling. Indeed, under this restriction, not all of the examples in [6] are valid. For example (see page 222) we cannot rewrite the immediate subterm of

$$even(\boxed{s(\underline{\boxed{s(\underline{x})}^{\uparrow}})}^{\uparrow} + y)$$

with the recursive definition of plus given by $\boxed{s(\underline{U})}^{\uparrow} + V \rightarrow \boxed{s(\underline{U+V})}^{\uparrow}$, since the left-hand side of this wave-rule is $\boxed{s(\underline{U})}^{\uparrow} + V$ and there is an extra wave-front in the subterm being rewritten.

Rippling was implemented in the CLAM system without the above restriction and it suffered from the problems described in §3 that arise when first-order rewriting is used to implement rippling directly. In particular, ill-formed terms appeared during rewriting and an auxiliary routine occasionally would "clean-up" annotations (e.g., consider the multiplication example given in §3). The CLAM implementation of rippling has been replaced with the calculus and parser described here.

The measures and orders we give are considerably simpler than those in [6] where the properties of structure preservation and the reduction of a measure are intertwined. Bundy *et al.* describe wave-rules schematically and show that any

instance of these schemata is skeleton preserving and measure decreasing under an appropriately defined measure. Mixing these two properties makes the definition of wave-rules very complex. For example, the simplest kind of wave-rule proposed are *longitudinal* outward directed wave-rules, defined as rules of the form,

$$\eta(\boxed{\xi_1(\underline{\mu_1^1},\ldots,\underline{\mu_1^{p_1}})}^{\uparrow},\ldots,\boxed{\xi_n(\underline{\mu_n^1},\ldots,\underline{\mu_n^{p_n}})}^{\uparrow}) \rightarrow \boxed{\zeta(\underline{\eta(\varpi_1^1,\ldots,\varpi_n^1)},\ldots,\underline{\eta(\varpi_1^k,\ldots,\varpi_n^k)})}^{\uparrow}$$

that satisfy a number of side conditions. These include: each $\varpi_i^j$ is either an unrippled wave-front, $\boxed{\xi_i(\underline{\mu_i^1},\ldots,\underline{\mu_i^{p_i}})}$, or is one of the wave-holes, $\mu_i^l$; for each $j$, at least one $\varpi_i^j$ must be a wave-hole. $\eta$, the $\xi_i$s, and $\zeta$ are terms with distinguished arguments; $\zeta$ may be empty, but the $\xi_i$s and $\eta$ must not be. There are other schemata for *transverse* wave-rules and *creational* wave-rules[4]. These schemata are combined in a general format, so complex that in [6] it takes four lines to print. It is notationally involved although not conceptually difficult to demonstrate that any instance of these schemata is a wave-rule under both our size and width measures.

Consider the longitudinal schema given above. Every skeleton on the LHS is a skeleton of the RHS because of the constraint on the $\varpi_j^i$. What is trickier to see is that it is measure decreasing. Under our order this is the case if LHS $\succ^\star$ RHS. We can show something stronger, namely, for every $r \in \mathit{weakenings}(RHS).\exists l \in \mathit{weakenings}(LHS).l \succ r$. To see this observe that any such $r$ must be a maximal weakening of

$$r' = \boxed{\zeta(\eta(\varpi_1^1,\ldots,\varpi_n^1),\ldots,\underline{\eta(\varpi_1^j,\ldots,\varpi_n^j)},\ldots\eta(\varpi_1^k,\ldots,\varpi_n^k))}^{\uparrow}$$

for some $j \in \{1,\ldots,k\}$. Corresponding to $r'$ is an $l'$ which is a weakening of the LHS where $l' = \eta(t_1,\ldots,t_n)$ and the $t_i$ correspond to the $i$th subterm of $\eta(\varpi_1^j,\ldots,\varpi_n^j)$ in $r'$: if $\varpi_i^j$ is an unrippled wave-front then $t_i = \varpi_i^j = \boxed{\xi_i(\underline{\mu_i^1},\ldots,\underline{\mu_i^{p_i}})}$, and alternatively if $\varpi_i^j$ a wave-hole $\mu_i^l$ then $t_i = \boxed{\xi_i(\mu_i^1,\ldots,\underline{\mu_i^l},\ldots,\mu_i^{p_i})}$. Now $r$ is a maximal weakening of $r'$ so there is a series of weakening steps from $r$ to $r'$. Each of these weakenings occurs in a $\varpi_i^j$ and we can perform the identical weakening steps in the corresponding $t_i$ in $l'$ leading to a maximal weakening $l$. $l$ and $r$ have the same skeleton and as they are maximally weak they may be compared under $\succ$. Their only differences are that $r$ has an additional wave-front at its root and is missing a wave-front at each $\varpi_i^j$ corresponding to a wave-hole. The depth of $\varpi_i^j$ is greater than the root and at this depth the out-measure of $l$ is greater than $r$ and at all greater depths they are identical. Hence $l \succ r$.

[4] Creational wave-rules are used to move wave-fronts between the hypotheses and conclusion during proofs by destructor induction. They complicate rippling in a rather specialized and uninteresting way. Our measures could be easily generalized to include these.

Similar arguments hold for the other schemata given in [6] and from this we can conclude that wave-rules acceptable under their definition are acceptable under ours. Moreover simple examples are wave-rules under our formalism but not theirs, e.g., the base-case of addition $\boxed{0 + \underline{x}}^{\uparrow} \rightarrow x$.

### 8.2. INKA

Hutter, in [14, 13], describes a calculus for rippling implemented in the INKA system [3]. Hutter rigorously develops an algebra of annotated terms, called C-terms. These are terms in an extended signature where functions and variables each carry a "colour", which represents annotation, or a variable over colours, which restricts potential annotation. Hutter's motivations and developments are similar: he defines congruence relations corresponding to equality of terms after erasure, equivalence of skeletons, and develops algorithms to unify and rewrite C-terms that respect these congruences.

The calculus he develops is more general than ours. However, it is significantly more complex, both conceptually, and in implementation. Wave-fronts can be thought of as contexts. In our calculus we augment the signature only as is required to specify these contexts: i.e., we introduce new function symbols so that we may mark the beginning of a context with a wave-front, and the end of the context with wave-holes. In Hutter's calculus, annotation is the primary concept and matching and rewriting of such terms can be understood independently of contexts.

Hutter has not addressed termination in his work. However, with minor restrictions on his calculus, our results should carry over. For example, we can consider a setting with three colours (indicating skeleton, inwards wave-fronts and outwards wave-fronts) restricted to C-terms which are *wats*. In this setting we can define the same kinds of well-founded orderings on terms based on annotation relative to the skeleton. It should be possible to carry over our proofs of stability and monotonicity in his setting, although we have not formally checked this.

## 9. Conclusions

We have defined a simple calculus for rippling where differences between the induction conclusion and the induction hypothesis are marked with annotations and annotated rewrite rules move these differences away. We have proved that rewriting in this calculus has various desirable properties: well-formedness (well-annotated terms rewrite to well-annotated terms), skeleton preservation (the unannotated part of terms are preserved), and correctness (the corresponding rewriting can be performed in the original unannotated theory, *i.e.* annotation merely guides search). We have shown how this calculus admits simple termination orders which are stable and monotonic. As well as providing a firmer theoretical foundation

to rippling, this work has led to a simpler and more complete implementation of rippling within the Edinburgh CLAM system.

## References


1. David Basin and Toby Walsh. Difference unification. In *Proceedings of the 13th IJCAI*, pages 116–122. International Joint Conference on Artificial Intelligence, 1993.
2. David Basin and Toby Walsh. Termination orderings for rippling. In *Proc. of 12th International Conference On Automated Deduction (CADE-12)*, Nancy, France, June 1994. Springer-Verlag.
3. Susanne Biundo, Birgit Hummel, Dieter Hutter, and Christoph Walther. The Karlsruhe induction theorem proving system. In *8th International Conference On Automated Deduction*, Oxford, UK, 1986.
4. Robert S. Boyer and J. Strother Moore. *A Computational Logic*. Academic Press, 1979.
5. Alan Bundy, Frank van Harmelen, Jane Hesketh, and Alan Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7:303–324, 1991.
6. Alan Bundy, Andrew Stevens, Frank van Harmelen, Andrew Ireland, and Alan Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993.
7. Alan Bundy, Frank van Harmelen, Christian Horn, and Alan Smaill. The Oyster-Clam system. In M.E. Stickel, editor, *10th International Conference on Automated Deduction*, pages 647–648. Springer-Verlag, 1990. Lecture Notes in Artificial Intelligence No. 449.
8. Alan Bundy, Frank van Harmelen, Alan Smaill, and Andrew Ireland. Extensions to the rippling-out tactic for guiding inductive proofs. In M.E. Stickel, editor, *10th International Conference on Automated Deduction*, pages 132–146. Springer-Verlag, 1990. Lecture Notes in Artificial Intelligence No. 449.
9. Alan Bundy and Bob Welham. Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation. *Artificial Intelligence*, 16(2):189–212, 1981.
10. Jürgen. Cleve and Dieter Hutter. A methodology for equational reasoning. In *HICSS-27*. IEEE, 1994. To appear.
11. Nachum Dershowitz. Termination of rewriting. In J.-P. Jouannaud, editor, *Rewriting Techniques and Applications*, pages 69–116. Academic Press, 1987.
12. Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, March 1982.
13. Dieter Hutter. Colouring terms to control equational reasoning. An Expanded Version of PhD Thesis: Mustergesteuerte Strategien für Beweisen von Gleichheiten (Universität Karlsruhe, 1991), in preparation.
14. Dieter Hutter. Guiding inductive proofs. In M.E. Stickel, editor, *10th International Conference on Automated Deduction*, pages 147–161. Springer-Verlag, 1990. Lecture Notes in Artificial Intelligence No. 449.
15. Andrew Ireland and Alan Bundy. Productive Use of Failure in Inductive Proof. *Journal of Automated Reasoning*, this issue.
16. Toby Walsh, Alex Nunes, and Alan Bundy. The use of proof plans to sum series. In D. Kapur, editor, *11th Conference on Automated Deduction*, pages 325–339. Springer Verlag, 1992. Lecture Notes in Computer Science No. 607.
17. Tetsuja Yoshida, Alan Bundy, Ian Green, Toby Walsh, and David Basin. Coloured rippling: An extension of a theorem proving heuristic. In *ECAI-94*. John Whiley and Sons, 1994.