Using Failure to Guide Inductive Proof [*]

Andrew Ireland
Email: air@aisb.ed.ac.uk
Tel: +44-31-650-2721

Alan Bundy
Email: bundy@aisb.ed.ac.uk
Tel: +44-31-650-2716

Fax: +44-31-650-6516

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

Abstract

Lemma discovery and generalization are two of the major hurdles in automating inductive proof. This paper addresses aspects of these related problems. We build upon *rippling*, a heuristic which plays a pivotal role in guiding inductive proof. Rippling provides a high-level explanation of how to control the search for a proof. We demonstrate how this high-level explanation can be exploited productively when a proof attempt fails. In particular we show how failure can be used to focus the search for lemmas and generalizations.

1 Introduction

Inductive proof is central to the formal verification and synthesis of computer programs. Two of the major hurdles in automating inductive proof are the problems of lemma discovery and generalization. By way of motivation, consider the following conjecture:

$$\forall v{:}list(obj).\, \forall w{:}obj.\ \ rev(rev(v) <> w :: nil) = w :: v \qquad (1)$$

[*]The research reported in this paper was supported by SERC grant GR/H/23610.

where :: and <> are the infix list construction and concatenation operators respectively. The function *rev* denotes list reversal. A proof of (1) may be constructed by simple induction over the list v generating a base and step case proof obligation. The base case is established by exhaustive rewriting using the base equations for *rev* and <>. In the step case, we have an induction hypothesis of the form:

$$\forall w{:}obj.\ \ rev(rev(v) <> w :: nil) = w :: v$$

from which we have to establish the induction conclusion of the form:

$$rev(rev(u :: v) <> w :: nil) = w :: u :: v$$

Using the step equation for *rev*[1]:

$$rev(U :: V) \quad = \quad rev(V) <> (U :: nil)$$

the *LHS* of the induction conclusion rewrites to give:

$$rev((rev(v) <> u :: nil) <> w :: nil) = w :: u :: v \tag{2}$$

If we only have the definitions of <> and *rev* then the partial term structures "$\ldots <> u :: nil$" and "$u :: \ldots$" embedded within the induction conclusion prevent any further rewriting. These partial term structures also prevent us from making use of the induction hypothesis. One strategy to overcome this problem is to attempt to move these partial term structures towards a term which corresponds to a universally quantified variable in the induction hypothesis. In this example w is such a candidate term. This strategy requires, however, both the synthesis of an additional lemma and a generalization of the goal: Firstly, in order to rewrite the partial term structures we require the following lemma:

$$A <> (B :: C) \quad = \quad (A <> B :: nil) <> C$$

Secondly, the goal must be modified so that the universally quantified variable w in the induction hypothesis is of type $list(obj)$. One appropriate generalization is:

$$\forall v, w{:}list(obj).\ \ rev(rev(v) <> w) = rev(w) <> v \tag{3}$$

This paper addresses aspects of the related problems of lemma discovery and generalization. We will return to the example above in §7. Our approach builds upon *rippling*, a heuristic which plays a pivotal role in guiding inductive proof. Rippling provides a high-level explanation of how to control the search for a proof. We demonstrate how this high-level explanation can be exploited productively when a proof attempt fails.

[1]We use upper case letters to denote variables while constants are denoted by lower case letters.

2 Proof plans

It has been shown how the high-level structure of proofs can be expressed in terms of *proof plans* [Bun88]. Proof plans are used to guide the search for proofs by exploiting this high-level structure. A proof plan has two complementary components: a proof *method* and a proof *tactic*. By prescribing the structure of a proof at the level of primitive inferences, a tactic [GMW79] provides the guarantee part of the proof. In contrast a method provides a more declarative explanation of the proof by means of *preconditions*. Each method has associated *effects*. The execution of the effects simulates the application of the corresponding tactic. The effects are necessary to allow methods to be composed. In [Ire92] an extension to the proof planning framework is proposed in which proof *critics* are introduced in order to complement proof methods. The role of the proof critic is to capture patchable exceptions to the proof method. Since a proof method may fail in various ways, each method may be associated with a number of critics. Like methods, critics have preconditions and effects. The preconditions of a critic characterise interesting failures while the effects prescribe how failure can be overcome.

3 Heuristic guidance for inductive proof

In the context of inductive proof rippling [BvHSI90, BSvH+91] is concerned with step case proof obligations. Rippling controls the rewriting of induction conclusions so that induction hypotheses can be exploited. The process of using induction hypotheses has become known as *fertilization* [BM79]. To illustrate rippling and fertilization consider the following conjecture:

$$\forall v, w{:}list(obj).\ \ rev(v) <> w = qrev(v, w) \tag{4}$$

where $qrev$ is the tail recursive version of rev. A proof of (4) follows by $u :: v$ induction. We concentrate here on the step case proof obligation which gives us an induction hypothesis of the form

$$\forall w{:}list(obj).\ \ rev(v) <> w = qrev(v, w)$$

from which we have to establish the induction conclusion

$$rev(u :: v) <> w = qrev(u :: v, w)$$

Rippling is based upon the observation that a copy of the induction hypothesis is embedded within the induction conclusion. We call this the *skeleton* term structure. The role of the ripple method is to reduce the mismatch between the conclusion and hypothesis while preserving the skeleton term structure. Meta-level annotations are used to control this process:

$$rev(\boxed{u :: \underline{v}}^{\uparrow}) <> \lfloor w \rfloor = qrev(\boxed{u :: \underline{v}}^{\uparrow}, \lfloor w \rfloor) \tag{5}$$

The annotated terms $\boxed{u :: \underline{v}}^{\uparrow}$ and $\lfloor w \rfloor$ are called *wave* and *sink* terms respectively. A wave-term has a *wave-front* and one or more *wave-holes*. Wave-fronts highlight the mismatch between the induction conclusion and the induction hypothesis. The wave-front of $\boxed{u :: \underline{v}}^{\uparrow}$ is the partial term "$u :: \ldots$". A wave-hole is subterm of the a wave-term which matches against a subterm of the induction hypothesis. The wave-hole of $\boxed{u :: \underline{v}}^{\uparrow}$ is "v". The arrow is used to indicate the direction of movement of the wave-front in the expression tree. The need for directed wave-fronts will be explained shortly. A sink is used to delimit term structure in the induction conclusion which corresponds to a universally quantified variable in the induction hypothesis.

The ripple method restricts the rewriting of an induction conclusion to *wave-rules*, a syntactic class of rewrites which are guaranteed to make progress towards fertilization and preserve the skeleton term structure of the induction conclusion.

Wave-rules also contain wave-fronts. Recursive definitions provide a rich source of wave-rules[2]:

$$\boxed{U :: \underline{V}}^{\uparrow} <> W \Rightarrow \boxed{U :: \underline{(V <> W)}}^{\uparrow} \quad (6)$$

$$rev(\boxed{U :: \underline{V}}^{\uparrow}) \Rightarrow \boxed{\underline{rev(V)} <> U :: nil}^{\uparrow} \quad (7)$$

$$qrev(\boxed{U :: \underline{V}}^{\uparrow}, W) \Rightarrow qrev(V, \boxed{U :: \underline{W}}^{\downarrow}) \quad (8)$$

There are two basic kinds of wave-rules, *longitudinal* (6 and 7) and *transverse* (8). A longitudinal wave-rule moves a wave-front into a less nested position within the term structure while a transverse wave-rule moves a wave-front sideways. These two kinds of wave-rules reflect the two strategies for achieving fertilization. Given a schematic induction conclusion of the form $G(\boxed{c(\underline{u})}^{\uparrow}, \lfloor v \rfloor)$ fertilization can be achieved by longitudinally rippling it into $\boxed{c'(\underline{G(u, \lfloor v \rfloor)})}^{\uparrow}$. Such wave-fronts are said to be *beached*. Alternatively, transverse wave-rules can be used to direct wave-fronts towards sinks $G(u, \left\lfloor \boxed{c''(\underline{v})}^{\downarrow} \right\rfloor)$. Such wave-fronts are said to be *sunk*. In general, the sinking of a wave-front may require rippling-in. To achieve this, longitudinal wave-rules can be used in reverse[3]:

$$\boxed{\underline{rev(V)} <> U :: nil}^{\downarrow} \Rightarrow rev(\boxed{U :: \underline{V}}^{\downarrow}) \quad (9)$$

A wave-rule can eliminate wave-fronts completely. Such wave-fronts are said to have *petered-out*. If any of these termination states is reached then the goal is *fully-rippled*. The directionality of wave-fronts is used to prevent looping between these alternative strategies.

[2]We use $\Rightarrow$ to denote rewrites and $\rightarrow$ to denote logical implication.

[3]Note the downward direction of the arrows.

Wave-rules are not restricted to recursive definitions. Non-definitional properties also provide wave-rules, *e.g.*

$$\boxed{\underline{U} <> V}^{\uparrow} <> W \Rightarrow U <> \boxed{V <> \underline{W}}^{\downarrow} \tag{10}$$

The applicability of a wave-rule is determined by matching its *LHS* with a sub-expression of the goal. Note that this matching involves both the object-level term structure as well as the meta-level annotations.

Wave-rules are used to motivate the choice of induction. In the case of (4), wave-rules (7) and (8) provide the motivation for a $u :: v$ induction.

Now consider the rewriting of (5). Using longitudinal wave-rule (7) we get:

$$\boxed{\underline{rev(v)} <> u :: nil}^{\uparrow} <> \lfloor w \rfloor = qrev(\boxed{u :: \underline{v}}^{\uparrow}, \lfloor w \rfloor)$$

An additional precondition to applying transverse wave-rules is that there is a sink at or below the position where the wave-front is being rewritten. This is true for the current goal and transverse wave-rule (8):

$$\boxed{\underline{rev(v)} <> u :: nil}^{\uparrow} <> \lfloor w \rfloor = qrev(v, \left\lfloor \boxed{u :: \underline{w}}^{\downarrow} \right\rfloor) \tag{11}$$

Finally, making use of the transverse wave-rule (10) we get:

$$rev(v) <> \left\lfloor \boxed{u :: nil <> \underline{w}}^{\downarrow} \right\rfloor = qrev(v, \left\lfloor \boxed{u :: \underline{w}}^{\downarrow} \right\rfloor)$$

Note the need to symbolically evaluate the wave-front on the *LHS* using the definition of $<>$. We call this evaluation process *wave-front reduction*:

$$rev(v) <> \left\lfloor \boxed{u :: \underline{w}}^{\downarrow} \right\rfloor = qrev(v, \left\lfloor \boxed{u :: \underline{w}}^{\downarrow} \right\rfloor)$$

Wave-front reductions are always guaranteed to preserve the skeleton term structure. The goal is now fully-rippled so fertilization with the induction hypothesis is possible.

4 Lemma discovery through calculation

We are interested here in seeing how well rippling works when restricted to wave-rules derived from definitions. By eliminating (10), the rewriting presented in §3 breaks down at (11). The wave-front on the *LHS* of (11) is said to be *blocked*. We call the term

$$\ldots \boxed{\underline{rev(v)} <> u :: nil}^{\uparrow} <> \lfloor w \rfloor \ldots$$

the *blockage term*. Note however that the wave-front on the *RHS* of (11) has been sunk. Using the sink term to instantiate the induction hypothesis we can rewrite the *RHS* of the induction conclusion as follows:

$$(rev(v) <> u :: nil) <> w = rev(v) <> u :: w$$

This goal generalizes to:

$$(z <> u :: nil) <> w = z <> u :: w$$

A simple inductive proof establishes this lemma. Note that this lemma gives rise to a specialization of (10), the non-definitional wave-rule we eliminated. This strategy of partially exploiting the induction hypothesis is embodied within NQTHM [BM79] where it is called *cross-fertilization*. An extension to this technique, called *weak fertilization*, is described in [BSvH+91].

A practical limitation of both these strategies is that they discover lemmas in-line. As a consequence the same lemma may be re-discovered many times during the course of a proof. The proof critic idea proposed in [Ire92] provides a framework for overcoming this limitation. We introduce a critic which recognizes the opportunity to exploit the induction hypothesis when rippling gets blocked. We call this process *lemma calculation*:

Definition 1 (Preconditions for lemma calculation) *The preconditions for lemma discovery through the partial use of the induction hypothesis are as follows:*

1. *The principal connective in the goal is equality or implication;*
2. *One side of the induction conclusion is blocked while the other is fully-rippled with respect to the principal connective.*

The effects of this lemma calculation critic is to initiate a subplanning task for the conjectured lemma.

Critics provide a framework for developing more powerful strategies for patching failed proofs. In §5 we present critics for rippling which build upon lemma calculation and incorporate a form of generalization based upon the failure of rippling-in.

5 Lemma discovery through speculation

We now consider the situation where rippling becomes blocked and where there is no immediate way of exploiting the induction hypothesis. For example, consider the conjecture:

$$\forall v, w{:}list(obj).\ \ rev(rev(v <> w)) = rev(rev(v)) <> rev(rev(w)) \qquad (12)$$

In proving (12), wave-rules (6) and (7) motivate a $u :: v$ induction giving rise to a blocked induction conclusion of the form:

$$rev(\boxed{\underline{rev(v <> w)} <> u :: nil}^{\uparrow}) = \\ rev(\boxed{\underline{rev(v)} <> u :: nil}^{\uparrow}) <> rev(rev(w)) \qquad (13)$$

Lemma calculation is not applicable. We have a choice, either we can search for an additional wave-rule or attempt a nested induction. For a nested induction to unblock a wave-front then there must exist a wave-rule such that the blocked wave-front matches with a prefix of the wave-front in the wave-rule. This observation forms the basis for the preconditions of our second lemma discovery critic[4]:

Definition 2 (Preconditions for lemma speculation) *The preconditions for recognizing the need for a lemma instead of a nested induction are as follows:*

1. *There exists at least one most nested blockage term in the goal of the form*[5]*:*

$$\ldots F(\boxed{C(\underline{U})}^{\uparrow}, V)\ldots$$

   where C is not a meta-variable;

2. *For all such blockage terms there exist no wave-rules of form:*

$$F(\boxed{C(C'(\underline{U}))}^{\uparrow}, \boxed{C''(\underline{V})}^{\uparrow}) \Rightarrow \ldots$$

   where C' and C'' may be composite constructors, and either is possibly empty.

The targeting of most nested blockage terms seems well motivated since the unblocking of such terms may in turn unblock a less nested one. In (13) there is no most nested blockage term so either will do. We choose the *RHS* blockage term:

$$\ldots rev(\boxed{\underline{rev(v)} <> u :: nil}^{\uparrow})\ldots$$

In order for precondition (2.2) to succeed we must ensure there are no wave-rules available of the form:

$$rev(\boxed{C'(\underline{V}) <> W}^{\uparrow}) \Rightarrow \ldots$$

Restricting ourself to the definitions of rev and $<>$ then (2.2) succeeds, consequently a nested induction is ruled out. To illustrate a situation where (2.2) fails consider the conjecture:

$$\forall i, j : nat.\, \forall l : list(obj).\; nth(i + j, l) = nth(j, nth(i, l))$$

[4]These preconditions generalize to include rippling-in.

[5]This form generalizes allowing V to denote multiple arguments as well as being empty.

where $nth(n, l)$ returns a list constructed by removing the first n elements of l. The definitions of $+$ and nth provide the following wave-rules:

$$\boxed{s(\underline{X})}^{\uparrow} + Y \;\Rightarrow\; \boxed{s(\underline{X+Y})}^{\uparrow} \tag{14}$$

$$nth(\boxed{s(\underline{X})}^{\uparrow}, \boxed{Y :: \underline{Z}}^{\uparrow}) \;\Rightarrow\; nth(X, Z) \tag{15}$$

Wave-rule (14) motivates a $s(i)$ induction and gives rise to a blocked induction conclusion of the form:

$$nth(\boxed{s(\underline{i+j})}^{\uparrow}, l) = nth(j, nth(\boxed{s(\underline{i})}^{\uparrow}, l)) \tag{16}$$

For precondition (2.2) to succeed there must be no available wave-rules of the form:

$$nth(\boxed{s(C'(\underline{U}))}^{\uparrow}, \boxed{C''(\underline{V})}^{\uparrow}) \Rightarrow \ldots$$

This wave-rule schema, however, unifies with the *LHS* of (15) instantiating C' to be $\lambda x.x$ and C'' to be $\lambda x.Y :: x$. Consequently the lemma speculation critic is not applicable in this case. Indeed, a nested induction on l is precisely what is required to unblock (16).

5.1 Construction of speculative wave-rules

Where a nested induction is ruled out, we require an additional lemma. The approach we take is to speculate the form of the wave-rule which the missing lemma will provide. Such a speculation can be achieved by the use of a higher-order meta-variable. This technique was pioneered by Hesketh[Hes91] where speculative terms are used in controlling the generalization of variables apart and the introduction of sinks. In terms of lemma discovery the meta-level annotations of rippling help constrain the construction of the wave-rule speculation as follows:

- the minimum blockage term defines the *LHS* of the wave-rule speculation;
- the directionality of the blocked wave-front and the existence of a sink are used to suggest the position of the wave-fronts on the *RHS*;
- the skeleton term structure of the minimum blockage term constrains the structure of the *RHS* of the wave-rule speculation;
- generalization of the object-level skeleton term structure is used to simplify the resulting wave-rule speculation.

To illustrate, consider again the blocked goal (13). The most nested blockage term provides the following *LHS* of the wave-rule speculation:

$$rev(\boxed{\underline{rev(V)} <> U :: nil}^{\uparrow}) \Rightarrow \ldots$$

The directionality of the blocked wave-front and the absence of a sink suggests the following positioning of the speculative wave-front on the *RHS*:

$$\ldots \Rightarrow \boxed{F(\underline{\ldots}, \ldots)}^{\uparrow}$$

where F denotes a higher-order meta-variable; a place-holder for the transformed wave-front. Using the skeleton term structure of the minimum blockage term we get:

$$rev(\boxed{\underline{rev(V)} <> U :: nil}^{\uparrow}) \Rightarrow \boxed{F(\underline{rev(rev(V))}, U, V)}^{\uparrow}$$

Finally, generalization completes the construction of the wave-rule speculation:

$$rev(\boxed{\underline{W} <> U :: nil}^{\uparrow}) \Rightarrow \boxed{F(\underline{rev(W)}, U, W)}^{\uparrow} \tag{17}$$

5.2 Guiding the instantiation of speculations

The instantiation of a wave-rule speculation is constrained by rippling and the validity of the instantiation. We use the available wave-rules to constrain the instantiation of the application of the speculation. Candidate instantiations are tested by attempting to verify the lemma which justifies the instantiation of the wave-rule speculation. This verification process involves eliminating non-theorems. We are not concerned here with the details of the process of disproving a conjecture. One possible candidate is Protzen's conjecture disprover [Pro92].

The process of rippling the application of a speculation must take into account the different ways in which rippling can terminate. Firstly, the possibility of a speculative wave-front petering-out must be considered. In terms of higher-order unification [Hue75] the petering-out of a wave-front corresponds to a projection. Wave-front annotations constrain us to consider projections which preserve the skeleton term structure. Note that as a consequence this rules out a projection where the wave-term has multiple wave-holes. Secondly, if a speculative wave-front is fully-rippled with respect to an equality or an implication then the lemma calculation critic can be invoked within an application of the speculation critic. An additional precondition is required, however, which will be discussed in §6. Thirdly, if a speculative wave-front is fully-rippled then rippling has failed to instantiate the wave-rule speculation. A proof of the wave-rule must then be attempted.

Returning to our example, the application of (17) to *RHS* of (13) gives us:

$$\ldots = \boxed{F(\underline{rev(rev(v))}, u, rev(v))}^{\uparrow} <> rev(rev(w))$$

The application is obviously not fully-rippled. Petering-out gives rise to an non-theorem so we proceed by attempting to rewrite the goal using one of the available wave-rules. The only applicable wave-rule is (6) and the associated unification instantiates F to be $\lambda x.\lambda y.\lambda z.y :: x$. Propagating this instantiation through (17) gives rise to the following wave-rule:

$$rev(\boxed{\underline{W} <> U :: nil}^{\uparrow}) \Rightarrow \boxed{U :: \underline{rev(W)}}^{\uparrow} \qquad (18)$$

This non-definitional wave-rule taken together with (6) enables (13) to be rewritten to give:

$$\boxed{u :: \underline{rev(rev(v <> w))}}^{\uparrow} = \boxed{u :: \underline{rev(rev(v)) <> rev(rev(w))}}^{\uparrow}$$

Finally, substitution provides a wave-rule of the form

$$\boxed{U :: \underline{V}}^{\uparrow} = \boxed{U :: \underline{W}}^{\uparrow} \Rightarrow V = W$$

which completes the rippling of the induction conclusion.

The process of speculative rippling takes into account the need for wave-front reductions as mentioned in §3.

6 Combining speculation and calculation

As mentioned in §5.2 the lemma calculation critic requires an additional precondition when invoked within an application of the speculation critic. The revised preconditions are as follows:

Definition 3 (Preconditions for lemma calculation within a speculation) *The preconditions for lemma discovery through the partial use of the induction hypothesis in the context of a speculation are as follows:*

1. *The principal connective in the goal is equality or implication;*
2. *One side of the induction conclusion is blocked while the other is fully-rippled with respect to the principal connective.*
3. *A fully-rippled speculative wave-front exists of the form:*

$$\ldots \left\lfloor \boxed{C(\underline{U}, V)}^{\downarrow} \right\rfloor \ldots$$

and at least one of the available function definitions is of type Utype $\rightarrow$ Vtype $\rightarrow$ Utype, where U is of type Utype and V is of Vtype, and V is optional.

The additional precondition[6] constrains the instantiation of a speculation to the available function definitions.

The refined preconditions for lemma calculation provide the opportunity for a generalization critic. The basic idea is that if precondition (3.3) is false then either a new definition must be discovered or alternatively the conjecture must be modified in such a way that the available function definitions can be used:

Definition 4 (Precondition for speculative sink generalization) *The precondition for speculation sink generalization is as follows:*

1. *For all speculative wave-fronts which have been sunk:*

$$\ldots \left\lfloor \boxed{C(\underline{U}, V)}^{\downarrow} \right\rfloor \ldots$$

there are no available function definitions of type Utype $\rightarrow$ Vtype $\rightarrow$ Utype, where U is of type Utype and V is of Vtype, and V is optional.

The effect of this critic is to generalize the sink occurrences based upon the range types of the available definitions. This will be illustrated in §7.

7 Example revisited

Consider again theorem (1). Using the meta-level annotations of rippling, the associated blocked induction conclusion (2) takes the form:

$$rev(\boxed{(\underline{rev(v)} <> u :: nil)}^{\uparrow} <> \lfloor w \rfloor :: nil) = \lfloor w \rfloor :: \boxed{u :: \underline{v}}^{\uparrow} \tag{19}$$

Both sides of the induction conclusion are blocked so preconditions (2.1) and (2.2) succeed. Two distinct blockage terms are identified so two wave-rule speculations are required. The occurrence of sinks motivates the speculation of transverse wave-rules:

$$\boxed{(\underline{V} <> U :: nil)}^{\uparrow} <> W :: nil \;\Rightarrow\; V <> \boxed{F(\underline{W :: nil}, U, V)}^{\downarrow} \tag{20}$$

$$W :: \boxed{U :: \underline{V}}^{\uparrow} \;\Rightarrow\; \boxed{G(\underline{W}, U, V)}^{\downarrow} :: V \tag{21}$$

[6]This precondition generalizes to deal with beached wave-front speculations as well.

The application of these speculations is carried out independently. This gives us tighter constraints and the possibility of using the lemma calculation strategy to instantiate F and G. Using (20) to rewrite the *LHS* of (19) gives us

$$rev(rev(v) <> \boxed{F(\underline{\lfloor w \rfloor :: nil}, u, v)}^{\downarrow}) \quad = \quad \ldots \tag{22}$$

while the application of (21) to the *RHS* of (19) gives us

$$\ldots \quad = \quad \left\lfloor \boxed{G(\underline{w}, u, v)}^{\downarrow} \right\rfloor :: v \tag{23}$$

In the case of (22), instantiating F to be a projection of its first argument is ruled out because $w :: nil$ is of type $list(obj)$ while the corresponding variable in the induction hypothesis is of type obj. None of the available wave-rules suggest an instantiation for F. Turning to (23), rippling is complete, so preconditions (3.1) and (3.2) of the revised lemma calculation critic succeed. In order for precondition (3.3) to succeed we require a function g of type $obj \rightarrow obj \rightarrow list(obj) \rightarrow obj$, where the second and third arguments of g are optional since they are not part of the skeleton. Precondition (3.3) fails since the only available functions are $<>$ and rev:

$$\begin{array}{lcl} <> & : & list(\_) \rightarrow list(\_) \rightarrow list(\_) \\ rev & : & list(\_) \rightarrow list(\_) \end{array}$$

The sink generalization critic, however, is applicable and using the range types of $<>$ and rev to coerce the sink types, the initial goal is generalized to:

$$\forall v, w{:}list(obj).\ \ rev(rev(v) <> M(w)) = N(w) <> v$$

Note the introduction of additional speculative term structure to reduce the possibility of over generalization. Propagating this generalization through constraint (23) gives us:

$$\ldots = \boxed{G(\underline{N(\lfloor w \rfloor)}, u, v)}^{\downarrow} <> v$$

Here we can exploit wave-rule (9) to complete the ripple:

$$\ldots = rev(\left\lfloor \boxed{u :: \underline{w}}^{\downarrow} \right\rfloor) <> v$$

This rewrite instantiates G to be $\lambda x.\lambda y.\lambda z.x <> y :: nil$ and N to be $\lambda x.rev(x)$. The resulting instantiation of wave-rule speculation (21) takes the form:

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

Note that the lemma which underpins this wave-rule also provides the wave-rule required for the unblocking of the *LHS* of (19):

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

If wave-rule (18) was available then there would have been a choice as to which side of the blockage to speculatively ripple. Using (18) the following alternative generalization is suggested:

$$\forall v, w{:}list(obj).\ \ rev(rev(v) <> rev(w)) = w <> v$$

This example illustrates the close relationship which exists between lemma discovery and generalization.

8 Implementation

The ideas of proof plans are implemented within the CLAM [vISN92] planning system. An implementation of proof critics, as presented here, is currently underway. λ-prolog [MN88] is being used to prototype the ideas. The initial work has focused upon controlling the selection of appropriate higher-order unifiers. Our strategy is to select the most general unifiers which preserve skeleton term structure. This strategy is successful for the example theorems and associated lemmas documented in tables 1 and 2 respectively.

1	$\forall x{:}nat.\ even(x + x)$
2	$\forall x, y{:}nat.\ ((x + y) - x) = ((y + x) - x)$
3	$\forall x, y{:}list(obj).\ rev(rev(x <> y)) = rev(rev(x)) <> rev(rev(y))$
4	$\forall x, y{:}list(obj).\ rotate(length(x), x <> y) = y <> x$
5	$\forall x{:}list(obj).\ \forall y{:}obj.\ rev(rev(x) <> y :: nil) = y :: x$

Table 1: Example theorems

Theorem	Lemmas	Calculation	Speculation	Generalization
1	$X + s(X) = s(X + X)$		x	
2	$s(X) - Y = s(X - Y)$	x		
	$X + s(Y) = s(X + Y)$		x	
3	$rev(V <> U :: nil) = U :: rev(V)$		x	
4	$(U <> V) <> W = U <> (V <> W)$		x	
	$W <> U :: V = (W <> U :: nil) <> V$	x		
5	$W <> U :: V = (W <> U :: nil) <> V$		x	x

Table 2: Lemma discovery examples

9 Related work

The rippling proof method is a rational reconstruction and extension of the heuristics implemented within NQTHM [BM79]. The proof critics presented here will enable CLAM to find proofs where the cross-fertilization and generalization heuristics of NQTHM breakdown.

Hutter [Hut90] uses first-order schemas to "guess" missing lemmas when the rewriting of an induction conclusion becomes blocked. Our use of higher-order meta-variables, although harder to control, is obviously more powerful.

Finally, superposition is often cited as a technique for lemma discovery within the induction completion community. Indeed the Huet-Hullot [HH82] completion procedure is able to prove (1). However, the application of superposition to generating auxiliary lemmas is rare, in general, and heuristics for its application are not as principled as those of rippling.

10 Conclusion

The potential for making productive use of failed proof attempts is greatly enhanced if proof strategies are expressed at a high-level. Proof plans provide such a framework for expressing proof strategies and we have presented a general method for exploiting this potential. In particular, we have illustrated how failure can be used to suggest lemmas and generalizations in the context of inductive proof.

Acknowledgements

We thank David Basin for his encouragement and constructive feedback. Toby Walsh provided useful comments on a early draft of this paper. Thanks also go to Jane Hesketh for useful discussions during the early stages of this work.

References

[BM79] R.S. Boyer and J.S. Moore. *A Computational Logic.* Academic Press, 1979. ACM monograph series.

[BSvH^{+}91] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. Research Paper 567, Dept. of Artificial Intelligence, Edinburgh, 1991. To appear in Artificial Intelligence.

[Bun88] A. Bundy. The use of explicit plans to guide inductive proofs. Research Paper 349, Dept. of Artificial Intelligence, Edinburgh, 1988. Short version published in the proceedings of CADE-9.

[BvHSI90] 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.

[GMW79] M.J. Gordon, A.J. Milner, and C.P. Wadsworth. *Edinburgh LCF - A mechanised logic of computation*, volume 78 of *Lecture Notes in Computer Science*. Springer Verlag, 1979.

[Hes91] J.T. Hesketh. *Using Middle-Out Reasoning to Guide Inductive Theorem Proving*. PhD thesis, University of Edinburgh, 1991.

[HH82] G. Huet and J. Hullot. Proofs by induction in equational theories with constructors. *Journal of the Association for Computing Machinery*, 25(2), 1982.

[Hue75] G. Huet. A unification algorithm for lambda calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Hut90] 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.

[Ire92] A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proofs. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning – LPAR 92, St. Petersburg*, Lecture Notes in Artificial Intelligence No. 624, pages 178–189. Springer-Verlag, 1992. Also available from Edinburgh as DAI Research Paper 592.

[MN88] D. Miller and G. Nadathur. An overview of λProlog. In R. Bowen, K. & Kowalski, editor, *Proceedings of the Fifth International Logic Programming Conference/ Fifth Symposium on Logic Programming.* MIT Press, 1988.

[Pro92] M. Protzen. Disproving conjectures. In D. Kapur, editor, *11th Conference on Automated Deduction*, pages 340–354, Saratoga Springs, NY, USA, June 1992. Published as Springer Lecture Notes in Artificial Intelligence, No 607.

[vISN92] F. van Harmelen, A. Ireland, A. Stevens, and S. Negrete. The CLAM proof planner, user manual and programmer manual *(version 1.5)*. Technical paper in preparation, DAI, 1992.