Rippling: A Heuristic for Guiding Inductive Proofs

Alan Bundy, Andrew Stevens, Frank van Harmelen,
Andrew Ireland and Alan Smaill

DAI Research Paper No. 567

February 5, 1992

Submitted to the Artificial Intelligence Journal

Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland

Rippling: A Heuristic for Guiding Inductive Proofs *

Alan Bundy Andrew Stevens[†] Frank van Harmelen[‡]

Andrew Ireland Alan Smaill

Abstract

We describe rippling: a tactic for the heuristic control of the key part of proofs by mathematical induction. This tactic significantly reduces the search for a proof of a wide variety of inductive theorems. We first present a basic version of rippling, followed by various extensions which are necessary to capture larger classes of inductive proofs. Finally, we present a generalised form of rippling which embodies these extensions as special cases. We prove that generalised rippling always terminates, and we discuss the implementation of the tactic and its relation with other inductive proof search heuristics.

Key words and phrases. Automated theorem proving, mathematicl induction, proof plans, tactics, rippling.

1 Introduction

In [Bundy 88] we introduced the concept of proof plans and gave a simple example proof plan for guiding inductive proofs. A proof plan is an outline of a family of proofs which can be used to guide the search for proofs in this family. Many simple inductive proofs fit the outline of our inductive proof plan. Figure 1 describes its basic structure.

The key idea of this inductive proof plan is a tactic for manipulating the induction conclusion to enable the induction hypothesis to be used in its proof. Following Aubin, [Aubin 75], we initially called this tactic *ripple-out*. In [Bundy *et al* 91] we described an implementation of this inductive proof plan within *Oyster-CIAM*, a theorem prover for a higher-order, intuitionist, typed logic, and we reported its performance on various standard example inductive theorems. *Oyster* is a reimplementation in Prolog of the Nuprl interactive proof editor, [Constable *et al* 86]. Tactics are Prolog programs which drive *Oyster* by applying its rules of inference. A proof plan is a tactic together with its *method*. The method of a tactic is a partial specification of it consisting of *preconditions* under which it is applicable and the *effects* of its application. These are written in a meta-logic. *CIAM* is a plan formation program which uses these methods to build a special-purpose proof plan for each theorem from a set of general-purpose proof plans.

In [Bundy *et al* 89] we described the relationship between our inductive proof plan and the technique used by Boyer and Moore, [Boyer & Moore 79], to choose an appropriate induction rule

*The research reported in this paper was supported by SERC grant GR/F/71799, an SERC Senior Fellowship to the first author and an SERC Postdoctoral Fellowship to the second author. We wish to thank our colleagues in the Edinburgh Mathematical Reasoning Group and three anonymous CADE-10 referees for feedback on this paper. An earlier, much shorter version of this paper appeared in the proceedings of CADE-10.

†Current affiliation: Oxford University Computing Laboratory.

‡Current affiliation: Department of Social Science Informatics (SWI), University of Amsterdam.

```
ind_strat(IndTerm(X),X) ≡
      induction(IndTerm(X),X) then
                                  [ sym_eval,
                                    ripple then fertilize
                                  ]
```

X is the induction variable and IndTerm is the wave front introduced into the induction conclusion by the induction rule of inference. Induction rules are indexed by their IndTerms. The application of the induction tactic exchanges the original theorem for a list of base and step cases: only one of each is shown in the figure. The sym_eval tactic is applied to the base case. This tactic is a combination of symbolic evaluation, tautology checking, equality substitution, etc. *The ripple tactic is applied first to the step case. This creates a copy of the induction hypothesis within the induction conclusion. The fertilize tactic then uses the induction hypothesis to prove the induction conclusion. Any remaining subgoals are proved recursively using sym_eval and ind_strat.*

Figure 1: The Inductive Proof Plan

and induction variable. We showed how we rationally reconstructed the Boyer-Moore technique as a process of look-ahead whose purpose is to maximise the chances that rippling-out would succeed.

We have tested our implementation extensively on a large number of inductive theorems, most of which are drawn from the Boyer-Moore corpus, [Boyer & Moore 79, appendix A], adapting the original proof plan as necessary. This paper reports the result of that study. The spirit of the proof plan has survived these tests, with rippling-out remaining the central idea of the proof plan. However, the tests have suggested principled extensions to the proof plan, which preserve the essential intuition behind it while increasing its range of application. The main result is a generalisation of the *ripple-out* tactic. Since this generalisation includes rippling in other directions than 'out', we have named the generalised tactic, *ripple*.

In this paper we give a complete account of rippling which supercedes earlier accounts in [Bundy *et al* 91, Bundy *et al* 90b]. We start with a simple introduction to the original *ripple-out* tactic and then motivate and explain various extensions, finishing with a description of the generalised *ripple* tactic, which embodies all these extensions in a uniform framework. We show that *ripple* terminates. The extensions to rippling-out reported below are: rippling with multi-wave-rules, rippling with conditional wave-rules, rippling-in, rippling-sideways, rippling-across and rippling under existential quantifiers. Each extension is illustrated with simple examples of proofs which require them. We argue that the extensions are natural improvements of the original idea.

2 The Basic Rippling-Out Tactic

In this section we briefly introduce the main idea of rippling-out and the role it plays within our inductive proof plan.

To understand rippling-out imagine a loch[1] in which the induction conclusion appears as a reflection of the hypothesis. The reflection is not a perfect image of the original because wherever the induction variable appears in the induction hypothesis, the induction term appears in the induction conclusion. The expressions which appear in the induction conclusion, but not in the induction hypothesis, we call *wave-fronts*. The rest of the induction conclusion, *i.e.* the expressions which *do* appear also in the induction hypothesis, we call the *skeleton*, following [Hutter 90]. Wave-fronts are like ripples on the surface of the loch, which spoil the reflection. Consider, for instance, a simple proof of the associativity of +:[2]

$$\forall X{:}nat.\forall Y{:}nat.\forall Z{:}nat.\ X + (Y + Z) = (X + Y) + Z$$

by successor induction on X. The induction hypothesis is

$$x + (y + z) \quad = \quad (x + y) + z \tag{1}$$

where x, y and z represent skolem constants[3]. The induction conclusion is

$$\boxed{s(\underline{x})} + (y + z) \quad = \quad (\boxed{s(\underline{x})} + y) + z \tag{2}$$

The induction term is $s(x)$ and the $s(\ldots)$ constructor function is the wave-front. $x + (y + z) = (x + y) + z$ is the skeleton.

2.1 Wave-Fronts and Wave-Rules

More generally, a *wave-front* is a term from which a proper subterm is deleted, *i.e.* a sequence of nested functions with the innermost argument removed. This innermost argument (the x in (2)) is the *wave-hole*. When the wave-hole is filled by a term the wave-front is said to *dominate* the term. We adopt the convention that wave-fronts are indicated by boxes with the wave-holes underlined, as in the example above. An alternative graphical representation of wave-fronts is given in figure 2.

Initially, the wave-fronts are functions which immediately dominate the induction variable. The role of rippling-out is to move them outwards — just like the ripples on a loch — leaving behind them an unspoilt reflection of the induction hypothesis. Rippling-out works by backwards reasoning from the induction conclusion to the induction hypothesis using *wave-rules*. A wave-rule is a rewrite rule of the form:

$$\eta(\boxed{\xi(\underline{\mu})}) \quad \Rightarrow \quad \boxed{\zeta(\underline{\eta(\mu)})} \tag{3}$$

where η, ξ and ζ are terms[4] with one distinguished argument. μ is usually a variable, but can be any term. ζ may be empty, but ξ and η must not be. ξ and ζ are called the *old* and *new* *wave-fronts*, respectively. Note that the effect of applying such a rule is to move the old wave-front ξ, in the induction conclusion, outwards past the η and to turn it into a new wave-front ζ.

We adopt the convention that $\Rightarrow$ stands for rewriting and $\rightarrow$ stands for implication. Recall that rippling-out reasons backwards: from the theorem towards the axioms . To apply the implication $\phi \rightarrow \psi$ it uses the rewrite rule $\psi \Rightarrow \phi$. Thus the left/right orientation of implication is opposite to that of rewriting.

[1] The Scottish word for 'lake'.

[2] To improve readability we have translated the type-theoretic logic used by *Oyster-CLAM* into a more conventional notation.

[3] We adopt the Prolog convention that identifiers starting with upper case letters indicate variables and those starting with lower case letters indicate constants.

[4] We adopt the convention of using greek letters for meta-symbols, *e.g.* for describing the patterns of wave-rules.

$$
\begin{aligned}
\boxed{s(\underline{U})} + V &\Rightarrow \boxed{s(\underline{U+V})} && (4)\\
\boxed{s(\underline{U})} \times V &\Rightarrow \boxed{\underline{U \times V} + V} && (5)\\
even(\boxed{s(s(\underline{U}))}) &\Rightarrow even(U) && (6)\\
U + (\boxed{\underline{V} + W}) &\Rightarrow \boxed{(\underline{U+V}) + W} && (7)\\
(\boxed{U + \underline{V}}) + W &\Rightarrow \boxed{U + (\underline{V+W})} && (8)
\end{aligned}
$$

Wave-rules (4), (5) and (6) are formed from recursive definitions. Wave-rules (7) and (8) are both formed from the associative law of +, which is not a recursive definition. Note that the wave-front of rule (6) is compound, i.e. *contains more than one function symbol, and that the ζ part of this rule is empty. Note also that rules (7) and (8) show that an equation can give rise to several different wave-rules and that these can be oriented in different directions provided that the wave-front annotations are different.*

Table 1: Examples of Wave Rules

This general form includes all rewrite rules formed from the step cases of recursive definitions. Restricted to such recursive rules, rippling-out is a constrained version of unfolding or symbolic evaluation. But, as we will see, wave-rules can also be formed from non-recursive definitions and from lemmas, so rippling-out extends unfolding. *CLAM* analyses all the definitions given to, and lemmas proved by, *Oyster* and transforms them into wave rules in as many ways as possible. Examples of wave rules can be found in table 1.

Definition 1 (Preconditions of Applying Wave-Rules) *The preconditions of applying a wave-rule to rewrite a sub-expression of the induction conclusion are that:*

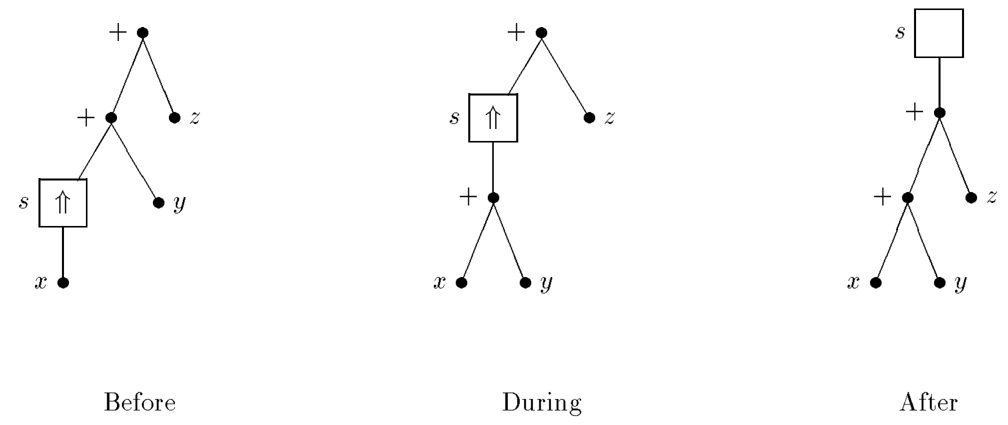
- *the left hand side of the rule matches the sub-expression;*
- *the sub-expression contains at least one wave-front; and*
- *this wave front in the sub-expression is matched with the old wave-front in the rule.*

These preconditions ensure that wave-rules always move wave-fronts outwards towards the root of the skeleton. Thus wave rules are applied selectively, not exhaustively.

Repeated application of rule (4) from table 1 to induction conclusion (2) ripples the two wave-fronts to right outside the left and the right hand terms of the induction conclusion, as follows:

$$
\begin{aligned}
\boxed{s(\underline{x})} + (y+z) &= (\boxed{s(\underline{x})} + y) + z\\
\boxed{s(\underline{x + (y+z)})} &= \boxed{s(\underline{x+y})} + z\\
\boxed{s(\underline{x + (y+z)})} &= \boxed{s(\underline{(x+y)+z})} && (9)
\end{aligned}
$$

A graphical representation of this rippling-out is given in figure 2.



The three trees show the term before, during and after rippling-out. Each tree represents the term as an expression tree. The nodes of each tree are labelled by a function or predicate symbol or a constant or variable. A function or predicate of arity n has n descendent subtrees: one for each of its arguments. Wave-fronts are indicated by square nodes and all other nodes by dots. Note that the three trees differ from each other only in the position of the square nodes. These square nodes are higher in each successive tree; the 'After' one being beached at the top. The $\Uparrow$s in the square nodes indicate the direction of rippling-out.

Figure 2: Rippling-Out $(s(x) + y) + z$

2.2 Kinds of Termination

The basic version of rippling-out outlined in this section applies only to induction conclusions that are equations or equivalences. Wave-fronts are rippled-out until they dominate the left- and right-hand sides of these induction conclusions.

The movement of wave-fronts can terminate in three ways.

- If a wave-front is moved to dominate the left- or right-hand term of the induction conclusion then we say it is *beached*. No further rippling-out of this wave-front is then possible or necessary. Two examples can be found in equation (9), above.
- If a wave-rule is applied in which ζ is empty then there is no new wave-front. We say it *peters out*. Two examples can be found in equation (21), §3, below.
- If a wave-front is not beached, but no wave-rule applies to it, then we say the wave-front is *blocked*. An example can be found in equation (25), §4, below.

If all wave-fronts in a term are beached or peter out then then we say it is *fully rippled*. Otherwise, we say the rippling-out is *blocked*.

When no further rippling-out is possible then the induction hypothesis can often be used as a rewrite rule to produce an equation between two identical terms. Following Boyer and Moore, we call this tactic *fertilize*. The induction hypothesis may be used either way round. In our example we can use it left to right on the left hand side of the induction conclusion, (9), to produce the equation:

$$\boxed{s(\underline{(x+y)+z})} \quad = \quad \boxed{s(\underline{(x+y)+z})}$$

which is readily proved, so completing the step case of the inductive proof.

2.3 Search Control

Both rippling-out and fertilization are subject to careful control by our overall inductive proof plan.

- Rippling-out is applied to the induction conclusion immediately after the application of induction. The precondition of rippling-out ensures that it is applied only to move wave-fronts outwards. This greatly restricts the search space of rippling-out. It also ensures the termination of rippling-out, since the wave-fronts can only be moved outwards, through an unchanging skeleton, a finite number of times. This remains true even if the same equation is used as a wave-rule in different left/right orientations, *cf.* rules (7) and (8) in table 1.
- Fertilization is applied to the induction conclusion when no further rippling-out is possible. It uses the induction hypothesis as a rewrite rule either left to right to the left-hand side of the induction conclusion or right to left to its right-hand side. If the wave-fronts on both sides of the induction conclusion are fully rippled then either one of these is sufficient. If only one side is fully rippled then this side is rewritten. If neither side is fully rippled then fertilization is not possible.

2.4 Merging and Splitting

Note that the general wave-rule format allows for wave-fronts to be compound terms. This opens up the possibility that compound wave-fronts might be split into several sub-wave-fronts and each of these rippled-out separately. For instance, suppose the following compound wave-front appears in the induction conclusion:

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

as it stands this wave-front is blocked as none of the wave-rules we have considered so far meets the preconditions of rippling-out. However, if we split the wave front into two:

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

then wave-rule (4) will apply twice to produce first:

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

and then:

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

Now once again rippling-out is blocked because no wave-rule meets its preconditions. This time the two adjacent wave-fronts must be merged into one:

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

so that wave-rule (6) will apply and produce:

$$\ldots even(x + y) \ldots$$

Splitting a compound wave-front and rippling each sub-wave-front separately can result in nested non-adjacent wave-fronts, *e.g.*

$$\ldots \boxed{\neg \underline{even(\boxed{s(\underline{x})} + y)}} \ldots$$

Note that the inner wave-front must always be in the wave-hole of the outer one. We say that the inner wave-front is *beneath* the outer one.

In *CIAM* this splitting and merging of wave-fronts is done on demand. First rippling-out is attempted with the current wave-front annotation. When this has been completed (either successfully or unsuccessfully) all possible splittings and mergings are made and rippling-out is tried with each of these in turn.

An alternative mechanism would be to hold all wave-fronts in a normal form, *e.g.* maximally split or maximally merged. The maximally split normal form is particularly attractive as it allows us to continue to use regular matching to ensure wave-front agreement. If the maximally merged normal form is used then the matcher must be modified to dynamically split wave-fronts as required. The INKA system records wave-fronts by annotating each symbol as either in the wave-front or in the skeleton, [Hutter 90]. This solution has some similarity to the maximally split normal form.

2.5 Branching Rates

In practice, the branching rate of rippling-out is very low. The rippling-out of one wave front is independent of the rippling-out of the others, so that they can be dealt with in a fixed order and redundant branching can be avoided[5]. Genuine branching happens only when more than one wave-rule applies to the same wave front. But this is rare.

To see why this is so consider how it might happen. Suppose the wave-front occurs in the context $\ldots \eta(\boxed{\xi(\underline{\alpha})}) \ldots$ In the simplest case there might be two wave-rules which both ripple ξ out through η, *i.e.*

$$\begin{array}{lcl} \eta(\boxed{\xi(\underline{\mu})}) & \Rightarrow & \boxed{\zeta_1(\underline{\eta(\mu)})} \\ \eta(\boxed{\xi(\underline{\mu})}) & \Rightarrow & \boxed{\zeta_2(\underline{\eta(\mu)})} \end{array}$$

Suppose, without loss of generality, that these rules are both based on equations. Since the left hand sides of the rules are equal, the right hand sides are also equal, *i.e.*

$$\zeta_1(\eta(\mu)) \quad = \quad \zeta_2(\eta(\mu))$$

So ζ_1 and ζ_2 are equal over the range of the function η. If we keep wave-fronts on the right hand sides of wave-rules in a simple normal form then in most cases ζ_1 and ζ_2 will be identical. Thus it is rare to have distinct wave-rules with identical left hand sides, as required for this simple form of branching.

The general wave-rule format allows for both η and ξ to be compound. In these cases branching is possible. However, the alternative branches will often lead to similar proofs. Consider, for

[5] This is not implemented in the current version of *CIAM*.

instance, the situation where $\xi(\mu)$ has the compound form $\xi'(\xi''(\mu))$ and we have the alternative wave-rules:

$$\begin{aligned}\eta(\boxed{\xi'(\xi''(\underline{\mu}))}) &\Rightarrow \boxed{\zeta_1(\underline{\eta(\mu)})}\\ \eta(\boxed{\xi'(\underline{\nu})}) &\Rightarrow \boxed{\zeta_2(\underline{\eta(\nu)})}\end{aligned}$$

leading to two branches in the rippling-out search space. Note that if the second branch is not to immediately become blocked, there must also be a wave-rule of the form:

$$\eta(\boxed{\xi''(\underline{\mu})}) \Rightarrow \boxed{\zeta_3(\underline{\eta(\mu)})}$$

So, by a similar calculation as in the simple case:

$$\zeta_1(\eta(\mu)) = \zeta_2(\zeta_3(\eta(\mu)))$$

So that the alternative routes will lead to similar proofs. In this case it is not usually possible to remove the duplication of wave-rules by normalising the right hand side wave-fronts.

As an example of this kind of branching consider the pair of wave-rules:

$$even(\boxed{s(s(\underline{U}))}) \Rightarrow even(U) \tag{10}$$

$$even(\boxed{s(\underline{U})}) \Rightarrow \boxed{\neg \underline{even(U)}} \tag{11}$$

where ζ_1 is empty and both ζ_2 and ζ_3 are $\neg$. Note that neither ζ_2 nor $zeta_3$ can be normalised on their own, but in combination they normalise to ζ_1, as expected.

The extensions to ripple-out described below will introduce new possibilities for branching. Nevertheless, this will remain a relatively rare phenomenon, so that, in practice, the search space of rippling is very small.

3 Multi-Wave Rules and Strong Fertilization

Our first extension to rippling-out is to allow wave-rules which ripple out more than one wave-front at once. We call these *multi-wave-rules*. They are highly desirable if one side of the induction conclusion contains more than one wave-front — as it will do if it contains more than one occurrence of the induction variable. They allow rippling to transform several wave-fronts into one and, hence, continue rippling beyond the point at which the old version of rippling-out would have become blocked. They also allow us to strengthen rippling-out and fertilization and, thereby, apply them to induction conclusions that are not equations or equivalences.

3.1 General Wave Rule Format

While we are making this extension we will also extend wave-rules to cope with wave-fronts containing more than one wave-hole and to the introduction of multiple skeletons. This leads to the generalisation of wave rule format (3) to the following:

$$\begin{aligned}&\eta(\boxed{\xi_1(\underline{\mu_1^1},\ldots,\underline{\mu_1^{p_1}})},\ldots,\boxed{\xi_n(\underline{\mu_n^1},\ldots,\underline{\mu_n^{p_n}})})\\ &\Rightarrow \boxed{\zeta(\underline{\eta(\varpi_1^1,\ldots,\varpi_n^1)},\ldots,\underline{\eta(\varpi_1^k,\ldots,\varpi_n^k)})}\end{aligned} \tag{12}$$

$$\boxed{s(\underline{U})} = \boxed{s(\underline{V})} \Rightarrow U = V \tag{13}$$

$$\boxed{s(\underline{U})} \geq \boxed{s(\underline{V})} \Rightarrow U \geq V \tag{14}$$

$$\boxed{\underline{U_1^1} + \underline{U_1^2}} = \boxed{\underline{U_2^1} + \underline{U_2^2}} \Rightarrow \boxed{\underline{U_1^1 = U_2^1} \wedge \underline{U_1^2 = U_2^2}} \tag{15}$$

$$\boxed{max(\underline{U_1^1}, \underline{U_1^2})} \geq \boxed{min(\underline{U_2^1}, \underline{U_2^2})} \Rightarrow \boxed{\underline{U_1^1 \geq U_2^1} \wedge \underline{U_1^2 \geq U_2^2}} \tag{16}$$

$$\boxed{(\underline{U} + \underline{V})} \times W \Rightarrow \boxed{\underline{U \times W} + \underline{V \times W}} \tag{17}$$

$$max\_ht(\boxed{tree(\underline{U}, \underline{V})}) \Rightarrow \boxed{s(max(\underline{max\_ht(U)}, \underline{max\_ht(V)}))} \tag{18}$$

$$min\_ht(\boxed{tree(\underline{U}, \underline{V})}) \Rightarrow \boxed{s(min(\underline{min\_ht(U)}, \underline{min\_ht(V)}))} \tag{19}$$

$$binom(\boxed{s(\underline{U})}, \boxed{s(\underline{V})}) \Rightarrow binom(U, \boxed{s(\underline{V})}) + binom(U, V) \tag{20}$$

Note the occurrence of multiple μs, multiple ηs and even multiple ξs in these rules. The first five rules are further examples of wave-rules not formed from recursive definitions. Note that rules (15) and (16) are sound because the orientation of implication is right to left. Note also that in rule (20) the wave-fronts are only partially removed from one of the new skeletons.

Table 2: Example Multi-Wave Rules

where each ϖ_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, μ_i^l. For each j, at least one ϖ_i^j must be a wave-hole. As before, η, the ξ_is and ζ are terms with distinguished arguments. ζ may be empty, but the ξ_is and η must not be. The ξ_is are the old wave-fronts and ζ is the new wave-front. Note that the application of a multi-wave-rule with k ηs in the right hand side will replace one skeleton with k skeletons in the induction conclusion. Some example multi-wave-rules are given in table 2.

The advantages of multi-wave-rules of format (12) with simple wave-rules of format (3) are as follows.

- The generalisation of the single ξ in (3) to the multiple ξ_is in (12) enables the simultaneous rippling-out of multiple wave-fronts. However, not all these ξ_is have to ripple-out — *cf.* rule (20).
- The generalisation of the single μ to the multiple μ_is enables the rippling-out of wave-fronts containing more than one wave-hole. However, we may sometimes want to treat some of these holes as part of the wave-front — see §3.4 below.
- The generalisation of the single η argument of ζ to the multiple η arguments enables several different induction hypothesis to fertilize the induction conclusion.

These last two extensions go together; multiple μs tend to create multiple ηs, each one making a different selection of μs for its arguments.

Following this generalisation of the wave-rule format it is necessary to revise the last item of the preconditions of wave-rule application (definition 1).

Definition 2 (Preconditions of Applying Multi-Wave-Rules) *The preconditions of applying a wave-rule to rewrite a sub-expression of the induction conclusion are that:*

- *the left hand side of the rule matches the sub-expression;*
- *the sub-expression contains at least one wave-front; and*
- each *wave-front in the sub-expression is matched with* an *old wave-front in the rule.*

3.2 Strong Fertilization

We can now revisit and reconsider the associativity of + example from the last section §2. Instead of applying fertilization to the induction conclusion (9), we can use multi-wave-rule (13) to continue the rippling-out and infer:

$$x + (y + z) \quad = \quad (x + y) + z \tag{21}$$

which is identical to the induction hypothesis, (1), *i.e.* the wave-fronts have petered out. The proof thus terminates by direct appeal to the induction hypothesis. This direct use of the induction hypothesis constitutes a new form of fertilization which we will call *strong fertilization*. The form of fertilization described above in §2 we will rename *weak fertilization*. Since strong fertilization applies to induction conclusions of any form (and not just to equalities and equivalences) it is more generally applicable, and its application will normally be the purpose of rippling-out. However, weak fertilization must still be retained to cope with equational or equivalential induction conclusions that get blocked on one side.

3.3 A Worked Example

To illustrate the use of multi-wave-rules to prove non-equational theorems with the aid of multiple induction variables and multiple induction hypotheses, consider the following example.

$$\forall T{:}tree.\ \mathit{max\_ht}(T) \quad \geq \quad \mathit{min\_ht}(T)$$

where $\mathit{max\_ht}(T)$ is the length of the longest path in a binary tree, T, and $\mathit{min\_ht}(T)$ is the length of the shortest. Trees will be built from the constructor functions $leaf(T)$ and $tree(T_1, T_2)$. This example has been designed to exhibit all the features of multi-wave rippling, and as a consequence is rather artificial.

Our proof will use the standard, structural induction rule on binary trees, *i.e.* the theorem is proved for the empty tree and then assumed for the left and right sub-trees and proved for the whole tree. Thus, in the step case we will have the following two induction hypotheses:

$$\mathit{max\_ht}(l) \geq \mathit{min\_ht}(l) \qquad \mathit{max\_ht}(r) \geq \mathit{min\_ht}(r)$$

and must prove the induction conclusion:

$$\mathit{max\_ht}(\boxed{tree(\underline{l}, \underline{r})}) \quad \geq \quad \mathit{min\_ht}(\boxed{tree(\underline{l}, \underline{r})}) \tag{22}$$

in which both l and r are induction variables.

To ripple this induction conclusion out we will use the multi-wave-rules from table 2.

$$\begin{array}{c}
\boxed{s(max(\underline{\mathit{max_ht}(l)}, \underline{\mathit{max_ht}(r)}))} \geq \boxed{s(min(\underline{\mathit{min_ht}(l)}, \underline{\mathit{min_ht}(r)}))} \\
\boxed{max(\underline{\mathit{max_ht}(l)}, \underline{\mathit{max_ht}(r)})} \geq \boxed{min(\underline{\mathit{min_ht}(l)}, \underline{\mathit{min_ht}(r)})} \\
\boxed{\underline{\mathit{max_ht}(l) \geq \mathit{min_ht}(l)} \wedge \underline{\mathit{max_ht}(r) \geq \mathit{min_ht}(r)}}
\end{array} \tag{23}$$

to which strong fertilization applies, finishing the proof of the step case.

Note the following points from this proof.

- Because the constructor function *tree* contains two induction variables, l and r, it forms a wave-front with two wave-holes. This situation is inherited by subsequent versions of the induction conclusion, with *max*, *min* and $\wedge$ also having two hole wave-fronts.
- The theorem is not an equation or equivalence, so strong fertilization must be used to complete the rippling-out.
- The skeleton is duplicated by the first wave-rule. This eventually leads to two copies of the induction hypothesis, which are fertilized independently.
- The two wave-fronts in line (23) are compound terms. They are each split into two wave-fronts and rippled-out separately.
- The last wave-front is a logical connective. Rippling-out can proceed through predicates and connectives as well as functions.

3.4 Rule Weakening

It is sometimes desirable to use multi-wave-rules at less than their full strength, *i.e.* to treat some of the wave-holes as part of the wave-front. We can see two examples of this in the following proof of the associativity of $\times$,

$$\forall X{:}nat.\forall Y{:}nat.\forall Z{:}nat.\ X \times (Y \times Z) = (X \times Y) \times Z$$

The rippling of the step case uses the rules (5) from table 1 and (15) and (17) from table 2.

$$\begin{array}{rcl}
\boxed{s(\underline{x})} \times (y \times z) & = & (\boxed{s(\underline{x})} \times y) \times z \\
\boxed{\underline{x \times (y \times z)} + y \times z} & = & (\boxed{(\underline{x \times y}) + y}) \times z \\
\boxed{\underline{x \times (y \times z)} + y \times z} & = & \boxed{\underline{(x \times y) \times z} + y \times z} \\
\multicolumn{3}{c}{\boxed{\underline{x \times (y \times z) = (x \times y) \times z} \wedge y \times z = y \times z}}
\end{array}$$

Note that wave-rules (15) and (17) are used as if their wave-fronts were:

$$\begin{array}{rcl}
\boxed{\underline{U_1^1} + U_1^2} = \boxed{\underline{U_2^1} + U_2^2} & \Rightarrow & \boxed{\underline{U_1^1 = U_2^1} \wedge U_1^2 = U_2^2} \\
\boxed{(\underline{U} + V)} \times W & \Rightarrow & \boxed{\underline{U \times W} + V \times W}
\end{array}$$

i.e. with the second wave-hole merged with the wave-front on both left and right hand sides of each rule. This is called wave-rule *weakening*. We currently implement weakening at compile time, *i.e.* when it is storing a new wave-rule CL^AM also calculates all its possible weakenings and stores these as well. We are also exploring the implementation of weakening by modification of the matcher, *i.e.* by allowing it to ignore wave-fronts in the wave-rule.

The weakening of wave-rules can result in the capture of an inner wave-front nested beneath an outer one. The inner one goes from being in the wave-hole of the outer one to being within its wave-front proper. We say the inner wave-front is *inside* the outer one. Contrast this situation with the nesting *beneath* described in §2.4. Inside wave-fronts are merged with the outer wave-front — all their wave-front annotation being dropped.

3.5 Simplification of Wave Fronts

Some of the multi-wave-rules call for identical wave-fronts at different positions with the induction conclusion, *e.g.* rule (13) in table 2 calls for each side of an equation to start with $\boxed{s(\underline{\ldots})}$. As a result of earlier rewriting it sometimes happens that an induction conclusion has equal, but not identical, wave-fronts at these positions. An example occurs during the proof of the distributivity law of $\times$ over $+$.

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

which prevents the application of rule (15).

To reduce the chances of this problem blocking a ripple, *CIAM* regularly simplifies wave-fronts by applying a simple normalisation tactic to them. The details of this normalisation are beyond the scope of this paper, but it consists of symbolic evaluation with recursive definitions and other complexity reducing rewrite rules. Note that this normalisation process must be applied to the wave-fronts without perturbing their wave-holes. These wave-holes form part of the skeleton and any perturbation will prevent the application of strong fertilization, unless it is applied uniformly to all skeletons and to the induction hypothesis.

4 Weak Fertilization and Rippling-In

4.1 Example of a Blocked Ripple

Weak fertilization is useful when a rippling process gets blocked on one side of an equation or equivalence and strong fertilization is not possible. Consider, for instance, the theorem $\forall X : nat.\ half(X+X) = X$, where *half* is the half integer function defined by:

$$\begin{aligned} half(0) &= 0 \\ half(s(0)) &= 0 \\ half(s(s(U))) &= s(half(U)) \end{aligned}$$

This definition provides the wave-rule:

$$half(\boxed{s(s(\underline{U}))}) \quad \Rightarrow \quad \boxed{s(\underline{half(U)})} \tag{24}$$

Suppose we try to prove the theorem with $s(x)$ induction, so the induction hypothesis is:

$$half(x+x) \quad = \quad x$$

and the induction conclusion is:

$$half(\boxed{s(\underline{x})} + \boxed{s(\underline{x})}) \quad = \quad \boxed{s(\underline{x})}$$

The rule (4) applies to the left hand side of the induction conclusion, which ripples the left-most wave-front once. The induction conclusion becomes:

$$half(\boxed{s(\underline{x + \boxed{s(\underline{x})}})}) \quad = \quad \boxed{s(\underline{x})} \tag{25}$$

At this point no further wave-rules apply and the ripple is blocked. Note that the wave-rule required to unblock this ripple is:

$$U + \boxed{s(\underline{V})} \quad \Rightarrow \boxed{s(\underline{U+V})} \tag{26}$$

a commuted version of (4). For the sake of this example we are assuming that this rule is not available.

Fortunately, the right hand side of (25) is trivially fully rippled, so we can apply weak fertilization to get:

$$half(s(x + s(x))) \quad = \quad \boxed{s(\underline{half(x + x)})}^{\downarrow}$$

After strong fertilization, wave-fronts are removed since they have fully completed their job. After weak fertilization they may still have a role to perform on the unblocked side of the equation, *i.e.* the right hand side in this example, so we have left them in place on this side. Their residual role is to be *rippled-in* by applying reversible wave-rules right to left. We have indicated this by annotating the wave-front with the direction in which it should be rippled: inwards[6]. Wave-fronts that should be rippled outwards will be annotated by an upwards arrow. Rippling-in is our second extension to rippling-out.

4.2 Why Rippling-In is a Good Idea

To see why rippling-in is a good idea, consider the following generic induction step case. The induction hypothesis has the following schematic form:

$$+++++++ \quad = \quad *******$$

and the induction conclusion is:

$$+++\boxed{?\underline{+}?}^{\uparrow}+++ \quad = \quad ***\boxed{?\underline{*}?}^{\uparrow}*** \tag{27}$$

Suppose that the right hand side of the induction conclusion can be completely rippled but the left hand side becomes blocked.

$$++\boxed{?\underline{+++}?}^{\uparrow}++ \quad = \quad \boxed{?\underline{*******}?}^{\uparrow} \tag{28}$$

Fertilizing the right hand side gives:

$$++?+++?++ \quad = \quad \boxed{?\underline{+++++++}?}^{\downarrow} \tag{29}$$

Note that the $+++++++$ skeletons on either side of the $=$ are identical, but that the ?? wave-fronts may differ. The intuition is that since the two sides of equation (29) are so similar it will be easier to prove than the original equation, (27), in which the two sides may be completely different.

If the wave-front surrounding the right hand side can be rippled-in towards the induction variable then the induction conclusion can be transformed to the form:

$$++?+++?++ \quad = \quad +?+++++?+$$

where the wave-front has been dropped since no further rippling-in is possible. At this point the outermost function symbols on the left and right are bound to be identical since they each belong to identical $+++++++$ skeletons. Thus, they may be cancelled, reducing the induction conclusion to:

$$+?+++?+ \quad = \quad ?+++++? \tag{30}$$

[6] Or downwards in the graphical representation.

The aim of rippling-in is to enable this cancellation to take place. In *CIAM*, rippling-in and cancellation are interleaved until no further cancellation is possible. The intuition is that equation (30) is even easier to prove than equation (29) because it is syntactically simpler, while retaining the property that the two sides of the equation are very similar.

Another way to look at rippling-in is as a technique for bi-directional rippling-out. Ideally, both sides of the equation would ripple-out fully. The left hand side would then have the form: $\boxed{?\underline{+ + + + + + +}?}^{\uparrow}$, which is the same form as the right hand side after fertilization, *cf.* equation (29). Unfortunately, the left hand side gets blocked at: $+ + \boxed{?\underline{+ + +}?}^{\uparrow} + +$. Rippling-in starts from $\boxed{?\underline{+ + + + + + +}?}^{\downarrow}$ and performs what would have been the last few rippling-outs in reverse to get: $+\boxed{?\underline{+ + + + +}?}^{\downarrow}+$. Note that the final subgoal, equation (30), could be made into a wave-rule. In fact, if this wave-rule were available the rippling-out would not have been blocked, since it is just what is needed to continue rippling the left hand side of equation (28). It is the missing wave-rule — or, more accurately, an instance of one such missing wave-rule.

4.3 Example of Rippling-In and Lemma Discovery

To illustrate rippling-in we return to our half integer example just after weak fertilization.

$$half(s(x + s(x))) \quad = \quad \boxed{s(\underline{half(x + x)})}^{\downarrow}$$

Rippling-in can be performed by applying wave-rule (24) right to left. This gives:

$$half(s(x + s(x))) \quad = \quad half(\boxed{s(s(\underline{x + x}))}^{\downarrow})$$

after which further rippling-in would not assist cancellation. The wave-fronts are now dropped and the outermost function symbols cancelled to give:

$$x + s(x) \quad = \quad s(x + x) \tag{31}$$

This remaining subgoal is proved by generalising the second occurrence of x on each side of the equation to y and then using $s(x)$ induction. The details of this generalisation step are beyond the scope of this paper.

Note that the subgoal (31) that remains, after weak fertilization and rippling-in, is an instance of the missing wave-rule (26). This is not a coincidence. In fact, we can see from the above analysis that if a ripple is blocked for the lack of a single wave-rule, and if weak fertilization and rippling-in succeed, then the subgoal they leave is bound to be an instance of the missing rule. In proving this subgoal, *CIAM* often starts by generalising it into the missing wave-rule. After the proof it is added to *CIAM*'s stock of wave-rules for future use, so ripples will not block for the lack of this rule in future. We will refer to this phenomenon as the 'in-line' proof of wave-rules. It shows the power of our proof plans technique to conjecture and prove lemmas needed to complete a proof.

4.4 Controlling Rippling-In

Controlling rippling-in raises a couple of complications that are not already present in rippling-out.

- Firstly, there may be a choice between different inwards directions. For instance, an expression $\boxed{\eta(\underline{\mu(\alpha, \beta)})}^{\downarrow}$ may be rippled-in to either $\mu(\boxed{\eta'(\underline{\alpha})}^{\downarrow}, \beta)$ or $\mu(\alpha, \boxed{\eta''(\underline{\beta})}^{\downarrow})$. Fortunately, we can be guided by the shape of the other side of the equation. For cancellation to be possible this should have either the form $\mu(\alpha, \ldots)$ or the form $\mu(\ldots, \beta)$, and whichever it is will determine which rippling-in we perform.

- Secondly, we need to know when to stop. Again, we only ripple-in if this will allow cancellation to take place. For instance, if the equation is $\mu(\alpha, \eta''(\beta)) = \boxed{\eta(\underline{\mu(\alpha, \beta)})}^{\downarrow}$ and the only available wave-rule will ripple-in the right hand side to $\mu(\boxed{\eta'(\underline{\alpha})}^{\downarrow}, \beta)$ then rippling-in is stopped.

- Thirdly, not all wave-rules can be legally applied right to left, *e.g.* if they are formed from implications then they can only be used left to right on sub-formulae of positive polarity and right to left on sub-formulae of negative polarity. Similar remarks hold for inequalities. We call such wave-rules *irreversible*. On the other hand, wave-rules formed from equalities and equivalences are always *reversible*. For irreversible wave-rules *CIAM* uses a general notion of polarity to determine in which orientation and to which sub-expressions they can be legally applied. For instance, for wave-rules formed from implications the polarity of a sub-formulae is the parity of the number of implicit or explicit negations within which it is contained, *e.g.* in $\neg p \wedge q \rightarrow r$, p and r are of positive polarity and q is of negative polarity. For wave-rules formed from inequalities the polarity corresponds to the monotonicity of the functions containing the sub-expression.

In §6.3 and 10.2.2 we will meet a variation of rippling-in that uses a different kind of wave-rule.

5 Conditional Wave Rules

Our third extension is to allow conditional wave-rules, *i.e.* wave-rules that are only true under some condition. They have the form:

$$Cond \quad \rightarrow \quad LHS \quad \Rightarrow \quad RHS$$

where $LHS \Rightarrow RHS$ is a wave-rule and $Cond$ is a formula.

If the condition of a rule is provable from the current hypotheses then, clearly, we can use the rule. But even if it is not currently provable we can still use the rule provided we divide the proof into two cases using the condition and its negation. The condition is then trivially provable in the first case. So a major problem to be solved in the use of conditional rules is when to try to prove the condition within the current case (either immediately or later) and when to use the condition to split the current case into two sub-cases.

As a partial solution to this problem, related conditional rules are stored, by *CIAM* , in complementary sets of the form:

$$\begin{array}{lclcl} Cond_1 & \rightarrow & \eta(\boxed{\xi_1^1(\underline{\mu_1^1}, \ldots, \underline{\mu_1^{p_1}})}, \ldots, \boxed{\xi_n^1(\underline{\mu_n^1}, \ldots, \underline{\mu_n^{p_n}})}) & \Rightarrow & RHS_1 \\ \ldots & & \ldots & & \ldots \\ Cond_k & \rightarrow & \eta(\boxed{\xi_1^k(\underline{\mu_1^1}, \ldots, \underline{\mu_1^{p_1}})}, \ldots, \boxed{\xi_n^k(\underline{\mu_n^1}, \ldots, \underline{\mu_n^{p_n}})}) & \Rightarrow & RHS_k \end{array}$$

where RHS_i is either of the form:

$$\boxed{\zeta_i(\underline{\eta(\mu_1^{q_1^i}, \ldots, \mu_n^{q_n^i})}, \ldots, \underline{\eta(\mu_1^{r_1^i}, \ldots, \mu_n^{r_n^i})})}$$

$$El = H \rightarrow El \in (\boxed{H :: \underline{T}}) \Rightarrow true$$

$$El \neq H \rightarrow El \in (\boxed{H :: \underline{T}}) \Rightarrow El \in T$$

$$H \in S \rightarrow (\boxed{H :: \underline{T}}) \cap S \Rightarrow \boxed{H :: (\underline{T \cap S})}$$

$$\neg H \in S \rightarrow (\boxed{H :: \underline{T}}) \cap S \Rightarrow T \cap S$$

These two complementary sets of conditional rules are taken from the definitions of set membership and the intersection of two sets, respectively. Sets are represented as lists, so $H :: T$ is the set formed by adding element H to the set T. Note that in the $\in$ set only the second rule is a conditional wave rule, but that in the $\cap$ set both rules are.

Table 3: A Set of Conditional Wave Rules

$1 \leq q^i_j, r^i_j \leq p_j$ for $1 \leq j \leq n$ and $1 \leq i \leq k$, or is an expression not containing η and where we require that:

$$Cond_1 \vee \ldots \vee Cond_k$$

and where for each $1 \leq i \leq n$ there exists a wave-front, ξ_i, which subsumes the wave-fronts ξ^j_i for all $1 \leq j \leq k$, where the ξ_is are the wave-fronts to be rippled-out in the induction conclusion. A wave-front, ξ, subsumes a wave-front ξ' if ξ consists of nested copies of ξ'. For instance, the wave-front $s(s(\ldots))$ subsumes the wave-front $s(\ldots)$. Note that two consecutive ripples of $s(\ldots)$ will ripple $s(s(\ldots))$ once.

Note that it is not always possible to form a complementary set from conditional wave-rules alone. Some conditional rules in a complementary set will be conditional wave-rules and some will be conditional, non-recursive definitions of η. See, for instance, the complementary set for $\in$ in table 3.

Now that wave-rules can be conditional it is necessary add an additional item to the preconditions of wave-rule application (definition 2)..

Definition 3 (Preconditions of Applying Conditional Wave-Rules) *The preconditions of applying a wave-rule to rewrite a sub-expression of the induction conclusion are that:*

- *the left hand side of the rule matches the sub-expression;*
- *the sub-expression contains at least one wave-front;*
- *each wave-front in the sub-expression is matched with a wave-front in the rule; and*
- *the condition of the wave-rule is provable from the current hypotheses.*

Some examples of conditional rules are given in table 3.
When applying conditional wave-rules *CIAM* proceeds as follows.

- If the condition is a variant of one of the existing hypotheses of the current case then the rule can be applied with no further work.

- Else if the rule is a member of a complementary set then the current case is split into sub-cases using the conditions of that set.

- Otherwise, the condition is set up as an additional sub-goal.

In this way we avoid dividing into sub-cases unless each of the sub-cases is likely to succeed. Since, for each sub-case, there is a conditional rule whose condition is satisfied, then one of the following two situations will obtain.

- If the rule for a sub-case is a conditional wave-rule, then rippling-out can continue.

- If the right hand side of the rule for a sub-case does not contain η then the rule provides a non-recursive definition of η in this case and the proof proceeds by symbolic evaluation (see [Bundy *et al* 91]) rather than induction.

This gives some assurance of success in each sub-case.

This is only a partial solution to the problems of using conditional rules because we might not invest enough effort into proving the condition in the current case before giving up and dividing into cases. Thus we will create an unnecessary case split.

To illustrate the use of conditional wave-rules consider the theorem:

$$\forall A{:}nat.\forall B{:}set(nat).\forall C{:}set(nat).\, A \in B \wedge A \in C \quad \rightarrow \quad A \in B \cap C$$

The induction conclusion of this theorem is:

$$(a \in \boxed{e :: \underline{b}}) \wedge (a \in c) \quad \rightarrow \quad a \in (\boxed{e :: \underline{b}} \cap c)$$

Consider the second wave-front. Assume that the only matching wave-rules are the complementary set for $\cap$ in table 3. We can use this complementary set to suggest a division into two cases, each of which has some assurance of success. After dividing into these two cases and applying the conditional wave-rules we get:

$$e \in c \quad \vdash \quad a \in \boxed{e :: \underline{b}} \wedge a \in c \quad \rightarrow \quad a \in \boxed{e :: \underline{(b \cap c)}}$$

$$\neg e \in c \quad \vdash \quad a \in \boxed{e :: \underline{b}} \wedge a \in c \quad \rightarrow \quad a \in b \cap c$$

The remaining wave-fronts can be rippled-out once each with the wave-rule:

$$X \in \boxed{H :: \underline{T}} \quad \Rightarrow \quad \boxed{X = H \vee \underline{X \in T}}$$

This gives:

$$e \in c \quad \vdash \quad (\boxed{a = e \vee \underline{a \in b}}) \wedge a \in c \rightarrow \boxed{a = e \vee \underline{a \in b \cap c}}$$

$$\neg e \in c \quad \vdash \quad (\boxed{a = e \vee \underline{a \in b}}) \wedge a \in c \rightarrow a \in b \cap c$$

and the wave-fronts introduced by this rule can be beached with various propositional wave-rules, *e.g.* the distributive law of $\wedge$ over $\vee$. The second case, for example, becomes:

$$\neg e \in c \quad \vdash \quad \boxed{(a = e \wedge a \in c \rightarrow a \in b \cap c) \ \wedge \ \underline{(a \in b \wedge a \in c \rightarrow a \in b \cap c)}}$$

After strong fertilization this leaves the residue:

$$\neg e \in c \quad \vdash \quad (a = e \wedge a \in c \rightarrow a \in b \cap c) \quad \wedge \quad true$$

which can be readily proved. The first case is similar.

6 Rippling-Sideways into Sinks

Our fourth extension is to consider a third way in which rippling can successfully terminate. In our examples so far we have transformed universally quantified variables into skolem constants in both induction hypothesis and induction conclusion. However, universally quantified non-induction variables can be transformed into free variables in the induction hypothesis and skolem constants in the induction conclusion. These free variables are not constrained to be matched to their corresponding skolem constants during fertilization — they can be matched to any expressions.

This significantly increases the options open to the theorem prover. Wave-fronts can be *rippled-sideways* to surround the skolem constants, which we will call *sinks*. These sinks absorb the new wave-fronts and the engorged sinks can then be matched to the corresponding free variables during fertilization. In principle, this rippling-sideways can always be done using ordinary wave-rules: first applied forwards then applied backwards. In practice, not all the required wave-rules are likely to be available, and the central part of the rippling must be done by a new kind of wave-rule. We will call these new kinds of wave-rule *transverse wave-rules*, and we will rename the original wave-rules as *longitudinal wave-rules*. This terminology of *sideways*, *transverse* and *longitudinal* is natural in the graphical representation of rippling, *cf.* figure 3.

6.1 The Format of Transverse Wave Rules

A simple transverse wave-rule is a rewrite rule of the form:

$$\eta(\boxed{\xi(\underline{\mu})}^{\uparrow}, \nu) \quad \Rightarrow \quad \eta(\mu, \boxed{\zeta(\underline{\nu})}^{\downarrow})$$

i.e. it moves a wave-front from one argument of the function η to another one. η, ξ and ζ are each a non-empty term with a distinguished argument. If ζ were empty then the rule would be a longitudinal wave-rule, *cf.* format (3) with ζ empty. Note the directional annotations on the wave-fronts. These reflect the fact that the purpose of applying a transverse wave-rule is to reverse the outwards direction of the original wave-front and send it inwards towards a sink.

Just as for longitudinal rules, the simple transverse wave form can be generalised, to give the multi-wave form:

$$\begin{aligned} Cond \quad \rightarrow \quad & \eta(\boxed{\xi_1(\underline{\mu_1})}^{\uparrow}, \ldots, \boxed{\xi_n(\underline{\mu_n})}^{\uparrow}, \nu_1, \ldots, \nu_m) \quad \Rightarrow \\ & \eta(\mu_1, \ldots, \mu_n, \boxed{\zeta_1(\underline{\nu_1})}^{\downarrow}, \ldots, \boxed{\zeta_m(\underline{\nu_m})}^{\downarrow}) \end{aligned}$$

although, we cannot generalise this to wave-fronts with multiple wave-holes or to multiple ηs. A longitudinal wave-front would be required on the left hand side to contain multiple ηs. Without multiple ηs we require a 1-1 mapping between the μs and νs on the left and right hand sides, and this prevents multiple wave holes. However, these restrictions are lifted when we consider hybrid transverse and longitudinal wave-rules in §8.

Note that the format is reversible, *i.e.* if the left hand side and right hand side are exchanged and the directional annotations of the wave fronts are inverted then we get another transverse wave rule. Thus each transverse wave rule has a dual obtained by this reversal operation. If a transverse wave-rule is irreversible, then if it can be applied to sub-expressions of positive polarity then its reverse dual can be applied to sub-expressions of negative polarity, and *vice versa*. The use of both a transverse wave-rule and its reverse dual does not lead to looping because of the sense of direction imposed by the preconditions of rippling-sideways (see §6.3).

The preconditions for applying both conditional transverse wave-rules are the same as those for conditional longitudinal wave-rules (definition 3) except that we must now check for the directional annotation on the wave-front.

$$
\begin{aligned}
(\boxed{\underline{U} <> V}^{\uparrow}) <> W &\Rightarrow U <> (\boxed{V <> \underline{W}}^{\downarrow}) \\
U <> (\boxed{V <> \underline{W}}^{\uparrow}) &\Rightarrow (\boxed{\underline{U} <> V}^{\downarrow}) <> W \\
qrev(\boxed{Hd :: \underline{Tl}}^{\uparrow}, L) &\Rightarrow qrev(Tl, \boxed{Hd :: \underline{L}}^{\downarrow}) \\
qrev(Tl, \boxed{Hd :: \underline{L}}^{\uparrow}) &\Rightarrow qrev(\boxed{Hd :: \underline{Tl}}^{\downarrow}, L)
\end{aligned} \qquad (32)
$$

Two pairs of transverse wave-rules are shown. The members of each pair are reverse duals. The first pair is formed from the associative law of list append, where $<>$ *is infix append. This shows that associativity laws can be interpreted as transverse wave-rules as well as longitudinal wave-rules (*cf. *rules (7) and (8) in table 1). The second pair is formed from the step case of the tail-recursive definition of list reversal. Tail-recursive definitions which use accumulators are a major source of transverse wave-rules.*

Table 4: Transverse Wave Rules

Definition 4 (Revised Preconditions of Applying Wave-Rules) *The preconditions of applying a wave-rule to rewrite a sub-expression of the induction conclusion are that:*

- *the left hand side of the rule matches the sub-expression;*
- *the sub-expression contains at least one wave-front;*
- *each wave-front in the sub-expression is matched with a wave-front* of the same kind *in the rule; and*
- *the condition of the wave-rule is provable from the current hypotheses.*

Some examples of transverse wave-rules are given in table 4.

6.2 A Worked Example

To illustrate rippling-sideways, consider the following example. Let unary rev be the naive list reversal function and binary $qrev$ be the tail recursive list reversal function, so that the following rewrite rules are available.

$$
\begin{aligned}
rev(nil) &\Rightarrow nil \\
rev(\boxed{Hd :: \underline{Tl}}^{\uparrow}) &\Rightarrow \boxed{\underline{rev(Tl)} <> (Hd :: nil)}^{\uparrow} && (33) \\
qrev(nil, L) &\Rightarrow L \\
qrev(\boxed{Hd :: \underline{Tl}}^{\uparrow}, L) &\Rightarrow qrev(Tl, \boxed{Hd :: \underline{L}}^{\downarrow}) && (34)
\end{aligned}
$$

The second argument of $qrev$ is an accumulator. Note that (33) is a longitudinal wave-rule and (34) is a transverse wave-rule.

Consider the following theorem connecting these two list reversal functions.

$$\forall L{:}list(nat).\forall M{:}list(nat).\ rev(L) <> M \quad = \quad qrev(L, M)$$

To prove this theorem let L be the induction variable. The other universally quantified variable, M, will become a skolem constant in the induction conclusion (represented by m) and a free variable in the induction hypothesis (represented by M). Note that when the induction hypothesis is used to fertilize the induction conclusion it is not necessary for M to be instantiated to m. M can take any value, provided all occurrences of M are instantiated to identical values. This opens the possibility that wave-fronts can be rippled around the sink m so that fertilization will absorb them into the instantiation of M. We will annotate the sink, m, with $\lfloor m \rfloor$, in order to signify that it is a target for wave-fronts.

So, by the discussion above, the induction hypothesis is:

$$rev(l) <> M \quad = \quad qrev(l, M) \tag{35}$$

where M is a free variable, and the induction conclusion is:

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

To make the induction conclusion match the induction hypothesis we will ripple the two wave-fronts sideways so that each surrounds an $\lfloor m \rfloor$. Applying rule (34) does this to the right hand side wave-front in one step.

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

We draw the sink annotation outside the wave-front to show that it has absorbed the wave-front.

Rippling the left hand wave-front is more difficult. First, (33) is applied to ripple the wave-front out.

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

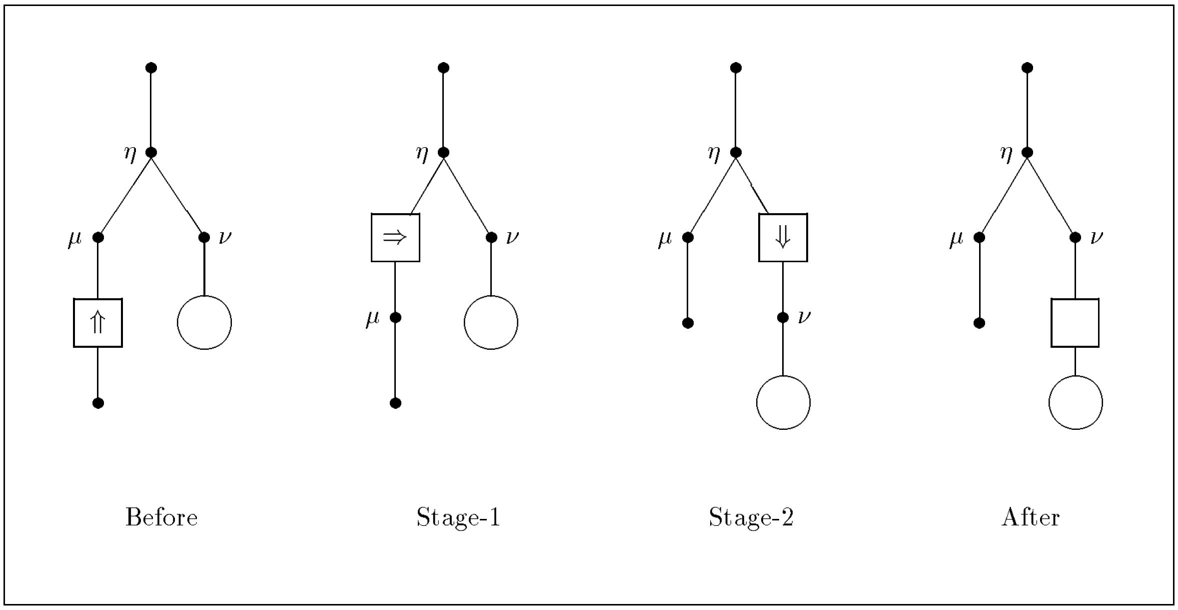
Second, wave-rule (32), the associativity of $<>$, is used to ripple the wave-front sideways to the left hand $\lfloor m \rfloor$.

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

Third, normalisation is applied to simplify the wave-fronts in the sinks and make them identical. This rewrites $(h :: nil) <> m$ to $h :: m$ on the left hand side. This permits strong fertilization to match the induction hypothesis, (35), to the induction conclusion, which instantiates M to $h :: m$, and completes the step case.

6.3 Control of Rippling-Sideways

Rippling-sideways is controlled as follows. Initially, all wave-fronts are annotated as outward directed waves. When a transverse wave-rule is applied the new wave-fronts are all annotated as inwards directed (*cf.* the general format of transverse wave-rules above). The wave-holes of each of these new inwards wave-fronts must contain a sink, otherwise the transverse rule application is not permitted. Only rippling-in is permitted to move these inward directed wave-fronts. The version of rippling-in used at this stage is controlled differently to the version described in §4. Instead of wave-fronts being directed in order to enable cancellation, they are directed towards the sinks in their wave-holes. Rippling-in terminates when all inward directed wave-fronts are absorbed into these sinks. The effect of this control is that wave-fronts are first rippled-out, then -sideways and then -in towards sinks. A diagrammatic representation of a simple case of this process is given in figure 3. Finally, the engorged sinks are simplified to facilitate fertilization. For fertilization to succeed each occurrence of a universally quantified variable in the induction hypothesis must match against an identical sink. Simplifying wave-fronts can turn equal wave-fronts into identical ones, as in the third stage of the example above.



The four expression trees show the expression before, during and after rippling-sideways. Sinks are indicated by circle nodes and wave-fronts by square nodes, as in figure 2. η is the least upper bound of the wave-front and sink. μ and ν are the immediate daughters of η in the direction of the wave-front and sink, respectively. In Stage-1 *the wave-front has been rippled-out until it is between the μ and η nodes. In* Stage-2 *it has been rippled-sideways so that it is between the ν and η nodes. In* After *it has been rippled-in to the sink.*

Figure 3: Stages of Transverse Rippling

The strong direction imposed on rippling by the directional annotation of wave-fronts prevents looping. Once a wave-front has started to move towards a sink it cannot move backwards just because there is a rule available to so move it. This explains why the presence of both a transverse wave-rule and its reverse dual does not cause a loop. This is another example of how the use of wave annotations and the preconditions of rippling tame a potentially explosive set of rewrite rules.

However, the presence of both longitudinal and transverse wave rules does introduce a new cause of branching into rippling. It is now possible to have a longitudinal and a transverse wave-rule Both meet the preconditions of rippling, so that there is a genuine choice in the search space. Moreover, these choices do not lead to slight variants of the same proof, as in our previous examples of such choice. If they do both lead to proofs they will be significantly different ones. An example of a pair of alternative wave-rules is:

$$\boxed{s(\underline{U})}^{\uparrow} + V \quad \Rightarrow \quad \boxed{s(\underline{U+V})}^{\uparrow}$$

$$\boxed{s(\underline{U})}^{\uparrow} + V \;\Rightarrow\; U + \boxed{s(\underline{V})}^{\downarrow}$$

In §4 we described the use of longitudinal wave-rules applied backwards after weak fertilization to enable cancellation of identical functions from either side of an equation or similar formula. We called this rippling-in. There is an equivalent process for the backwards[7] application of transverse wave-rules. Wave-fronts are rippled sideways *out* of sinks to enable cancellation. The control of this process is almost identical to that described in §4.4. An illustration of this process can be found in §10.2.2.

6.4 Tail-Recursion and Transverse Wave-Rules

We noted in the caption to table 4 that the step cases of tail-recursive functions which use accumulators are a good source of transverse wave-rules. This is more than just a coincidence. The movement of a wave-front from recursion variable to accumulator is what defines the use of an accumulator to achieve tail-recursive behaviour. This definition then provides a wave-rule that moves a wave-front from induction variable to sink. Each occurrence of an accumulator provides a sink. Moreover, the verification, synthesis or transformation of functions using accumulators usually requires rippling-sideways during any inductive proofs. Otherwise, the function's recursive definition cannot be used in the proof. The *qrev* example above illustrates this.

7 Rippling-Across and Destructor Inductions

Our fifth extension is to adapt rippling to proofs by destructor induction. In all the proofs considered so far the induction rules have been in the constructor style, that is the induction hypothesis has been of the form $\phi(\chi)$ and the induction conclusion has been of the form $\phi(c(X))$, for some constructor function, c. This neglects proofs in the destructor style of induction, that is where the induction conclusion has the form $\phi(\chi)$ and there are one or more induction hypotheses, each of the form $\phi(d(\chi))$ for some destructor function, d. Many functions can only be defined using a destructor style of recursion, *e.g.* quicksort, and for these the destructor style of induction is usually more natural.

One approach to rippling in destructor induction proofs is to ripple forwards on the induction hypotheses rather than rippling backwards on the induction conclusion. We have attempted this approach. It is described in [Bundy *et al* 90b]. The difficulty is in using a rippled induction hypothesis, $\boxed{\zeta(\underline{\phi(\chi)})}$, to prove the induction conclusion, $\phi(\chi)$. Unless ζ is empty, no fertilization is possible, so the overall inductive proof plan breaks down.

In this paper we take another approach — we ripple the wave-front across from the hypotheses to the conclusion, and then ripple the induction conclusion as before. This rippling-across consists of two stages:

Creation: A new kind of wave-rule is applied to the induction conclusion which creates a wave-front and some anti-wave-fronts, where neither kind of wave-front existed before. We call this new kind of rule a *creational rule*.

Neutralisation: The new anti-wave-fronts in the induction conclusion neutralise the old wave-fronts in the induction hypotheses.

The general format of creational rules is:

[7] Except note that an inverted transverse wave-rule is still a transverse wave-rule, so this is a another kind of ripple-sideways.

$$
\begin{array}{rcrcl}
U \neq 0 & \rightarrow & U & \Rightarrow & \boxed{s(\underline{\boxed{p(\underline{U})}^{-}})}^{+} \\
U \neq 0 & \rightarrow & U+V & \Rightarrow & \boxed{s(\underline{\boxed{p(\underline{U}}^{-}+V)})}^{+} \\
x \neq nil & \rightarrow & conscells(U) & \Rightarrow & \boxed{s(\underline{conscells(\boxed{car(\underline{U})}^{-})}+\underline{conscells(\boxed{cdr(\underline{U})}^{-})})}^{+} \\
x \neq nil & \rightarrow & len(U) & \Rightarrow & \boxed{s(\underline{len(\boxed{cdr(\underline{U})}^{-})})}^{+}
\end{array}
$$

Creational rules are mainly extracted from destructor style recursive definitions. For instance, the last three rules above are extracted from the destructor style recursive definitions of +, conscells and len, respectively. The first rule is from a lemma about s and p.

Table 5: Creational Wave Rules

$$
Cond \rightarrow \eta(\nu_1, \ldots, \nu_m) \Rightarrow
$$

$$
\boxed{\zeta(\underline{\eta(\boxed{\zeta_1^1(\underline{\nu_1})}^{-}, \ldots, \boxed{\zeta_m^1(\underline{\nu_m})}^{-})}, \ldots, \underline{\eta(\boxed{\zeta_1^k(\underline{\nu_1})}^{-}, \ldots, \boxed{\zeta_m^k(\underline{\nu_m})}^{-})})}^{+}
$$

where $Cond$ is a formula and η, the ζ_i^js and ζ are terms with distinguished arguments. ζ may be empty, but the ζ_i^js and η must not be. The ζ_i^js are the new anti-wave fronts and ζ is the new wave front. Both wave fronts and anti-wave fronts are indicated by boxes, annotated with a + for wave fronts and a $-$ for anti-wave fronts[8].

The preconditions for applying creational wave-rules require two new items in addition to those for conditional wave-rules (definition 4). We postpone giving these until §8, definition 5, where we give the final set of preconditions is given for a general form of wave-rule.

Some examples of creational rules are given in table 5.

Here is an example destructor induction step case from the proof of the associativity of +. The destructor-style definition of + is as follows:

$$
\begin{array}{rcl}
U+V & = & \text{if } U=0 \text{ then } V \\
 & & \text{else } s(p(U)+V)
\end{array}
$$

From this definition we can readily extract the following creational rule:

$$
U \neq 0 \rightarrow U+V \Rightarrow \boxed{s(\underline{\boxed{p(\underline{U})}^{-}+V})}^{+}
$$

which is used in the first two steps of the proof as follows: (The induction hypothesis is to the left of the $\vdash$ and the induction conclusion is to its right.)

$$
\boxed{p(\underline{x})}^{+} + (y+z) = (\boxed{p(\underline{x})}^{+} + y) + z \quad \vdash \quad x + (y+z) = (x+y)+z
$$

[8] All wave-fronts in previous sections of this paper should be regarded as annotated with an implicit +.

$$\boxed{p(\underline{x})}^{+} + (y+z) = (\boxed{p(\underline{x})}^{+} + y) + z \;\vdash\; \boxed{s(\underline{\boxed{p(\underline{x})}^{-} + (y+z)})}^{+} = \boxed{s(\underline{\boxed{p(\underline{x})}^{-} + y})}^{+} + z$$

$$p(x) + (y+z) = (p(x)+y)+z \;\vdash\; \boxed{s(\underline{p(x)+(y+z)})}^{+} = \boxed{s(\underline{p(x)+y})}^{+} + z$$

The last step does not represent an object-level inference, but is just a rearrangement of the meta-level wave front annotations.

Rippling-across is applied immediately after any application of destructor style induction. It rewrites the induction conclusion into a form to which the other forms of rippling are applicable and neutralises wave-fronts in the induction hypotheses. Each of the wave-fronts in each induction hypothesis is considered in turn. Let $\mathcal{C}$ be the set of creational rules with corresponding anti-wave-fronts. By 'corresponding' we mean that not only must the wave-front and anti-wave-front match, but they must occur at the same relative position in their respective expressions, *i.e.* if the wave-front, $\boxed{\xi(\underline{\chi})}^{+}$, occurs within the induction hypothesis, $\phi(\eta(\boxed{\xi(\underline{\chi})}^{+}))$, then the creational rule must have the form $\eta(\nu) \Rightarrow \boxed{\zeta(\underline{\eta(\boxed{\xi(\underline{\nu})}^{-})})}^{+}$.

If $\mathcal{C}$ is empty for some wave-front in some induction hypothesis, then this hypothesis is unusable, so it is labelled 'unviable' and is not considered further. Otherwise, each of the rules in $\mathcal{C}$ is considered in turn. The subexpression, $\eta(\chi)$, in the induction conclusion, $\phi(\eta(\chi))$, is rewritten with this rule. Each of the newly introduced anti-wave-fronts is neutralised with any corresponding wave-fronts in viable induction hypotheses, *i.e.* their wave-front annotations are both dropped. Note that we attempt to neutralise all new anti-wave-fronts and not just the one that triggered the rule application. Note also that corresponding wave-fronts in several induction hypotheses may be neutralised and not just the one that triggered the rule application. Note finally that induction hypotheses may be discovered to be unviable after an anti-wave-front has been neutralised with one of its wave-fronts. This neutralisation must be undone, and another way must be sought neutralise the anti-wave-front.

This process succeeds if at least one induction hypothesis remains viable and all its wave-fronts are neutralised. If any anti-wave-front in the induction conclusion remains unneutralised then it is merged with the wave-front that immediately surrounds it.

To illustrate this process, consider the following example, which has been chosen to exhibit all the various complications. The theorem to be proved is:

$$\forall L{:}list(nat).\ conscells(L) \geq len(L)$$

where *conscells* counts the number of cons cells used in a list and *len* counts its length. Suppose we try to prove this with a *car/cdr* destructor induction on lists. The step case will be:

$$l \neq nil$$

$$conscells(\boxed{car(\underline{l})}^{+}) \geq len(\boxed{car(\underline{l})}^{+})$$

$$conscells(\boxed{cdr(\underline{l})}^{+}) \geq len(\boxed{cdr(\underline{l})}^{+})$$

$$\vdash \quad conscells(l) \geq len(l)$$

Note that there are two induction hypotheses. We will refer to them as the *car* induction hypothesis and the *cdr* induction hypothesis, respectively.

The only relevant creational rules are:

$$x \neq nil \quad \rightarrow \quad conscells(U) \quad \Rightarrow \quad \boxed{s(\underline{conscells(\boxed{car(\underline{U})}^{-})} + \underline{conscells(\boxed{cdr(\underline{U})}^{-})})}^{+}$$

$$x \neq nil \quad \rightarrow \quad len(U) \quad \Rightarrow \quad \boxed{s(\underline{len(\boxed{cdr(\underline{U})}^{-})})}^{+}$$

We consider, in turn, each of the wave-fronts in each of the induction hypotheses. Consider the first wave-front of the *car* induction hypothesis. The *conscells* rule has a corresponding anti-wave-front. Applying it produces:

$$l \neq nil$$

$$conscells(\boxed{car(\underline{l})}^{+}) \geq len(\boxed{car(\underline{l})}^{+})$$

$$conscells(\boxed{cdr(\underline{l})}^{+}) \geq len(\boxed{cdr(\underline{l})}^{+})$$

$$\vdash \boxed{s(\underline{conscells(\boxed{car(\underline{l})}^{-})} + \underline{conscells(\boxed{cdr(\underline{l})}^{-})})}^{+} \geq len(l)$$

Each of the two new anti-wave-fronts neutralises with a corresponding wave-front in one of the two induction hypotheses.

$$l \neq nil$$

$$conscells(car(l)) \geq len(\boxed{car(\underline{l})}^{+})$$

$$conscells(cdr(l)) \geq len(\boxed{cdr(\underline{l})}^{+})$$

$$\vdash \boxed{s(\underline{conscells(car(l))} + \underline{conscells(cdr(l))})}^{+} \geq len(l)$$

Now consider the second wave-front of the *car* induction hypothesis. Note that neither of the creational rules contains an anti-wave-front corresponding to this wave-front, so the *car* induction hypothesis is labelled 'unviable' and takes no further part in the proof. We will delete it. The *car* anti-wave-front created by the *conscells* rule was neutralised by a wave-front in this unviable induction hypothesis. Since this *car* anti-wave-front cannot be neutralised in any other way in must be reinstated. So the current proof state is:

$$l \neq nil$$

$$conscells(cdr(l)) \geq len(\boxed{cdr(\underline{l})}^{+})$$

$$\vdash \boxed{s(\underline{conscells(\boxed{car(\underline{l})}^{-})} + \underline{conscells(cdr(l))})}^{+} \geq len(l)$$

Now consider the second wave-front of the *cdr* induction hypothesis. The *len* rule has a corresponding anti-wave-front. Applying it and neutralising the corresponding wave and anti-wave-fronts produces:

$$l \neq nil$$

$$conscells(cdr(l)) \geq len(cdr(l))$$

$$\vdash \boxed{s(\underline{conscells(\boxed{car(\underline{l})}^{-})}) + \underline{conscells(cdr(l))})}^{+} \geq \boxed{s(\underline{len(cdr(l))})}^{+}$$

The induction conclusion now contains an unneutralised anti-wave-rule. This must be merged with the wave-front that immediately surrounds it, to produce:

$$\boxed{s(conscells(car(l)) + \underline{conscells(cdr(l))})}^{+} \quad \geq \quad \boxed{s(\underline{len(cdr(l))})}^{+}$$

Constructor-style rippling can now be brought to bear on the induction conclusion using the longitudinal wave-rules:

$$\begin{aligned} \boxed{s(\underline{U})}^{+} \geq \boxed{s(\underline{V})}^{+} \quad &\Rightarrow \quad U \geq V \\ \boxed{U + \underline{V}}^{+} \geq W \quad &\Rightarrow \quad V \geq W \end{aligned}$$

To apply these rules it is necessary first to split the left hand side wave-front.

$$\boxed{s(\underline{\boxed{conscells(car(l)) + \underline{conscells(cdr(l))}}^{\uparrow}})}^{\uparrow} \quad \geq \quad \boxed{s(\underline{len(cdr(l))})}^{\uparrow}$$

and ripple each wave-front out separately. Strong fertilization then applies, completing the step case.

8 A General Format for Wave Rules

We have now introduced three different kinds of wave-rule: longitudinal, transverse and creational, each of which can also be conditional. In the search for more power we have lost some of the original simplicity of rippling. There seems to be a danger that this complexity will grow indefinitely. In this section we try to reduce the complexity by defining a general format which includes all three kinds of wave-rule.

As a starting point we repeat below the forms of the three different kinds of wave-rules.

Longitudinal

$$\begin{aligned} Cond \quad \rightarrow \quad & \eta(\boxed{\xi_1(\underline{\mu_1^1}, \ldots, \underline{\mu_1^{p_1}})}^{\uparrow}, \ldots, \boxed{\xi_n(\underline{\mu_n^1}, \ldots, \underline{\mu_n^{p_n}})}^{\uparrow}) \quad \Rightarrow \\ & \boxed{\zeta(\underline{\eta(\varpi_1^1, \ldots, \varpi_n^1)}, \ldots, \underline{\eta(\varpi_1^k, \ldots, \varpi_n^k)})}^{\uparrow} \end{aligned}$$

Transverse

$$\begin{aligned} Cond \quad \rightarrow \quad & \eta(\boxed{\xi_1(\underline{\mu_1})}^{\uparrow}, \ldots, \boxed{\xi_n(\underline{\mu_n})}^{\uparrow}, \nu_1, \ldots, \nu_m) \quad \Rightarrow \\ & \eta(\mu_1, \ldots, \mu_n, \boxed{\zeta_1(\underline{\nu_1})}^{\downarrow}, \ldots, \boxed{\zeta_m(\underline{\nu_m})}^{\downarrow}) \end{aligned}$$

Creational

$$\begin{aligned} & Cond \rightarrow \eta(\nu_1, \ldots, \nu_m) \Rightarrow \\ & \boxed{\zeta(\underline{\eta(\boxed{\zeta_1^1(\underline{\nu_1})}^{-}, \ldots, \boxed{\zeta_m^1(\underline{\nu_m})}^{-})}, \ldots, \underline{\eta(\boxed{\zeta_1^k(\underline{\nu_1})}^{-}, \ldots, \boxed{\zeta_m^k(\underline{\nu_m})}^{-})})}^{+} \end{aligned}$$

These three formats can be generalised to:

$$
\begin{array}{l}
Cond \;\rightarrow\; \eta(\boxed{\xi_1(\underline{\mu_1^1},\ldots,\underline{\mu_1^{p_1}})}^{\uparrow},\ldots,\boxed{\xi_n(\underline{\mu_n^1},\ldots,\underline{\mu_n^{p_n}})}^{\uparrow},\nu_1,\ldots,\nu_l,\nu_{l+1},\ldots,\nu_m) \;\Rightarrow \\
\boxed{\begin{array}{l}
\zeta(\underline{\eta(\varpi_1^1,\ldots,\varpi_n^1,\boxed{\zeta_1^1(\underline{\nu_1})}^{-},\ldots,\boxed{\zeta_l^1(\underline{\nu_l})}^{-},\boxed{\zeta_{l+1}^1(\underline{\nu_{l+1}})}^{\downarrow},\ldots,\boxed{\zeta_m^1(\underline{\nu_m})}^{\downarrow})},\ldots \\
\ldots,\underline{\eta(\varpi_1^k,\ldots,\varpi_n^k,\boxed{\zeta_1^k(\underline{\nu_1})}^{-},\ldots,\boxed{\zeta_l^k(\underline{\nu_l})}^{-},\boxed{\zeta_{l+1}^k(\underline{\nu_{l+1}})}^{\downarrow},\ldots,\boxed{\zeta_m^k(\underline{\nu_m})}^{\downarrow})})
\end{array}}^{\uparrow}
\end{array}
$$

where each ϖ_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, μ_i^l. For each j, at least one ϖ_i^j must be a wave-hole. *Cond* is a formula. η, the ξ_is, ζ and the ζ_i^j are terms with distinguished arguments. ζ may be empty, but the ξ_is, ζ_i^j and η must not be. For notational convenience, but without loss of generality, we have ordered the arguments of η so that the outwards wave-fronts are the first n, the anti-waves are the next l and the inwards wave-fronts are the last $m-l$. Either n, m or l may be 0. If m is 0 then the rule is purely longitudinal. If n is 0 and $l=m$ then the rule is purely creational. If ζ is empty and $l=0$ then $p_i=1$ and the rule is purely transverse.

This generalised form also increases the power of rippling, since it allows hybrid wave-rules, *e.g.* rules that are partly longitudinal and partly transverse, for instance,

$$
\sum_{i=0}^{\boxed{s(\underline{n})}^{\uparrow}} u_i \;\Rightarrow\; \boxed{u_0+\underline{\textstyle\sum_{i=0}^{n} u_{\boxed{s(\underline{i})}^{\downarrow}}}}^{\uparrow}
$$

Rippling-out, rippling-sideways and rippling-across are combined in $\mathit{CL}^{A}\mathit{M}$ as the selective application of wave-rules in the above general format. The wave annotations determine what sort of rippling takes place, with hybrid forms of rippling happening in a natural way. In practice, all ζ_i^j wave-fronts are labelled as inward directed. Whether they are treated as anti-wave-fronts depends on whether there is a positive wave-front to be neutralised in the induction hypothesis.

Rippling-in applies general wave-rules from right to left. When used for rippling-in, the wave-front directional annotations are different from that given above. So rippling-in applies, left to right, waves of the form:

$$
\begin{array}{l}
Cond \;\rightarrow \\
\boxed{\zeta(\underline{\eta(\varpi_1^1,\ldots,\varpi_n^1,\boxed{\zeta_1^1(\underline{\nu_1})},\ldots,\boxed{\zeta_m^1(\underline{\nu_m})}),\ldots,\eta(\varpi_1^k,\ldots,\varpi_n^k,\boxed{\zeta_1^k(\underline{\nu_1})},\ldots,\boxed{\zeta_m^k(\underline{\nu_m})})})}^{\downarrow} \\
\qquad\qquad \Rightarrow\; \eta(\boxed{\xi_1(\underline{\mu_1^1},\ldots,\underline{\mu_1^{p_1}})}^{\downarrow},\ldots,\boxed{\xi_n(\underline{\mu_n^1},\ldots,\underline{\mu_n^{p_n}})}^{\downarrow},\nu_1,\ldots,\nu_m)
\end{array}
$$

where the range of the symbols is as before, except that ζ must not be empty, since purely transverse rules are not used for rippling-in. The direction of the ζ_i^j wave-fronts can be either inwards or outwards.

$\mathit{CL}^{A}\mathit{M}$ has as a single *ripple* tactic that applies general wave rules left to right or right to left. It analyses the current induction conclusion and chooses a rule which will move the wave fronts in a desirable direction while keeping the skeleton intact. The generalised *ripple* tactic embodies, as special cases: rippling-out, rippling-sideways, rippling-across and rippling-in. Despite the complications of applying wave rules in both left/right orientations, the duplication of skeletons, the choice of directions to ripple in, *etc.*, the *ripple* tactic always terminates. This is proved in appendix A.

The preconditions of generalised rippling are the same as those for conditional rippling (definition 3), but with two additional items for dealing with creational rippling.

Definition 5 (Preconditions of Applying Generalised Wave-Rules) *The preconditions of applying a wave-rule to rewrite a sub-expression of the induction conclusion are that:*

- *the left hand side of the rule matches the sub-expression;*
- *the sub-expression contains at least one wave-front;*
- *each wave-front in the sub-expression is matched with a wave-front of the same kind in the rule;*
- after neutralisation of anti-wave-fronts at least one viable induction hypothesis remains;
- After merging unneutralised anti-wave-fronts there exists at least one instance of the skeleton in the new induction conclusion; *and*
- *the condition of the wave-rule is provable from the current hypotheses.*

The generalised *ripple* tactic does not embody our last extension: existential rippling. Nor is existential rippling guaranteed to terminate. This form of rippling extends generalised rippling and is still under development. We describe its current state in the next section.

9 Rippling under Existential Quantifiers

The *ripple* tactic developed so far is suitable for proving theorems containing universal quantifiers, but it cannot cope with existential quantifiers. The problem is that the witness of an existential quantifier, *i.e.* the term that the quantifier asserts to exist, may well contain occurrences of the induction variable. In this case the witness of the existential quantifier in the induction conclusion will contain wave-fronts and must take part in rippling. Until the identity of the witness is known this rippling cannot be completed. On the other hand, we would like to use the possibilities for successful rippling to help us determine what the witness should be - a Catch-22 situation.

To illustrate the problem consider the following theorem.

$$\forall X{:}nat.\forall Y{:}nat.\forall Z{:}nat.\exists W{:}nat.\ X+(Y+Z)=W+Z$$

where the witness for W is clearly going to be $X+Y$, but we don't know that yet. If we apply $s(X)$ induction then the induction conclusion is:

$$\forall Y{:}nat.\forall Z{:}nat.\exists W{:}nat.\ \boxed{s(\underline{x})}+(Y+Z)=W+Z$$

This can be rippled to:

$$\forall Y{:}nat.\forall Z{:}nat.\exists W{:}nat.\ \boxed{s(\underline{x+(Y+Z)})}=W+Z \tag{36}$$

but no further rippling is possible.

In the purely universal version of this theorem, namely the associativity of $+$, we would ripple the right hand side into the form $\boxed{s(\underline{\ldots})}$, and then ripple both s functions away together. The right hand side s function would be put in place by an application of wave-rule (4), reproduced below:

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

but this rule is not applicable unless W is instantiated to something of the form $\boxed{s(\underline{\ldots})}$. One solution to this problem is to create an existential version of the wave-rule (4) and apply this instead, namely:

$$\begin{aligned} &\exists \boxed{U'}{:}nat.\ \boxed{s(\underline{x+(y+z)})} = \boxed{U'} + V \\ \Rightarrow\quad &\exists \boxed{U}{:}nat.\ \boxed{s(\underline{x+(y+z)})} = \boxed{s(\underline{\boxed{U}+V})} \end{aligned} \tag{37}$$

Note that U' is marked as a wave-front because it is implicitly $\boxed{s(\underline{U})}$. For the application of this rule to (36) to qualify as rippling, W must also be seen as a wave-front. This is legitimate because, as we have seen it might be instantiated to a term containing a wave-front. We therefore regard all existentially quantified variables in goals as *potential wave-fronts*, which we mark as $\boxed{W}$. The same argument holds for U in (37). So applying (37) to (36) gives:

$$\forall Y{:}nat.\forall Z{:}nat.\exists \boxed{W_1}{:}nat.\ \boxed{s(\underline{x+(Y+Z)})} = \boxed{s(\underline{\boxed{W_1}+Z})}$$

which can be rippled to:

$$\forall Y{:}nat.\forall Z{:}nat.\exists \boxed{W_1}{:}nat.\ x+(Y+Z) = \boxed{W_1} + Z)$$

to which strong fertilization applies, completing the step case of the proof.

In this proof the relationship between U and U' in wave-rule (37), and hence that between W and W_1 in the induction conclusion, is not explicitly recorded. This does not matter if we are just interested in provability, but it does if we are also interested in the identity of the existential witness. The main application of our research is to program synthesis, [Bundy *et al* 90a], in which the existential witnesses play the vital role of defining the programs to be synthesised. Fortunately, the logic we use — a variant of Martin-Löf Type Theory — also contains a solution to the problem. Each rule of inference of the logic has an associated program construction rule. A partial program is associated with each formula of the proof and is incrementally constructed in the course of the proof. The relationship between U and U' in wave-rule (37) is contained in the program associated with its proof from wave-rule (4). The program associated with the step case of our example is a recursion equation partially defining a function, say w, which can serve as the witness of our existential quantifier, namely:

$$w(s(X), Y, Z) \quad = \quad s(w(X, Y, Z))$$

The program associated with the base case will be:

$$w(0, Y, Z) \quad = \quad Y$$

so that $w(X, Y, Z) = X + Y$.

Existential wave-rules are generated dynamically by *CIAM*, from regular wave-rules, according to need. This is essential since each regular wave-rule can give rise to infinitely many existential wave-rules. In the general case, the regular wave-rule:

$$\begin{aligned} Cond \quad \rightarrow \quad & \eta(\boxed{\xi_1(\underline{\mu_1^1}, \ldots, \underline{\mu_1^{p_1}})}^{\uparrow}, \ldots, \boxed{\xi_i(\underline{\mu_i^1}, \ldots, \underline{\mu_i^{p_i}})}^{\uparrow}, \boxed{\xi_i(\underline{\mu_{i+1}^1}, \ldots, \underline{\mu_{i+1}^{p_{i+1}}})}^{\uparrow}, \ldots \\ & \ldots, \boxed{\xi_n(\underline{\mu_n^1}, \ldots, \underline{\mu_n^{p_n}})}^{\uparrow}, \nu_1, \ldots, \nu_m) \\ \Rightarrow \quad & \boxed{\begin{array}{l} \zeta(\underline{\eta(\varpi_1^1, \ldots, \varpi_i^1, \varpi_{i+1}^1, \ldots, \varpi_n^1, \boxed{\zeta_1^1(\underline{\nu_1})}^{\downarrow}, \ldots, \boxed{\zeta_m^1(\underline{\nu_m})}^{\downarrow})}, \ldots \\ \ldots, \underline{\eta(\varpi_1^k, \ldots, \varpi_n^k, \varpi_{i+1}^k, \ldots, \varpi_n^k, \boxed{\zeta_1^k(\underline{\nu_1})}^{\downarrow}, \ldots, \boxed{\zeta_m^k(\underline{\nu_m})}^{\downarrow})}) \end{array}}^{\uparrow} \end{aligned}$$

can be made into the existential rule:

$$
\begin{aligned}
Cond \quad \rightarrow \quad & \exists\, \chi_1 : \tau_1 \ldots \exists\, \chi_i : \tau_i . \\
& \phi(\eta(\chi_1, \ldots, \chi_i, \boxed{\xi_i(\underline{\mu^1_{i+1}}, \ldots, \underline{\mu^{p_{i+1}}_{i+1}})}^{\uparrow}, \ldots, \boxed{\xi_n(\underline{\mu^1_n}, \ldots, \underline{\mu^{p_n}_n})}^{\uparrow}, \nu_1, \ldots, \nu_m)) \\
\Rightarrow \quad & \exists\, \varpi^1_1 : \tau_1 \ldots \exists\, \varpi^1_i : \tau_i \ldots \exists\, \varpi^k_1 : \tau_1 \ldots \exists\, \varpi^k_i : \tau_i . \\
& \phi(\boxed{\begin{array}{l} \zeta(\underline{\eta(\varpi^1_1, \ldots, \varpi^1_i, \varpi^1_{i+1}, \ldots, \varpi^1_n, \boxed{\zeta^1_1(\underline{\nu_1})}^{\downarrow}, \ldots, \boxed{\zeta^1_m(\underline{\nu_m})}^{\downarrow})}, \ldots \\ \quad \ldots, \underline{\eta(\varpi^k_1, \ldots, \varpi^k_i, \varpi^k_{i+1}, \ldots, \varpi^k_n, \boxed{\zeta^k_1(\underline{\nu_1})}^{\downarrow}, \ldots, \boxed{\zeta^k_m(\underline{\nu_m})}^{\downarrow})}) \end{array}}^{\uparrow})
\end{aligned}
$$

where, ϕ is an arbitrary formula and, for notational convenience but without loss of generality, we have ordered the arguments of η so that those to be existentially quantified are the first i. Note that the ϖ_q and ϖ^q_j must all be variables. Any non-empty subset of the n wave-fronts of η can be chosen to be existentially quantified, *i.e.* it can be done in $2^n - 1$ ways.

Each side of the regular wave-rule is surrounded by a formula ϕ, so that the existential quantifiers can be applied to a formula, rather than a term. *CIAM* chooses ϕ, and the subset of wave-fronts to be existentially quantified, dynamically by inspection of the formula to be rewritten. For instance, in wave-rule (37), $\phi(\ldots)$ is chosen to be $\boxed{s(\underline{x + (y + z)})} = \ldots$.

The presence of the potential wave-fronts around the μ^q_j means that the well-founded measure, introduced in appendix A, might not be decreased by an existential rule application. Thus rippling under existential quantifiers is not guaranteed to terminate. One way to see this is that until the identity of the existential witnesses is fixed it is not clear how much rippling we will need to do. Since we can add to the complexity of these witnesses during the construction of the proof, the amount of rippling required is unbounded.

Work on existential rippling is still in progress. We have an initial implementation and have successfully tested it on a few simple examples. The problems of search are clearly exacerbated by the presence of existential quantifiers, but more empirical work is required to assess the size of the problem and extent to which existential rippling cures it.

10 Results

The major hypothesis advanced in this paper is that in a significant number of inductive proofs the *ripple* tactic alone will successfully guide the step case between the application of the inductive rule to the use of the induction hypothesis. We predict that we can use just the *ripple* tactic in place of the mixture of heuristics used by other inductive theorem provers for this stage of inductive proofs, with similar or improved success. For instance, in the Boyer-Moore theorem prover, Nqthm, this stage of the proof is controlled by the heuristics of: rewriting terms; eliminating destructors; using definitions, axioms and lemmas; and by the user giving hints. We predict that rippling will successfully replace all of these giving a more uniform search control regime and will prove some theorems that Nqthm cannot prove without user guidance.

To test this prediction we have used rippling to control the search of a large number of theorems drawn from the literature. Below we give statistics on its performance which confirm our prediction. We then give a number of selected examples to show how rippling generalises and replaces the heuristics used in other theorem provers and how it can guide steps which in other systems must be controlled by human intervention.

10.1 Quantitative Study

We have tested rippling by running CL^AM successfully on 82 simple inductive theorems drawn from [Boyer & Moore 79], [Kanamori & Fujita 86] and other standard sources. In each proof, the *ripple* tactic is solely responsible for controlling the proof between the application of *induction* and the application of *fertilization*, *i.e.* rippling alone puts the induction conclusion into a form in which it can be simplified with the aid of the induction hypothesis.

A selection of the theorems proved by CL^AM is listed in table 6. Table 7 shows the kinds of rippling used to prove each of these theorems. Note that CL^AM can combine two or more kinds of rippling within a single proof.

$$
\begin{array}{rcll}
\forall X{:}nat.\forall Y{:}nat.\forall Z{:}nat.\ X+(Y+Z) & = & (X+Y)+Z & (38) \\
\forall L{:}list(int).\forall M{:}list(int).\ rev(L) <> M & = & qrev(L,M) & (39) \\
\forall A{:}int.\forall L{:}list(int).\ count(A, sort(L)) & = & count(A,L) & (40) \\
\forall L{:}list(int).\ length(sort(l)) & = & length(l) & (41) \\
\forall X{:}int.\forall N{:}nat.\forall L{:}list(int).\ X \in nth(N,L) & \rightarrow & X \in L & (42) \\
\forall X{:}nat.\forall Y{:}nat.\ even(X) \wedge even(Y) & \rightarrow & even(X+Y) & (43) \\
\forall X{:}nat.\forall Y{:}nat.\ min(X,Y) & \leq & max(X,Y) & (44) \\
\forall X{:}int.\forall A{:}list(int).\forall B{:}list(int).\ X \in A & \rightarrow & X \in A \cup B & (45) \\
\forall X{:}nat.half(X+X) & = & X & (46) \\
\forall L{:}list(int).\forall P{:}list(int).rotate(length(L), L <> P) & = & P <> L & (47) \\
\forall X{:}nat.\forall Y{:}nat.X & \leq & (X+Y) & (48) \\
\forall X{:}sexp.\forall Y{:}sexp.\forall Z{:}sexp.Z \in X & \rightarrow & X \in (X <> Y) & (49) \\
\forall X{:}int^{+}.\exists L{:}list(prime).prod(L) & = & X & (50)
\end{array}
$$

Table 6: Examples of Theorems Proved using Rippling

To test the hypotheses, advanced in §2.5, that there is little or no search during rippling, we conducted the following experiment. Each wave-rule required during the proof of each of our 82 test theorems was made available to CL^AM. A complete test run of the 82 theorems was made and any occurrences of branching were automatically recorded. There was no branching at all. That is, at each step, no more than one wave-rule met the rippling preconditions. We did not even detect the only kind of branching anticipated in §2.5, namely a choice between longitudinal and transverse wave-rules. It seems that, in practice, it is rare to have both kinds of wave-rule apply to the same expression.

10.2 Case Studies

Below we give some case studies where rippling, as implemented in CL^AM, can automatically prove theorems that other theorem provers cannot prove unaided. In each case we show how rippling is responsible for the improved performance. In comparing the performance of theorem provers we have tried to aid comparison between them by translating their formalisms into the one used in this paper. We claim that these translations do not violate the spirit of their proof attempts.

Theorem Number	Out	Sideways	In	Across	Existential
38	x				
39	x	x			
40	x				
41	x				
42	x	x			
43	x	x			
44	x	x			
45	x	x			
46	x		x		
47	x	x	x		
48	x			x	
49	x			x	
50	x				x

Each row of the table shows the kinds of rippling used by CLAM to prove a theorem in table 6. The first entry gives the number of the theorem from table 6 and the subsequent entries give the kinds of rippling used in its proof.

Table 7: Kinds of Rippling used in Proofs

10.2.1 Naive and Tail Recursive List Reversal

Consider the theorem:

$$\forall L{:}list(nat).\forall M{:}list(nat).\ rev(L) <> M \quad = \quad qrev(L, M)$$

discussed in §6.2. This example is based on one in [Burstall & Darlington 77] used to illustrate folding and unfolding. Unfolding is like rippling but restricted to the application recursive rewrite rules and applied exhaustively. Folding is the inverse to unfolding, *i.e.* it is like rippling-in. In fact, the proof cannot be handled by folding and unfolding alone, because it requires the application of a non-recursive equation: the associativity of <>. Darlington calls the associativity of <> a *law*. Laws are applied by user intervention.

As we have seen in §6.2, the associativity of <> can be viewed as a wave rule, in various ways. The appropriate way to use it is (usually uniquely) determined by the location of the wave-fronts in the induction conclusion. Thus rippling subsumes unfolding and is able to account for the application of a'law' that otherwise requires human intervention. In this case the annotation required on the associativity of <> is as a transverse rule:

$$(\boxed{\underline{U} <> V}^{\uparrow}) <> W \quad \Rightarrow \quad U <> (\boxed{V <> \underline{W}}^{\downarrow})$$

and it is used for the following piece of rippling-sideways:

$$\begin{aligned} (\boxed{\underline{rev(l)} <> (h :: nil)}^{\uparrow}) <> \lfloor m \rfloor \quad &= \quad qrev(l, \left\lfloor \boxed{h :: \underline{m}}^{\downarrow} \right\rfloor) \\ rev(l) <> (\left\lfloor \boxed{(h :: nil) <> \underline{m}}^{\downarrow} \right\rfloor) \quad &= \quad qrev(l, \left\lfloor \boxed{h :: \underline{m}}^{\downarrow} \right\rfloor) \end{aligned}$$

Nqthm can prove this theorem automatically provided the associativity of <> is available as a rewrite rule *and that the user has oriented it in the appropriate direction.*

10.2.2 Rotate Length

Consider the theorem:

$$\forall L{:}list(\tau).\forall A{:}list(\tau).\ rotate(len(L), L <> A) \quad = \quad A <> L$$

The function $rotate$ takes a number, n, and a list, l, and returns a list identical to l but with the first n elements transported from the beginning to the end. Nqthm cannot prove this theorem. This is because it does not have the rippling-sideways tactic. Although part of this missing tactic is mimicked by Nqthm's symbolic evaluation, the final rippling-in part is missing.

Both Nqthm and $CI\!AM$ choose a hd :: induction[9] on L. Since A is also universally quantified, it is a sink, so the $CI\!AM$ representation of the induction conclusion is:

$$rotate(len(\boxed{hd :: \underline{tl}}^{\uparrow}), (\boxed{hd :: \underline{tl}}^{\uparrow}) <> \lfloor a \rfloor) \quad = \quad \lfloor a \rfloor <> (\boxed{hd :: \underline{tl}}^{\uparrow})$$

The recursive definitions of <>, len and $rotate$ provide the following wave rules:

$$\begin{aligned} (\boxed{Hd :: \underline{Tl}}^{\uparrow}) <> L \quad &\Rightarrow \quad \boxed{Hd :: (\underline{Tl <> L})}^{\uparrow} \\ len(\boxed{Hd :: \underline{Tl}}^{\uparrow}) \quad &\Rightarrow \quad \boxed{s(\underline{len(Tl)})}^{\uparrow} \\ L \neq nil \quad \rightarrow \quad rotate(\boxed{s(\underline{N})}^{\uparrow}, L) \quad &\Rightarrow \quad rotate(N, \boxed{cdr(\underline{L}) <> (car(\underline{L}) :: nil)}^{\downarrow}) \end{aligned}$$

[9] Actually Nqthm uses a destructor-style version of this, but it will aid the comparison and does not seriously distort the account, to pretend that it uses a constructor-style induction.

$CL\quad M$ can use these wave rules for rippling; Nqthm can use the recursive definitions for symbolic evaluation. Thus both systems can use these definitions, together with some simplification, to rewrite the induction conclusion to:

$$rotate(len(tl), \boxed{(\underline{tl <> \lfloor a \rfloor}) <> (hd :: nil)}^{\downarrow}) \quad = \quad \lfloor a \rfloor <> (\boxed{hd :: \underline{tl}}^{\uparrow})$$

$C\!L^A\!M$ completes the rippling of the left-hand wave-front. It applies the associativity of $<>$ backwards as a longitudinal wave to ripple this wave-front towards its sink. Without a rather special-purpose wave-rule the right-hand-side wave-front is blocked.

This puts the induction conclusion in the form:

$$rotate(len(tl), tl <> (\left\lfloor \boxed{\underline{a} <> (hd :: nil)}^{\downarrow} \right\rfloor)) \quad = \quad \lfloor a \rfloor <> (\boxed{hd :: \underline{tl}}^{\uparrow})$$

Since the left-hand-side is fully rippled, weak fertilization applies to give:

$$(\left\lfloor \boxed{\underline{a} <> (hd :: nil)}^{\uparrow} \right\rfloor) <> tl \quad = \quad a <> (hd :: tl)$$

Rippling-in[10] now uses the associativity of $<>$ as a transverse wave-rule to ripple the remaining wave-front out of the sink. This gives:

$$(\lfloor a \rfloor <> (\boxed{(hd :: nil) <> \underline{tl}}^{\downarrow})) \quad = \quad a <> (hd :: tl)$$

to which *cancellation* and *simplification* apply, yielding:

$$hd :: tl \quad = \quad hd :: tl$$

which is proved by the *identity* tactic.

Provided the associativity of $<>$ is available to Nqthm and has been appropriately oriented by the user, it will mimic the inwards ripple on the left-hand-side. Although the left-hand-side is now fully rippled, Nqthm is unable to apply weak fertilization because its logic does not permit any instantiation of A except at the time of determining the induction rule. Nqthm then loses it way, over-generalises and fails.

Nqthm lacks any heuristics corresponding to rippling-in or rippling-sideways and because of the limitations of its logic is unable to fully exploit sinks. If the transverse wave rules used for rippling-sideways are based on recursive definitions (as is the case with the definition of *rotate*) then symbolic evaluation will mimic the effect of rippling-sideways. Also if it happens to have a rewrite rule oriented in the direction required for a particular application of rippling-out, -sideways or -in, then it will mimic these processes. But this is more a matter of luck than judgement; in many cases no happy coincidence will arise and Nqthm fail.

11 Implementation

All the various forms of rippling described above have been implemented and tested in the *Oyster*/$C\!L^A\!M$ system, which is written in Prolog. Full documentation of *Oyster* is available in [Horn 88] and of $C\!L^A\!M$ in [van Harmelen 89]. Versions for Quintus and Sicstus Prolog exist. A *ripple* tactic has been built. A tactic is a Prolog program which applies rules of inference of the *Oyster* type theory. The *ripple* tactic forms part of the *ind_strat* tactic, which controls the whole of an application of inductive inference. Within *ind_strat*, *ripple* is applied during the step case

[10] This is the kind of rippling-in using transverse wave-rules that was described in §6.3.

of an induction just after the *induction* tactic and before the *fertilize* tactic (see figure 1). The *ripple* tactic calls the *wave* sub-tactic, which applies individual wave-rules. Each tactic has a corresponding method, which specifies its preconditions and effects using a meta-logic implemented as a Prolog program. For instance, the precondition of the *wave* method is a Prolog program representing the preconditions given in English in definition 5. These methods are used by *CIAM* to build a proof plan customised for the current theorem. Most proof plans are built from *ind_strat* and a few other high-level tactics; their sub-tactics are only rarely used independently. More details of the *Oyster/CIAM* system can be found in [Bundy *et al* 91].

In order to ensure the correct, selective application of wave-rules by *ripple* it is necessary to mark wave-fronts and wave-holes in both goals and wave-rules. *CIAM* does this by annotating the object-level expressions with meta-level functions. The Prolog functions `wave_front/3` and `wave_hole/1` are used to indicate wave-fronts and wave-holes, respectively. The arguments of `wave_front` are:

1. the type of wave-front: `definite` or `potential`;
2. the direction of rippling: `out` or `in`; and
3. the object-level expression itself.

The `in` argument is used to indicate both inwards directed and anti-wave-fronts. The argument of wave_hole is just the object-level expression. Thus the expression $\boxed{s(\underline{x})}^{\uparrow} + y$ is represented by the Prolog expression:

$$\mathtt{wave\_front(definite, out, s(wave\_hole(x))) + y}$$

This representational technique has the advantage that alignment of wave-fronts in wave-rules and goals is automatically checked as a side effect of the process of matching rule to goal.

Whenever a new definition is made or a new theorem proved, *CIAM* analyses it and extracts any wave-rules that it suggests. If necessary, these rules are proved as lemmas from the original definition or theorem. The wave-rules are automatically annotated with wave-fronts, possibly in multiple ways and in different left/right orientations. These multiple ways include weakenings, reverse duals, and both transverse and longitudinal readings of the same formula (*cf.* the many different ways in which the associativity laws can be regarded as wave-rules, see tables 1 and 4). They are then stored for future use. The rippling capabilities of *CIAM* thus increase with use. Although the search space also increases this is not a problem in practice because the strong preconditions of rippling tame the combinatorial explosion.

12 Related Work

The use of sets of rewrite rules in theorem provers is commonplace. The most common usage is to apply them exhaustively to put expressions in a normal form. By 'exhaustively' we mean that rules are applied until no further rule applications are possible. In contrast, we apply wave-rules selectively, that is as well as a rule being applicable some additional preconditions must be met, *i.e.* that each wave-front in the sub-expression being rewritten must match a wave-front in the rule rewriting it, (see definition 1).

Termination of the exhaustive application of a set of rewrite rules is an important property. This is usually proved by exhibiting a well-founded ordering that is strictly decreased by each rule application. Since wave-rules are selectively applied, we are able to take account of the preconditions on the application when proving termination. For instance, in appendix A we used the information that each rule application moves the wave-front along a fixed length path. This allows us to use wave-rules in either orientation without the risk of looping, something that is impossible with exhaustive application. The first exploitation, of which we are aware, of selective application of rewrite rules to prove termination, is in [Bundy & Welham 81].

The decision as to which rules to include in a rewrite rule set and which way round to orient them is often partially determined by the termination ordering. It might also be determined automatically by a Knuth-Bendix process which is trying to create a confluent set, [Knuth & Bendix 70]. Alternatively, it may simply be determined by the intuitions of the designer. In contrast, wave-rules are chosen and oriented automatically by *CLAM* on the basis of their syntactic structure, as given in §8. The same formula may used in several different ways, including both orientations. This technique for selection and orientation ensures that the rule set meets its specification of rippling wave-fronts. It also means that we can add to the rule set dynamically without risk of deviation from this specification, of non-termination or of combinatorial explosion.

The first documented use of rewrite rules based on recursive definitions was in [Burstall & Darlington 77], who called it *unfolding*. Exhaustive unfolding has since become standard practice in inductive theorem provers. [Aubin 75] first pointed out how unfolding rippled (what we call) wave-fronts through the induction conclusion. He called this process *rippling-out* and exploited it in his *generalisation-apart* heuristic. In [Bundy 88] we pointed out how Aubin's observation could be used to extend the class of rewrite rules used for rippling-out from recursive definitions to wave-rules. The present paper extends further our initial definition of wave-rules.

Building on the proposals in [Bundy 88], Hutter has implemented a rippling-like heuristic in the INKA inductive theorem prover, [Hutter 90]. His implementation differs from the one described here in the following respects.

- His rewrite rule set contains any rule that moves wave-fronts[11] within a skeleton. This is a larger rule set than our wave-rules, because it includes rules that move wave-fronts in directions other than out, sideways or across.
- These rewrite rules are controlled by a series of strategies that correspond roughly to rippling-out, rippling-sideways and rippling-across. There are minor differences that affect the relative power of the two implementations, but these are not significant.

Hutter independently invented the concept of rippling-across and possibly rippling-sideways. He also invented an interesting technique for the representation of wave-fronts. Each function has an additional argument that indicates whether it belongs to the skeleton or the wave-front. The rippling precondition of wave-front agreement is taken care of by an enhanced matcher. As currently realised this technique has some messy complications, *e.g.* each variable has to have an infinite number of aliases and the matcher is very complicated. We hope it may be possible to simplify the technique and adopt it in *CLAM*.

Rippling is the central idea behind our rational reconstruction of the induction heuristics embedded in the Nqthm theorem prover, [Boyer & Moore 79]. In [Bundy *et al* 89] we have shown how the method used by Boyer and Moore to select the best form of induction to prove a theorem can be rationally reconstructed as a look ahead to see what form of induction maximises the chances of that rippling will succeed. Similar remarks apply to the generalisation of a theorem prior to induction — an area on which we are currently working. Our understanding of the relation between rippling and induction has also enabled us to extend the Boyer-Moore heuristics beyond purely universal theorem to those involving existential quantifiers.

The 'black box' nature of the Nqthm rewriting subsystem makes a direct comparison with rippling difficult. It is possible, however, to make some comparison based upon some general observations.

[11] He calls wave-fronts, *contexts*.

Firstly, the most general wave rule schema presented in §8 demonstrates the uniformity of rippling. Such uniformity within the rewriting capabilities of Nqthm is less obvious. Nqthm has four classes of rules of which REWRITE and ELIM relate to rippling. The REWRITE class encapsulates both recursive definitions and the use of lemmas. The rules classified as ELIM are used to eliminate destructors from an induction hypothesis. In some sense the ELIM rules play the same role as creational wave rules do within destructor style induction.

Secondly, the merits of rippling with respect to controlling the application of rewrite rules is demonstrated by its success at handling potentially non-terminating sets of rewrites. In particular, by being more focussed, rippling is able to fold as well as unfold recursive definitions. In contrast, Nqthm makes no provision for the folding of definitions and the onus of ensuring against the non-termination of rewriting is left to the user. The Nqthm user must decide how to orient lemmas as rewrite rules. Rippling determines the orientation of lemmas as wave-rules. The same lemma or definition can be used in both orientations by rippling without the risk of non-termination.

Thirdly, rippling was developed for a much stronger logic than Nqthm. As a consequence rippling can be seen as providing a more general control strategy. For instance, the notion sinks cannot be exploited by Nqthm since universal quantification is implicit within its logic.

13 Limitations and Further Work

The work described above has made considerable extensions to rippling. Some further testing may suggest additional extensions. For instance, our experience with higher-order functions is limited, so rippling may need extending in this area.

Some mopping-up operations remain, *e.g.* the consideration of normal forms for wave-front representation (see §2.4) and the ability to overlap old and new wave-fronts positions, as in the transverse wave-rule:

$$rotate(\boxed{s(\underline{N})}^{\uparrow}, \boxed{Hd :: \underline{Tl}}^{\uparrow}) \quad \Rightarrow \quad rotate(N, \boxed{\underline{Tl} <> (Hd :: nil)}^{\downarrow})$$

Also, we are exploring the analogue of rippling when programs are represented not in the functional programming style of nested functions but in the logic programming style of conjunctions of literals, *e.g.*

$$plus(x, y, z) \wedge even(z, w)$$

rather than:

$$even(x + y)$$

However, our main effort in future work will be going, not into further extensions, but into developing the other theorem proving tactics that rippling suggests and enables.

13.1 Unblocking Ripples

Rippling can fail due to the lack of an appropriate wave-rule. We say that it is blocked. We are currently exploring various techniques for *unblocking* ripples. These mostly work by perturbing the induction conclusion in some way, but without disturbing the skeleton, in the hope that a wave-rule will apply to the perturbed expression. For instance, transverse wave-rules can be used in this way. Here is a simple, but rather artificial, example. Consider the blocked induction conclusion (25) from §4, which contained the sub-expression $x + \boxed{s(\underline{x})}^{\uparrow}$, but where the wave-rule (26):

$$U + \boxed{s(\underline{V})}^{\uparrow} \quad \Rightarrow \boxed{s(\underline{U + V})}^{\uparrow}$$

is not available. Suppose, however, that the rule:

$$U + \boxed{s(\underline{V})}^{\uparrow} \quad \Rightarrow \boxed{s(\underline{U})}^{\uparrow} + V \tag{51}$$

is available. We can first apply rule (51), to unblock the ripple by moving the wave-front to the adjacent argument. This gives $\boxed{s(\underline{x})}^{\uparrow} + x$ to which rule (4) applies to give $\boxed{s(\underline{x + x})}^{\uparrow}$, as required. Blocked from moving outwards directly, the wave-front is first moved sideways around the blockage, and then outwards. Note the non-standard annotation of rule (51), where the new wave-front is directed outwards rather than inwards. This reflects the non-standard use to which the rule is being put.

An alternative method of unblocking in the above example would be to apply the commutative law of $+$:

$$U + V \quad \Rightarrow V + U$$

It is often the case that commutative laws, and similar permutative rewrite rules, can be used to unblock ripples. Unfortunately, note that commutative laws are not wave-rules, and their uninhibited use could lead to looping. We are still studying how best to apply such permutative rules to unblock ripples. Possibilities include; (a) using them under loop checking constraints, (b) building them into the unification algorithm or (c) using them to generate permutated variants of wave-rules. Solution (c) is currently the favourite.

Another technique for unblocking ripples is to allow a blocked outwards wave-front with a single wave-hole containing a sink to reverse direction and be rippled-in to this sink. This change of direction is a substitute for a missing transverse wave-rule. Instead of the wave-front going out, sideways and in, it goes out then in. Note that this is not just backtracking, because at least the later part of the path in will be different from the earlier part of the path out. For instance, consider the proof of:

$$\forall L{:}list(nat).\forall N{:}nat.\ len(L) + N \quad = \quad len_2(L, N)$$

where len is the naive list length function and len_2 is the tail-recursive version. Suppose we apply $h :: l$ induction and ripple-out with the recursive definitions of len and $+$ and ripple-sideways with the recursive definition of len_2.

$$\boxed{s(\underline{len(l) + \lfloor n \rfloor})}^{\uparrow} \quad = \quad len_2(l, \left\lfloor \boxed{s(\underline{n})}^{\downarrow} \right\rfloor)$$

The ripple is now blocked. It can be unblocked by rippling-in with the wave-rule:

$$U + \boxed{s(\underline{V})}^{\uparrow} \quad \Rightarrow \quad \boxed{s(\underline{U + V})}^{\uparrow}$$

applied right to left, to produce:

$$s(len(l) + \left\lfloor \boxed{s(\underline{n})}^{\downarrow} \right\rfloor) \quad = \quad len_2(l, \left\lfloor \boxed{s(\underline{n})}^{\downarrow} \right\rfloor)$$

as required. This unblocking is a substitute for the (possibly missing) transverse wave-rule:

$$\boxed{s(\underline{U})}^{\uparrow} + V \quad \Rightarrow \quad U + \boxed{s(\underline{V})}^{\downarrow}$$

13.2 Middle-Out Reasoning

In [Bundy *et al* 89] we describe how rippling can suggest which induction rule to use to prove a theorem. The idea is to look-ahead to the step case and find an induction rule that would enable all the wave-fronts to be rippled at least once. This look-ahead is conducted as part of the preconditions of the *induction* tactic. We call it *recursion analysis*.

Recursion analysis is both inadequate and too inflexible: inadequate, because it only checks for the first step of rippling, and inflexible because it insists that all wave-fronts must be rippled at least once, when the best available induction rule may only be able to ripple some of them. In practice, we overcome these limitations with *ad hoc* devices. We are thus seeking an alternative mechanism that overcomes them in a principled way.

The mechanism we are currently exploring is to use meta-variables to postpone the choice of induction rule and allow it to be fixed by higher order unification during rippling, [Bundy *et al* 90a]. We call this *middle-out reasoning* because it allows us to postpone the beginning of a proof and tackle the middle first, allowing decisions made in the middle to influence the beginning. We can illustrate this with an example. Consider the theorem:

$$\forall X{:}nat.\forall Y{:}nat.\ even(X) \wedge even(Y) \rightarrow even(X+Y)$$

Assume that the induction rule will introduce the induction terms $\chi(x)$ or $\gamma(y)$ or both. The step case will be:

$$\begin{array}{l} even(x) \wedge even(y) \rightarrow even(x+y) \\ \vdash \quad even(\chi(\underline{x})) \wedge even(\gamma(\underline{y})) \rightarrow even(\chi(\underline{x}) + \gamma(\underline{y})) \end{array}$$

where we have used potential wave-fronts since we are not sure yet which of them will turn out to be required. The wave-rules (4) and (6) from table 1 will both apply to the induction conclusion: the first in one way and the second in two. Each of these rule applications will eventually lead us to the same conclusion, but the most instructive is to apply rule (4), so let us start with that.

$$\begin{array}{l} even(x) \wedge even(y) \rightarrow even(x+y) \\ \vdash \quad even(\boxed{s(\underline{\chi_1(\underline{x})})}) \wedge even(\gamma(\underline{y})) \rightarrow even(\boxed{s(\underline{\chi_1(\underline{x}) + \gamma(\underline{y})})}) \end{array}$$

where $\chi(X) = s(\chi_1(X))$. Now, since we always prefer to ripple real wave-fronts before potential ones, the ripple with (6) is indicated, giving:

$$\begin{array}{l} even(x) \wedge even(y) \rightarrow even(x+y) \\ \vdash \quad even(\chi_2(\underline{x})) \wedge even(\gamma(\underline{y})) \rightarrow even(\boxed{s(\underline{\boxed{s(\underline{\chi_2(\underline{x})})} + \gamma(\underline{y})})}) \end{array}$$

where $\chi_1(X) = s(\chi_2(X))$. Another ripple with (4) is indicated, giving:

$$\begin{array}{l} even(x) \wedge even(y) \rightarrow even(x+y) \\ \vdash \quad even(\chi_2(\underline{x})) \wedge even(\gamma(\underline{y})) \rightarrow even(\boxed{s(s(\underline{\chi_2(\underline{x}) + \gamma(\underline{y})}))}) \end{array}$$

and finally another with (6), giving:

$$\begin{array}{l} even(x) \wedge even(y) \rightarrow even(x+y) \\ \vdash \quad even(\chi_2(\underline{x})) \wedge even(\gamma(\underline{y})) \rightarrow even(\chi_2(\underline{x}) + \gamma(\underline{y})) \end{array}$$

to which strong fertilization applies, instantiating $\chi_2(X) = X$ and $\gamma(Y) = Y$. Thus $\chi(X) = s(s(X))$, *i.e.* a double induction on X has been chosen, as required.

Middle-out reasoning can also be used to deal with existential quantifiers, and we are exploring this usage in parallel with our work on existential rippling. Meta-variables are used to postpone the commitment to an existential witness, then unification during subsequent rippling is used to retrospectively identify the witness. This use of meta-variables is fairly standard.

13.3 Generalising Apart

Aubin in [Aubin 75] showed how to use recursion analysis to suggest generalisations of the theorem. He calls this mechanism *generalisation apart.* Consider the following theorem.

$$\forall X{:}nat.\ X + (X + X) = (X + X) + X$$

This is a simple instance of the associativity of $+$. However, if we try to prove this using the obvious $s(X)$ induction the ripple gets blocked. Only the wave-fronts surrounding the first and fourth occurrences of X can be fully rippled using wave-rule (4) from table 1. No wave-rules apply at all to the third, fifth and sixth occurrences and the second occurrence only ripples one level. The answer is to generalise the theorem to:

$$\forall X{:}nat.\forall Y{:}nat.\ X + (Y + Y) = (X + Y) + Y$$

before trying $s(X)$ induction. Two of the occurrences of X are generalised apart from the other four. This time wave-fronts only appear in positions where they can be fully rippled.

Recursion analysis can help suggest this generalisation. It locates those induction variable occurrences that can be rippled at least once. Unfortunately, this is not exactly what is wanted. It will locate the first, second and fourth occurrences and suggest as generalisation the non-theorem:

$$\forall X{:}nat.\forall Y{:}nat.\ X + (X + Y) = (X + Y) + Y$$

We want to reject the second occurrence on the grounds that it ripples once, but then gets blocked. So to reject this occurrence requires looking ahead more than one level. This suggests integrating generalisation suggestions into the middle-out reasoning process described in §13.2 above. Jane Hesketh is currently looking at this and other uses of middle-out reasoning for generalising theorems.

13.4 Difference Matching

We have assumed, so far, that the wave annotations in the induction conclusion are initialised by the induction rule. However, it is desirable to have an independent method of introducing wave-fronts into an expression. One situation is when the induction rule suggests inappropriate wave-fronts and another is when rippling is used for non-inductive proofs. We illustrate each of these situations below.

A situation when rippling requires wave-fronts different to those initialised by the induction rule occurs in one of the proofs of the following simple example. Consider the theorem.

$$\forall N{:}nat.\ even(N) \vee even(s(N))$$

Recursion analysis uses wave-rule (6) from table (1 to suggest an $s(s(N))$ induction, and this yields to the *ind_strat* tactic with no problems. However, there is also a simple proof using $s(N)$ induction. The wave annotation suggested by this induction rule is:

$$even(\boxed{s(\underline{n})}) \vee even(s(\boxed{s(\underline{n})}))$$

but this is no good. What is required instead is the wave annotation:

$$even(s(n)) \vee even(\boxed{s(s(\underline{n}))})$$

The commutativity of $\vee$ makes the skeleton of this induction conclusion match the induction hypothesis. One application of rule (6) then peters out the ripple and allows strong fertilization to finish the step case.

Summing series provides many examples of the use of rippling for non-inductive proofs[12]. For instance, consider the problem of finding a closed form solution to the series:

$$\sum_{i=0}^{n}(a.i+b) \tag{52}$$

where a and b are constants, using the standard forms:

$$\sum_{I=0}^{N} K = (N+1).K \tag{53}$$

$$\sum_{I=0}^{N} I = \frac{N.(N+1)}{2} \tag{54}$$

and the wave-rules:

$$\sum_{I=0}^{N}\boxed{K.\underline{U_I}} \Rightarrow \boxed{K.\underline{\textstyle\sum_{I=0}^{N} U_I}} \tag{55}$$

$$\sum_{I=0}^{N}\boxed{\underline{U_I}+\underline{V_I}} \Rightarrow \boxed{\underline{\textstyle\sum_{I=0}^{N} U_I}+\underline{\textstyle\sum_{I=0}^{N} V_I}} \tag{56}$$

where K is a constant.

We can use rippling by treating the standard forms as induction hypotheses and using them to suggest wave-fronts in the problem series. Rippling-out these wave-fronts will allow weak fertilization to use the standard forms to replace occurrences of $\sum$ with closed form expressions. One of the standard forms that our problem series, (52), will match against is (53) above. This suggests the wave annotations:

$$\sum_{i=0}^{n}\boxed{a.i+\underline{b}}$$

We can then ripple out with wave-rule (56) and weak fertilize with (53).

$$\begin{aligned}\sum_{i=0}^{n}\boxed{a.i+\underline{b}} &= \boxed{\textstyle\sum_{i=0}^{n} a.i+\underline{\textstyle\sum_{i=0}^{n} b}} \\ &= \sum_{i=0}^{n} a.i+(n+1).b\end{aligned}$$

We can then rematch the problem against the standard form (54) to suggest a new wave annotation, ripple out with wave-rule (55) and weak fertilize with (54).

$$\begin{aligned}\sum_{i=0}^{n}\boxed{a.\underline{i}}+(n+1).b &= \boxed{a.\underline{\textstyle\sum_{i=0}^{n} i}}+(n+1).b \\ &= a.\frac{n.(n+1)}{2}+(n+1).b\end{aligned}$$

[12] We are indebted to Toby Walsh for suggesting this application of rippling and providing the example below.

This technique has been implemented as an MSc project by Alex Nunes and is reported in [Walsh *et al* 91].

David Basin and Toby Walsh, in our research group, have designed a *difference matching* algorithm which inserts wave-fronts into goals, [Basin & Walsh 91]. It takes two expressions playing the role of induction hypothesis and induction conclusion, respectively, and identifies wave-fronts in the induction conclusion so that its skeleton matches the induction hypothesis. This algorithm can be used by a 'stand-alone' rippling tactic which can find its own wave annotations and thus act independently of any induction rule. It has been implemented by Alex Nunes and is used in his series summing program to extend rippling in this way.

14 Conclusion

In this paper we have described the *ripple* tactic: the key search control heuristic for guiding inductive proofs. We started with the basic *ripple-out* tactic and then extended it in seven directions: multi-wave rippling, conditional rippling, rippling-in, rippling-sideways, rippling-across and rippling under existential quantifiers. We then described a generalised *ripple* tactic which includes the first six of these extensions as special cases and allows hybrids between them.

Our extensions to rippling can be summarised as follows:

multi-wave rippling: The generalisation of the form of wave-rules to simultaneous rippling of one or more wave-fronts each containing one or more wave-holes, finally matching with one or more induction hypotheses. This enables the use of strong fertilization and extends the proof plan to non-equational and non-equivalential theorems.

conditional rippling: The generalisation of rippling-out to conditional wave-rules. The use of complementary sets of conditional rules to decide when to split into sub-cases.

rippling-in: The use of longitudinal wave-rules backwards either after weak fertilization or rippling-sideways, thus enabling these tactics to complete their task.

rippling-sideways: The use of transverse wave-rules to move wave-fronts into sinks. It enables *Oyster-CIAM* to take advantage of free variables in the induction hypothesis.

rippling-across: The adaptation of these constructor induction techniques to destructor induction.

rippling under existential quantifiers: The adaptation of rippling to deal with existential quantifiers.

Rippling plays the central role in our proof plan for inductive proofs. As well as controlling the proof search in the step case of an induction it also helps to determine what inductive rule to use, how to generalise the theorem and what the existential witnesses should be. This proof plan and the rippling tactic have proved surprisingly successful. It has been successfully tested on 82 simple examples from the literature. No search is required. Our proof plan is much less sensitive than the Boyer-Moore theorem prover to the order in which theorems are presented; if an essential lemma is not available for rippling when required, an appropriate instance of it is sometimes proved 'in-line'.

However, our proof plan does not constitute a decision procedure for induction proofs, which is in any case an undecidable area. Nor does rippling always succeed in bridging the gap between induction hypothesis and induction conclusion. Rippling will block if the right wave-rule is not available. If it blocks on both sides of an equation, even weak fertilization will not apply. One way to proceed in such situations is shake the current goal into a form in which a wave-rule *will* apply. For instance, a wave-front might be moved sideways to a non-sink with a commutative law or transverse wave-rule. The non-standard use of wave-rules for unblocking has already been implemented in *CIᴬM* and other techniques are currently under investigation.

The *Oyster-CIᴬM* system is written in Prolog. A general form of rippling has been implemented which includes as special cases: multi-wave rippling, conditional rippling, rippling-in, rippling-sideways and rippling-across. Existential rippling is partially implemented and under development.

A Termination

The various forms of rippling direct and restrict the search for an inductive proof. The main purpose of using rippling is to avoid the combinatorial explosion that is associated with exhaustive search techniques. It is, therefore, important to demonstrate that rippling always terminates. The termination of rippling seems plausible because it is always directed towards some finite objective, *e.g.* rippling a wave-front outwards until it is beached. Since expressions are of finite depth, the skeleton remains unchanged throughout rippling and each ripple moves the wave-front a non-zero distance through the skeleton in a fixed direction, so eventually no further movement will be possible. Unfortunately, this simple argument is complicated by the extensions to rippling described in this paper.

A.1 Why Proving Termination is Complicated

Below we list these complicating factors.

1. Creational rules create new wave-fronts as well as neutralising old ones. We must ensure that there is a limit to the new wave-fronts introduced.

2. Rippling-in sends wave-fronts in the opposite direction to rippling-out. We must ensure that the two forms of rippling do not cause a loop. Furthermore, when rippling-in there can be a choice of inwards directions. We must ensure that rippling does not loop between these choices.

3. Wave-rules can duplicate parts of the skeleton, which also increases the number of wave-fronts. We must ensure that there is a limit to the new wave-fronts introduced.

4. One wave-front can split into several, thus increasing the number of wave-fronts to be rippled. Each may be rippled along a different path. We must ensure that there is a limit to the new wave-fronts introduced.

5. The path along which a wave-front is being rippled may be partially shared with another wave-front, and the two ripplings may interfere. We must ensure that this interference does not cause looping.

6. If rippling fails to get a wave-front to its target then it backs up and tries a different target. We must ensure that the search space is finite.

A.2 The Design of the Well-Founded Measure

To show that rippling always terminates we exhibit a well-founded measure that is strictly decreased by each application of a wave-rule. In designing this measure we address the above complications as follows:

1. Problem 1 is solved by treating unneutralised positive wave-fronts in the induction hypotheses as if they were really outward wave-fronts in the corresponding positions in the induction conclusion. Creational rules then cancel these wave-fronts with anti-wave-fronts and reduce the measure.

2. Problem 4 is solved by attaching a weight to each wave-front such that if a compound front is split into several parts then the weight of a compound front is equal to the combined weights of its parts. We will adopt the simple solution that the weight of a wave-front is the number of function applications in it. So, for instance, the weights of $\boxed{s(s(\underline{x}))}$ and $\boxed{s(\boxed{s(\underline{x})})}$ will both be 2 — 1 for each application of s. Note that weights are natural numbers.

3. Problem 2 is solved by treating inward directed wave-fronts and outward directed ones, separately. The overall measure is a lexicographically ordered pair of an outwards measure and an inwards measure, so progress on the outward tree comes before progress on the inward one. Note that outward fronts can be relabelled as inward, but not vice versa. Thus a wave-front can change direction at most once.

4. Problems 3 and 5 are solved by considering the overall effect of the wave-front application on the whole skeleton. Different techniques are used for the outwards and inwards measures. For the outwards measure we use nested multisets where the nesting represents the structure of the skeleton and the possible positions within it for wave-fronts. For the inwards measure we treat the skeleton as a tree labelled with (unnested) multisets of all the weights at that position in any of the duplicates. Sometimes distinct induction hypotheses have different skeletons. These skeletons differ only in the name of the induction variable. In the discussion below these skeletons are treated as identical.

5. Problem 6 is solved because the search space is finite. There is only ever a finite branching rate because there are only a finite number of wave-rules. Each branch of the space is finite as shown by the well-founded measure.

A.3 The Well Founded Measure

The measure attached to a sequent is a lexicographically ordered pair, in which the first element measures the outwards directed wave-fronts and the second element measures the inwards directed ones. The outwards measure is a nested multiset of weights. The inwards measure is the skeleton labelled with multisets of weights.

We start by defining the outwards, inwards and overall measures. We represent nested and unnested multisets with set notation, *e.g.* $\{\ldots\}$, $\emptyset$, $\cup$. We represent labelled trees by nested expressions in which the nodes are functions and the labels are superscripts on these functions. Finally, we define well-founded orders on the measures.

Definition 6 (Outwards Measure) $||\Gamma||^{\uparrow}$ *is the outwards measure of the sequent* Γ. *It is a nested multiset defined recursively in two stages. We first define a labelled* AND/OR *tree,* $tr(\Gamma)$, *that represents the duplicated skeleton with its weights, and then we define a nested multiset,* $nm(tr(\Gamma))$, *from this* AND/OR *tree. Thus,* $||\Gamma||^{\uparrow} = nm(tr(\Gamma))$.

$$\begin{aligned} tr(\Gamma \vdash \Delta) &= tr(\Delta) \\ tr(\eta(\mu_1,\ldots,\mu_n)) &= \&^{w}(\eta(tr(\mu_1),\ldots,tr(\mu_n))) \\ tr(\xi(\mu_1,\ldots,\mu_n)) &= \&^{|\xi|+w_1+\cdots+w_n}(tr(\mu_1^1),\ldots,tr(\mu_1^{p_1}),\ldots,tr(\mu_n^1),\ldots,tr(\mu_n^{p_n})) \end{aligned}$$

where:

- *η is an n-ary function in the skeleton and ξ is an n-ary function not in the skeleton,*
- *$\&$ is a new variadic function,*
- *w is the sum of the weights of the positive wave-fronts in the induction hypothesis at this position in the skeleton,*
- *$|\xi|$ is 1 if ξ is part of an outwards wave-front and 0 otherwise, and*
- *$tr(\mu_i) = \&^{w_i}(tr(\mu_i^1), \ldots, tr(\mu_i^{p_1}))$.*

$$\begin{array}{rcl} nm(\eta(\mu_1, \ldots, \mu_n)) & = & \{nm(\mu_1), \ldots, nm(\mu_n)\} \\ nm(\&^w(\mu_1, \ldots, \mu_n)) & = & \{w, nm(\mu_1), \ldots, nm(\mu_n)\} \end{array}$$

For example, if Γ is:

$$\ldots \vdash \boxed{s(max(\underline{max\_ht(l)}, \underline{max\_ht(r)}))}^{\uparrow} \geq \boxed{s(min(\underline{min\_ht(l)}, \underline{min\_ht(r)}))}^{\uparrow}$$

then $tr(\Gamma)$ is:

$$\&^0(\geq (\&^2(max\_ht(\&^0(l)), max\_ht(\&^0(r)))), \&^2(min\_ht(\&^0(l)), min\_ht(\&^0(r)))))$$

and $||\Gamma||^{\uparrow}$ is:

$$\{0, \{\{2, \{\{0, \emptyset\}\}, \{\{0, \emptyset\}\}\}, \{2, \{\{0, \emptyset\}\}, \{\{0, \emptyset\}\}\}\}\}$$

and, if Γ is:

$$\begin{array}{l} l \neq nil \\ conscells(car(l)) \geq len(\boxed{car(\underline{l})}^{+}) \\ conscells(cdr(l)) \geq len(\boxed{cdr(\underline{l})}^{+}) \\ \quad \vdash \boxed{s(\underline{conscells(car(l))} + \underline{conscells(cdr(l))})}^{+} \geq len(l) \end{array}$$

then $tr(\Gamma)$ is:

$$\&^0(\geq (\&^2(conscells(\&^{\overline{0}}(l)), conscells(\&^{\overline{0}}(l))), \&^0(len(\&^{\underline{2}}(l)))))$$

The underlined number is the sum of the weights of the *car* and *cdr* wave-fronts in the induction hypothesis. The two overlined weights are 0 because the *car* and *cdr* in the induction conclusion are neutralised anti-wave-fronts. $||\Gamma||^{\uparrow}$ is:

$$\{0, \{\{2, \{\{0, \emptyset\}\}, \{\{0, \emptyset\}\}\}, \{0, \{\{2, \emptyset\}\}\}\}\}$$

Definition 7 (Inwards Measure) *$||\Gamma||^{\downarrow}$ is the inwards measure of the sequent Γ. It is a tree labelled with multisets and is defined recursively as follows.*

$$\begin{array}{rcl} ||\Gamma \vdash \Delta||^{\downarrow} & = & ||\Delta||^{\downarrow} \\ ||\eta(\mu_1, \ldots, \mu_n)||^{\downarrow} & = & \eta^{\{0\}}(||\mu_1||^{\downarrow}, \ldots, ||\mu_n)||^{\downarrow} \\ ||\xi(\eta_1, \ldots, \eta_m)||^{\downarrow} & = & \eta^{\{|\xi| + w : w \in \sigma_1 \cup \cdots \cup \sigma_m\}}(u(||\eta_1^1||^{\downarrow}, \ldots, ||\eta_1^m)||^{\downarrow}, \ldots, u(||\eta_n^1||^{\downarrow}, \ldots, ||\eta_n^m))||^{\downarrow} \end{array}$$

where:

-
$$
\begin{aligned}
& u(\eta^{\sigma_1}(\|\mu_1^1\|^{\downarrow}, \dots, \|\mu_n^1)\|^{\downarrow}, \dots, \eta^{\sigma_i}(\|\mu_1^m\|^{\downarrow}, \dots, \|\mu_n^m))\|^{\downarrow} \\
= \quad & \eta^{\sigma_1 \cup \dots \cup \sigma_m}(u(\|\eta_1^1\|^{\downarrow}, \dots, \|\eta_1^m)\|^{\downarrow}, \dots, u(\|\eta_n^1\|^{\downarrow}, \dots, \|\eta_n^m))\|^{\downarrow}
\end{aligned}
$$

- *η is an n-ary function in the skeleton and ξ is an n-ary function not in the skeleton,*
- *$|\xi|$ is 1 if ξ is part of an inwards wave-front and 0 otherwise, and*
- *$\|\eta_i\|^{\downarrow} = \eta^{\sigma_i}(tr(\mu_i^1), \dots, \|\mu_i^n)\|^{\downarrow}$. Note that the function η will be the same for all $1 \leq i \leq m$ up to renaming of induction variables, which we assume to take place as required.*

For example,

$$\| \dots \vdash \boxed{s(max(\underline{max\_ht(l)}, \underline{max\_ht(r)}))}^{\downarrow} \geq \boxed{s(min(\underline{min\_ht(l)}, \underline{min\_ht(r)}))}^{\downarrow} \|^{\downarrow}$$

is:

$$\geq^{\emptyset} (max\_ht^{\{2,2\}}(l^{\emptyset}), min\_ht^{\{2,2\}}(l^{\emptyset}))$$

Note that induction variable r is renamed to l in order to make all copies of the skeleton identical.

Definition 8 (Sequent Measure) *$\|\Gamma\|$ is the measure of the sequent Γ. It is the pair of the outwards and inwards measure,* i.e.

$$\|\Gamma\| \quad = \quad \langle \|\Gamma\|^{\uparrow}, \|\Gamma\|^{\downarrow} \rangle$$

Definition 9 (Well Founded Order) *Let $\prec$ be an order on sequent measures defined as follows.*

- *We will use a left/right lexicographic ordering on the pairs. That is, $\langle o_1, i_1 \rangle \prec \langle o_2, i_2 \rangle$ iff either $o_1 \ll^* o_2$ or $o_1 = o_2$ and $i_1 \prec_i i_2$, where $\ll^*$ is the nested multiset order, [Dershowitz & Manna 79], based on the numeric, $<$, order on the weights.*
- *We will use a root/leaves lexicographic ordering, $\prec_i$, on the second element of the pair,* i.e. *the skeleton trees labelled with multisets of weights that measure the inward wave-fronts. Let i_1 and i_2 be trees and let d be their* upper difference set *(defined below), consisting of a list of pairs of multisets of weights. $i_1 \prec_i i_2$ iff d is non-empty and for each pair $\langle \sigma_1, \sigma_2 \rangle \in d$, $\sigma_1 \ll \sigma_2$, where $\ll$ is the multiset order, [Dershowitz & Manna 79], based on the numeric $<$ order on the weights..*

Each of these constituent orders is well-founded. The well-foundedness of $<$ on the natural numbers and of lexicographic orderings on fixed length tuples are well known. The well-foundedness of $\ll$ and $\ll^*$ are both proved in [Dershowitz & Manna 79]. The well-foundedness of $\prec_i$ on fixed shape trees follows by a similar argument for lexicographic orderings on fixed length tuples. So $\prec$ is well-founded.

Definition 10 (Upper Difference Set) *The difference set of two labeled trees of the same shape , i_1 and i_2 consists of all pairs of labels, $\langle \sigma_1, \sigma_2 \rangle$, such that:*

- *σ_1 is the label of node n_1 and σ_2 is the label of node n_2;*
- *n_1 and n_2 are corresponding nodes in i_1 and i_2, respectively;*
- *$\sigma_1 \neq \sigma_2$; and*
- *n_1 and n_2 are both minimal differing nodes,* i.e. *all corresponding nodes below them have equal labels.*

By corresponding nodes *we mean that each node is at the same position in its tree, where the two trees have the same shape.*

A.4 Rippling Decreases the Measure

We next show that our measure is strictly decreased whenever a wave-rule is applied. To do this we will use the general format of wave-rules given in §8. However, we will also have to use the preconditions of rippling, including the outward and inward annotations on wave-fronts. Since these differ for different directions of rippling, the proof will be in two cases: forwards applications of wave-rules and backwards applications of wave-rules.

A.4.1 Forwards Applications of Wave Rules

When rippling-out, rippling-sideways, rippling-across, and hybrids of these, wave-rules are used forwards. The general format of wave-rules used when used forwards is:

$$\begin{array}{l} Cond \;\rightarrow\; \eta(\boxed{\xi_1(\underline{\mu_1^1}, \ldots, \underline{\mu_1^{p_1}})}^{\uparrow}, \ldots, \boxed{\xi_n(\underline{\mu_n^1}, \ldots, \underline{\mu_n^{p_n}})}^{\uparrow}, \nu_1, \ldots, \nu_l, \nu_{l+1}, \ldots, \nu_m) \;\Rightarrow \\ \boxed{\begin{array}{l} \zeta(\underline{\eta(\varpi_1^1, \ldots, \varpi_n^1, \boxed{\zeta_1^1(\underline{\nu_1})}^{-}, \ldots, \boxed{\zeta_l^1(\underline{\nu_l})}^{-}, \boxed{\zeta_{l+1}^1(\underline{\nu_{l+1}})}^{\downarrow}, \ldots, \boxed{\zeta_m^1(\underline{\nu_m})}^{\downarrow})}, \ldots \\ \ldots, \underline{\eta(\varpi_1^k, \ldots, \varpi_n^k, \boxed{\zeta_1^k(\underline{\nu_1})}^{-}, \ldots, \boxed{\zeta_l^k(\underline{\nu_l})}^{-}, \boxed{\zeta_{l+1}^k(\underline{\nu_{l+1}})}^{\downarrow}, \ldots, \boxed{\zeta_m^k(\underline{\nu_m})}^{\downarrow})}) \end{array}}^{\uparrow} \end{array}$$

Note that η is part of the skeleton. Either $l > 0$ or $n > 0$, *i.e.* if the rule is not purely creational then it must have some wave-fronts on the left hand side.

If there are some anti-wave-fronts, *i.e.* $l > 0$, then at least one of them must neutralise a positive wave-front. That is, for the rule application to be permitted then for some anti-wave-front, $\boxed{\zeta_i^j(\underline{\nu_i})}^{-}$, there must be a positive wave-front, $\boxed{\zeta_i^j(\underline{\nu_i'})}^{+}$, in the corresponding position of a viable induction hypothesis, such that ν_i' can be instantiated to ν_i. Recall that the weight of this positive wave is recorded in the outwards measure of every pair.

Lemma 1 (Decrease of Measure during Forward Application) *The sequent measure is strictly decreased by the forward application of a general wave-rule.*

As a result of the rewriting either some outwards directed wave-fronts are rippled-out or some anti-wave-fronts are introduced that neutralise some wave-fronts in the induction hypothesis. We will see that both of these events strictly decrease the outwards measure. The inwards measure may increase but since the outwards/inwards pairs are lexicographically ordered the overall measure is strictly decreased.

To see that the outwards measure is strictly decreased, consider the outwards measures, σ and σ', of the before and after sequents, respectively. These two sequents differ only at an occurrence of the η expression and at some of the first $n+l$ of its arguments. The before measure, σ, contains a nested multiset, $\{w, ||\eta||^{\uparrow}, \ldots\}$, corresponding to the η node. $||\eta||^{\uparrow}$ contains a nested multiset, σ_i, for each of the n outwards wave-fronts contained by η. The structure of σ_i is:

$$\{w_i, ||\mu_i^1||^{\uparrow}, \ldots ||\mu_i^{p_i}||^{\uparrow}\}$$

$||\eta||^{\uparrow}$ also contains a nested multiset $||\nu_i||^{\uparrow}$, for each of the l expressions that are about to have anti-wave-fronts wrapped around them. The after measure, σ', is just like σ except that an occurrence of $\{w, ||\eta||^{\uparrow}, \ldots\}$ is replaced by: $\{w + |\zeta|, ||\eta_1||^{\uparrow}, \ldots, ||\eta_k||^{\uparrow}, \ldots\}$.

We argue that this replacement causes a strict decrease in the outwards measure, since each of these new nested multisets is strictly smaller than one of the two they replace. Firstly, $w + |\zeta| \ll^* ||\eta||^{\uparrow}$, since $w+|\zeta|$ is a member of a base set, which is always smaller than any proper multiset under the nested multiset ordering, [Dershowitz & Manna 79]. Secondly, for $1 \leq j \leq k$, $||\eta_j||^{\uparrow} \ll^* ||\eta||^{\uparrow}$, since each $||\eta_j||^\uparrow$ is just like $||\eta||^\uparrow$ except that at least one of the nested multisets it contains is replaced with a strictly smaller one. This reduction takes place for one of the following reasons.

1. If $n > 0$ then in each $||\eta_j||^\uparrow$ at least one occurrence of one of the σ_i is replaced by the strictly smaller $\{w - |\xi_i|, ||\mu_i^{j'}||^\uparrow\}$, for some j'.

2. If $l > 0$ then in each η_j at least one anti-wave-front, $\boxed{\zeta_i^j(\underline{\nu_i})}^{-}$ is neutralised with its positive counter-part. This results in a strict decrease in $||\nu_i||^\uparrow$ and hence in $||\eta_j||^\uparrow$, since the weight $|\zeta_i^j|$ is subtracted from an element of $||\nu_i||^\uparrow$.

3. Any unneutralised anti-wave-front is merged with the ζ_i^j wave-front. Note that the η surrounding this anti-wave-front is also merged into the ζ wave-front and so $||\eta_1||^\uparrow$ is replaced in σ' by the strictly smaller $||\nu_i||^\uparrow$, and $|\zeta|$ is increased.

Since either $n > 0$ or $l > 0$ then one of these reductions must take place. Therefore, $\sigma' \lll^* \sigma$, *i.e.* the outwards measure is strictly decreased.

QED

A.4.2 Backwards Applications of Wave Rules

The general format of wave-rules when used for rippling-in is:

$$Cond \rightarrow$$

$$\boxed{\zeta(\underline{\eta(\varpi_1^1, \ldots, \varpi_n^1, \boxed{\zeta_1^1(\underline{\nu_1})}, \ldots, \boxed{\zeta_m^1(\underline{\nu_m})})}, \ldots, \underline{\eta(\varpi_1^k, \ldots, \varpi_n^k, \boxed{\zeta_1^k(\underline{\nu_1})}, \ldots, \boxed{\zeta_m^k(\underline{\nu_m})})})}^{\downarrow}$$

$$\Rightarrow \quad \eta(\boxed{\xi_1(\underline{\mu_1^1}, \ldots, \underline{\mu_1^{p_1}})}^{\downarrow}, \ldots, \boxed{\xi_n(\underline{\mu_n^1}, \ldots, \underline{\mu_n^{p_n}})}^{\downarrow}, \nu_1, \ldots, \nu_m)$$

Note that ζ must not be empty, since purely transverse rules are not used for rippling-in.

Lemma 2 (Decrease of Measure during Backward Application) *The sequent measure is strictly decreased by the backward application of a general wave-rule.*

1. Note that the ζ_i^j wave-fronts can be annotated either inward or outward. The rule application removes them all. So if $m \neq 0$ then this will result in a strict decrease in the size of either the outwards or inwards measure or both.

2. The inwards measures of the before and after formula differ only at the η node and at some of the first n of its arguments. Only the η position is a maximal differing node, so the corresponding weight pair from this makes up the upper difference set. Since the ζ wave-front has been removed from this position, the weight at this node has strictly decreased. So the measure of the inwards tree has strictly decreased, whereas the measure of the outwards tree has either decreased or stayed the same.

Thus the overall measure is strictly decreased.

QED

A.4.3 The Termination Proof

We can now put these two lemmas together to get the termination proof.

Theorem 1 (Termination of Rippling) *The rippling tactic will terminate.*

Rippling consists of repeated selective application of the general wave-rule format either forward or backward. By lemma 1 forward application decreases the sequent measure. By lemma 2 backward application decreases the sequent measure. The sequent measure is well-founded. Therefore, rippling terminates.

QED

Note that the argument only relied on the general format of wave-rules and the preconditions of rippling. Thus the termination proof holds for any set of wave-rules which fit the general format, even if some of the rules are reversals of others. No loop check is needed to ensure termination.

References

[Aubin 75] R. Aubin. Some generalization heuristics in proofs by induction. In G. Huet and G. Kahn, editors, *Actes du Colloque Construction: Amélioration et vérification de Programmes*. Institut de recherche d'informatique et d'automatique, 1975.

[Basin & Walsh 91] D. Basin and T. Walsh. Difference matching. Research Paper 556, Dept. of Artificial Intelligence, Edinburgh, 1991. Submitted to CADE-11.

[Boyer & Moore 79] R.S. Boyer and J.S. Moore. *A Computational Logic*. Academic Press, 1979. ACM monograph series.

[Bundy & Welham 81] A. Bundy and B. Welham. Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation. *Artificial Intelligence*, 16(2):189–212, 1981. Also available from Edinburgh as DAI Research Paper 121.

[Bundy 88] A. Bundy. The use of explicit plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *9th Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988. Longer version available from Edinburgh as DAI Research Paper No. 349.

[Bundy *et al* 89] A. Bundy, F. van Harmelen, J. Hesketh, A. Smaill, and A. Stevens. A rational reconstruction and extension of recursion analysis. In N.S. Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 359–365. Morgan Kaufmann, 1989. Also available from Edinburgh as DAI Research Paper 419.

[Bundy *et al* 90a] A. Bundy, A. Smaill, and J. Hesketh. Turning eureka steps into calculations in automatic program synthesis. In S.L.H. Clarke, editor, *Proceedings of UK IT 90*, pages 221–6, 1990. Also available from Edinburgh as DAI Research Paper 448.

[Bundy *et al* 90b] A. Bundy, F. van Harmelen, A. Smaill, and A. Ireland. Extensions to the rippling-out tactic for guiding inductive proofs. In M.E. Stickel, editor, *10th International Conference on Automated Deduction*, pages 132–146. Springer-Verlag, 1990. Lecture Notes in Artificial Intelligence No. 449. Also available from Edinburgh as DAI Research Paper 459.

[Bundy *et al* 91] A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7:303–324, 1991. Earlier version available from Edinburgh as DAI Research Paper No 413.

[Burstall & Darlington 77] R.M. Burstall and J. Darlington. A transformation system for developing recursive programs. *Journal of the Association for Computing Machinery*, 24(1):44–67, 1977.

[Constable *et al* 86] R.L. Constable, S.F. Allen, H.M. Bromley, *et al. Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, 1986.

[Dershowitz & Manna 79] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Comms. ACM*, 22(8):465–476, 1979.

[Horn 88] C. Horn. The Nurprl proof development system. Working paper 214, Dept. of Artificial Intelligence, Edinburgh, 1988. The Edinburgh version of Nurprl has been renamed Oyster.

[Hutter 90] D. 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.

[Kanamori & Fujita 86] T. Kanamori and H. Fujita. Formulation of induction formulas in verification of Prolog programs. In Joerg Siekmann, editor, *8th Conference on Automated Deduction*, pages 281–299. Springer-Verlag, 1986. Springer Lecture Notes in Computer Science No. 230.

[Knuth & Bendix 70] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational problems in abstract algebra*, pages 263–297. Pergamon Press, 1970.

[van Harmelen 89] F. van Harmelen. The CLAM proof planner, user manual and programmer manual. Technical Paper TP-4, Dept. of Artificial Intelligence, Edinburgh, 1989.

[Walsh *et al* 91] T. Walsh, A. Nunes, and A. Bundy. The use of proof plans to sum series. Research Paper 563, Dept. of Artificial Intelligence, Edinburgh, 1991. Submitted to CADE-11.