# A Divergence Critic

Toby Walsh

INRIA-Lorraine
615, rue du Jardin Botanique, B.P. 101
F-54602 Villers-les-Nancy, France
walsh@loria.fr

**Abstract.** Inductive theorem provers often diverge. This paper describes a critic which monitors the construction of inductive proofs attempting to identify diverging proof attempts. The critic proposes lemmas and generalizations which hopefully allow the proof to go through without divergence. The critic enables the system SPIKE to prove many theorems completely automatically from the definitions alone.

## 1 Introduction

Rippling is a powerful heuristic developed at Edinburgh for proving theorems involving explicit induction [6]. The essential idea behind rippling is to remove the "difference" between the induction conclusion and the induction hypothesis using a very goal directed form of rewriting. When augmented with a "difference matching" procedure to identify such differences, rippling has also proved useful in domains outside of explicit induction. For example, it has been used to sum series, to prove limit theorems, and to perform normalization [3, 15, 16]. In this paper, I describe an experiment to apply rippling and difference matching to a new domain, the problem of overcoming the divergence of a prover using implicit induction. The experiment is very successful. A critic has been implemented which is often able to identify diverging proof attempts and to propose lemmas and generalizations which will allow the proof to go through successfully. Although the critic is designed to work with SPIKE, a theorem prover which uses implicit induction [4], the critic should also work with other implicit and explicit induction provers.

Induction in SPIKE is performed by means of test sets. A test set is essentially a finite description of the initial model. It is a more powerful concept than the related notion of cover set since, in combination with a ground convergent rewrite system, a test set can be used to refute false conjectures. The basic idea in SPIKE is to instantiate induction variables in the conjecture to be proved with the members of the test set (using the `generate` rule), and then to use rewriting to simplify the resulting expressions (using the `simplify` rules). This rewriting uses any of the axioms, lemmas and induction hypotheses provided they are smaller (with respect to a well-founded relation) than the current conjecture. Although SPIKE has proved several challenging theorems without assistance (*eg.* the binomial theorem), its attempts to prove many theorems diverge without an appropriate generalization or the addition of a suitable lemma. The aim of the

critic described in this paper is to identify when a proof attempt is diverging, and to speculate a lemma or generalization which would prevent this divergence. The identification of divergence is described in more detail in the next two sections, and the speculation of an appropriate generalization or lemma is described in the following three sections.

## 2 Divergence Analysis

Several properties of rewrite rules have been identified which give rise to divergence (*eg.* forwards and backwards crossed systems [8]). However, these properties fail to capture all diverging rewrite systems since the problem is, in general, undecidable. The divergence critic proposed here studies just the proof attempt looking for patterns of divergence; no attempt is made to analyse the rewrite rules themselves for structures which give rise to divergence. The advantage of this approach is that the critic need not know the details of the rewrite rules applied, nor the type of induction being performed, nor the control structure used by the prover. The critic can thus recognise divergence patterns arising from complex mutual or multiple inductions with little more difficulty than divergence patterns arising from simple straightforward inductions. The disadvantage of this approach is that the critic will sometimes identify a "divergence" pattern when none exists. Fortunately, such cases appear to be rare, and even when they occur, the critic usually suggests a lemma or generalization which gives a shorter and more elegant proof.

The divergence critic attempts to find term structure introduced by induction which is accumulating in an equation and which is preventing simplification. To do this, the critic partitions the sequence of diverging equations generated by SPIKE. This is necessary since several diverging sequences can be interleaved in the prover's output. Several heuristics are used to reduce the number of partitions considered. The main heuristic is parentage; that is, the critic partitions the sequence so that each equation in a partition is derived from the previous one. Other heuristics which can be used include: the function and constant symbols which occur in one equation occur in the next equation in the partition, and the weights of the equations in a partition form a simple arithmetic progression. The critic then attempts to find term structure introduced by induction which is accumulating at some position in the equation and which is causing divergence. To identify such accumulating term structure, the critic uses the difference matching procedure introduced in [2]. This is explained in more detail in the next section. To fix divergence, the critic then speculates a lemma or generalization of the theorem which moves this accumulating term structure out of the way.

## 3 An Example

To illustrate the essential ideas behind the critic's divergence analysis, consider SPIKE's attempt to prove the length-append theorem.

$$len(app(a, b)) = len(app(b, a))$$

SPIKE begins by applying the `generate` rule. This instantiates the induction variables with members of the test set, $\{nil, cons(h, t)\}$. Up to variable renaming, this gives 3 distinct equations, which are rewritten by the `simplify` rules using the definitions of *len* and *app* to give a simple identity and the following 2 equations,

$$\begin{aligned} len(app(b, nil)) &= s(len(b)) \\ len(app(a, cons(d, b))) &= len(app(b, cons(c, a))) \end{aligned}$$

Since no further simplification can be made, SPIKE performs another induction by applying the `generate` rule. After simplification, this gives,

$$\begin{aligned} len(app(b, cons(c, nil))) &= s(s(len(b))) \\ len(app(a, cons(f, cons(d, b)))) &= len(app(b, cons(e, cons(c, a)))) \end{aligned}$$

No further simplification can be performed so SPIKE again applies the `generate` rule. Unfortunately, the proof attempt will continue to diverge like this ad infinitum.

Using the parentage heuristic, the divergence critic partitions the equations produced into two sequences. The first sequence is,

$$\begin{aligned} len(app(a, b)) &= len(app(b, a)) \\ len(app(a, cons(d, b))) &= len(app(b, cons(c, a))) \\ len(app(a, cons(f, cons(d, b)))) &= len(app(b, cons(e, cons(c, a)))) \\ &\vdots \end{aligned}$$

The second sequence is,

$$\begin{aligned} len(app(b, nil)) &= s(len(b)) \\ len(app(b, cons(c, nil))) &= s(s(len(b))) \\ len(app(b, cons(d, cons(c, nil)))) &= s(s(s(len(b)))) \\ &\vdots \end{aligned}$$

The critic then attempts to find the accumulating term structure in each sequence which is causing divergence. In this case, in both sequences, cons functions are accumulating on the second argument of append. Since append is defined recursively on its first argument, SPIKE is unable to simplify such terms, and the proof diverges. To identify this accumulating term structure, the critic uses difference matching. This procedure annotates terms with wavefronts, boxes with holes which mark where the terms differ. For example, taking the first sequence, difference matching successive equations gives the annotated sequence,

$$\begin{aligned} len(app(a, b)) &= len(app(b, a)) \\ len(app(a, \boxed{cons(d, \underline{b})})) &= len(app(b, \boxed{cons(c, \underline{a})})) \\ len(app(a, \boxed{cons(f, \underline{cons(d, b)})})) &= len(app(b, \boxed{cons(e, \underline{cons(c, a)})})) \end{aligned}$$

An annotation consists of a **wavefront**, a box, with a **wavehole**, an underlined term. The **skeleton** is formed by deleting everything that appears in the wavefront but not in the wavehole. The **erasure** of an annotated terms is formed by deleting the annotations but not the terms they contain. In the above sequence, the skeleton of every annotated equation is identical to the erasure of the previous equation in the sequence. Difference matching guarantees this; that is, difference matching $s$ with $t$ annotates $s$ so that its skeleton matches $t$. More formally, $s'$ is a **difference match** of $s$ with $t$ with substitution $\sigma$ iff $\sigma(skeleton(s')) = t$ and $erase(s') = s$ where $skeleton(s')$ and $erase(s')$ build the skeleton and erasure of the annotated term $s'$.

Difference matching successive equations in the second sequence gives,

$$\begin{aligned} len(app(b, nil)) &= s(len(b)) \\ len(app(b, \boxed{cons(c, \underline{nil})})) &= \boxed{s(\underline{s(len(b))})} \\ len(app(b, \boxed{cons(d, \underline{cons(c, nil)})})) &= \boxed{s(\underline{s(s(len(b)))})} \end{aligned}$$

Note that similar term structure is accumulating on the lefthand side of this sequence as in the first sequence.

The critic then tries to speculate a lemma which moves the accumulating and nested term structure out of the way. In this case, the critic speculates a rule for moving a cons off the second argument of append. That is, the lemma,

$$len(app(a, cons(d, b))) = s(len(app(a, b)))$$

With this lemma, SPIKE is able to prove the len-app theorem without divergence. In addition, this lemma is sufficiently simple that SPIKE can prove it without assistance. The heuristics used by the critic to perform this lemma speculation are described in more detail in the next two sections.

The divergence analysis performed by the critic can be summarised as follows:

1. There is a sequence of equations $s_i = t_i$ to which the `generate` rule is applied ($i = 0, 1 \ldots$);
2. There exists (non trivial) $G, H$ such that for each $j$, difference matching gives $s_j = G(U_j)$, and $s_{j+1} = G(\boxed{H(\underline{U_j})})$.

**Preconditions to the divergence critic.**

By "non-trivial" I wish to exclude unhelpful answers like $\lambda x.x$. $H$ is thus the accumulating and nested term structure. For the first sequence of equations in the len-app example, $H$ was $\lambda x.cons(y, x)$, $G$ was $\lambda x.app(a, x)$, and $U_0$ was $b$. For simplicity, I have ignored the orientation of equations. In addition, the preconditions can be easily generalised to include multiple and nested annotations. This allows the critic to recognise multiple sources of divergence in the same equation. Techniques like those of [7] which identify accumulating term structure by

most specific generalization cannot cope with divergence patterns that give rise to nested annotations.

The preconditions above leave the length of sequence undefined. If the sequence is of length 2, then the critic can be thought of as preemptive. That is, it will propose a lemma just before another induction is attempted and divergence begins. However, using such a short sequence risks identifying divergence when none exists. On the other hand using a long sequence is expensive to test and allows the prover to waste time on diverging proof attempts. Empirically, a good compromise appears to be to look for sequences of length 3. This is both cheap to test and reliable. To identify accumulating term structure, it also appears to be sufficient to use ground difference matching with alpha conversion of variable names. There exists a fast polynomial algorithm to perform such difference matching based upon the ground difference matching algorithm given in [3].

## 4 Lemma Speculation

One way of removing the accumulating and nested term structure is to apply a lemma, called a **wave rule**, which moves this difference to the top of the term leaving the skeleton unchanged. The hope is that the prover will then be able to cancel the difference with a similar difference on the other side of the equality.

For the len-app theorem, the divergence pattern suggests a lemma of the form,

$$len(app(a, \boxed{cons(d, \underline{b})})) = \boxed{F(\underline{len(app(a, b))})}$$

The only problem is to determine a suitable instantiation for $F$. Two heuristics are used for this: **cancellation** and **petering out**. The cancellation heuristic uses difference matching to identify term structure accumulating on the opposite side of the sequence which would allow cancellation to occur; failing that, it looks for suitable term structure to cancel against in a new sequence (the original sequence is usually a divergence pattern of a step case, whilst the new sequence is usually a divergence pattern of a base case). In the len-app example, the successor functions accumulating at the top of the righthand side of the second sequence of equations suggests that $F$ be instantiated to $\lambda x.s(x)$. Thus, as required, the cancellation heuristic suggests the lemma,

$$len(app(a, \boxed{cons(d, \underline{b})})) = \boxed{s(\underline{len(app(a, b))})}$$

The other heuristic used is petering out. In moving the differences up to the top of the term, they may disappear altogether. The petering out heuristic uses regular matching to identify such situations. Petering out occurs, for example, in the speculation of the lemma,

$$sorted(\boxed{insert(y, \underline{x})}) = sorted(x)$$

This lemma is proposed in the analysis of the diverging proof attempt of the theorem $sorted(isort(x))$ where $isort$ is insertion sort and $insert(y,x)$ inserts the

element $y$ into the list $x$ in order. In the case of petering out, $F$ is instantiated to the identity function $\lambda x.x$.

All lemmas speculated are filtered through a conjecture disprover. When a confluent set of rewrite rules exists for ground terms, exhaustive normalization of some represenative set of ground instances of the equations is used to filter out non-theorems. For more sophisticated techniques for disproving conjectures see [12]. Alternatively, SPIKE itself could be used to filter out non-theorems.

The critic's lemma speculation can be summarized as follows (using the same variable names as the preconditions):

1. The critic proposes a lemma of the form,

$$G(\boxed{H(\underline{U_0})}) = \boxed{F(\underline{G(U_0)})}$$

2. $F$ is instantiated by the cancellation or petering out heuristics;
3. Lemmas are filtered through a conjecture disprover;
4. If several lemmas are suggested, the critic deletes any that are subsumed.

**Postconditions for the divergence critic.**

As before, this definition can be easily extended to deal with multiple and nested wavefronts. Note that as the lemma proposed moves the wavefronts to top of the term, it usually only introduces fresh divergence in the rare cases that cancellation or fertilization fails. This is unlikely since the cancellation and petering out heuristics attempt to ensure that cancellation or fertilization can take place.

## 5 Generalization

One major cause of divergence is the need to generalize. Although any lemma proposed by the critic is usually sufficient to fix divergence, attempting to prove the lemma itself can cause fresh divergence. In addition, several speculated lemmas can often be replaced by a single generalization, and a generalized lemma frequently leads to a shorter, more elegant and natural proof. The critic therefore attempts to generalize the lemma speculated, using the conjecture disprover to guard against over-generalization.

The main heuristic used for generalization is an extension of the primary term heuristic of Aubin [1]. The **primary terms** are those terms encountered as a term is explored from the root to the leaves ignoring non-recursive argument positions to functions. Consider, for example, the theorem,

$$len(app(a, b)) = len(a) + len(b)$$

where $+$ is defined recursively on its second argument. The primary terms of the righthand side are $\{len(a) + len(b), len(b), b\}$.

Analysis of SPIKE's diverging attempt to prove this theorem suggests the lemma,

$$\boxed{s(\underline{len(a)})} + len(b) = \boxed{s(\underline{len(a) + len(b)})}$$

A set of candidate terms for generalization is constructed by computing the intersection of the primary terms of the two sides of the equation. In this case, the intersection of the primary terms is $\{len(b), b\}$. The critic then picks members of this set to generalize to new variables. Picking $b$ justs gives an equivalent lemma up to renaming of variables. Picking $len(b)$ gives the generalization,

$$\boxed{s(\underline{len(a)})} + y = \boxed{s(\underline{len(a) + y})}$$

The reason for considering just primary terms is that the recursive definitions typically provide wave rules for removing differences which accumulate at these positions. In addition to primary terms, the divergence critic therefore also considers the positions of the waveholes in the lemma being speculated. The motivation for this extension is that the speculated lemma will allow differences to be moved from the wavehole positions; such positions are therefore also candidates for generalization.

For instance, + in the above example is considered to be "recursive" on both the first and second arguments since + is recursively defined on its second argument and the lemma being speculated has a wavehole on the first argument of +. The terms $\{len(a), a\}$ are therefore also included in the intersection set of candidate terms for generalization. Picking $a$ to generalize gives, as before, an equivalent lemma up to renaming. Picking $len(a)$ gives the generalization,

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

The speculated lemma is now as general as is possible. This lemma allows the proof to go through without divergence.

The critic also has a heuristic for merging speculated lemmas. For instance, with the theorem $sorted(isort(x))$, the critic's divergence analysis and lemma speculation actually suggest the lemmas,

$$sorted(\boxed{insert(0, \underline{x})}) = sorted(x)$$
$$sorted(\boxed{insert(s(y), \underline{x})}) = sorted(x)$$

The critic identifies that $\{0, s(y)\}$ is a cover set for the natural numbers and merges these two lemma to the give the generalization,

$$sorted(\boxed{insert(y, \underline{x})}) = sorted(x)$$

## 6 Transverse Wave Rules

The lemmas speculated so far have moved differences directly to the top of the term where they are removed by cancellation or petering out. An alternative

way of removing a difference is to move the difference onto another argument position where: either it can be removed by matching with a "sink", a universally quantified variable in the induction hypothesis; or it can be moved upwards by rewriting with the recursive definitions. Theorems involving functions with accumulators provide a rich source of examples where such lemmas prevent divergence.

Consider, for example, the theorem,

$$qrev(a, b) = app(rev(a), b)$$

where *rev* is naive list reversal and *qrev* is tail recursive list reversal which builds the reversed list on its second accumulator argument. That is,

$$\begin{aligned} rev(nil) &= nil \\ rev(cons(h, t)) &= app(rev(t), cons(h, nil)) \\ qrev(nil, r) &= r \\ qrev(cons(h, t), r) &= qrev(t, cons(h, x)) \end{aligned}$$

SPIKE's attempt to prove this theorem diverges generating the following sequence of equations to which the `generate` rule are applied,

$$\begin{aligned} qrev(a, b) &= app(rev(a), b) \\ qrev(a, \boxed{cons(c, \underline{b})}) &= app(\boxed{app(\underline{rev(a)}, cons(c, nil))}, b) \\ qrev(a, \boxed{cons(c, \underline{cons(d, b)})}) &= app(\boxed{app(\underline{app(rev(a), cons(c, nil))}, cons(d, nil))}, b) \\ &\vdots \end{aligned}$$

Divergence analysis of the righthand side of these equations identifies some accumulating term structure. Rather than move this term structure to the top of the term, it is much simpler to move it onto the second argument of the outermost append. The critic therefore proposes a **transverse** wave rule, which preserves the skeleton but moves the difference onto a different argument position. In this example, this is a lemma of the form,

$$app(\boxed{app(\underline{rev(a)}, cons(c, nil))}, b) = app(rev(a), \boxed{F(\underline{b})})$$

In moving the difference onto another argument position, the difference may change syntactically. The righthand side of the lemma is therefore only partially determined. To instantiate $F$, the critic uses two heuristics: **fertilization** and **simplification**. The fertilization heuristic uses matching to find an instantiation for $F$ which enables immediate fertilization. In this case, matching against the induction hypothesis suggests,

$$app(\boxed{app(\underline{rev(a)}, cons(c, nil))}, b) = app(rev(a), \boxed{cons(c, \underline{b})})$$

The simplification heuristic uses matching to find an instantiation for $F$ which enables the term to be simplified using one of the recursive definitions.

Finally the critic generalizes the lemma using the same primary term heuristic as before (augmenting recursive positions with wavehole positions). This gives the lemma,

$$app(\boxed{app(\underline{a}, cons(c, nil))}, b) = app(a, \boxed{cons(c, \underline{b})})$$

This is exactly the lemma needed by SPIKE to complete the proof. In addition, the lemma is simple enough to be proved by itself without divergence; this is not true of the ungeneralized lemma.

The actions of the critic can be summarized as follows,

Preconditions:

1. There is a sequence of equations $s_i = t_i$ to which the `generate` rule is applied ($i = 0, 1 \ldots$);
2. There exists (non trivial) $G, H$ such that for each $j$, difference matching gives $s_j = G(U_j, Acc)$ and $s_{j+1} = G(\boxed{H(\underline{U_j})}, Acc)$.

Postconditions:

1. The critic proposes a lemma of the form,

$$G(\boxed{H(\underline{U_0})}, Acc) = G(U_0, \boxed{F(\underline{Acc})})$$

2. $F$ is instantiated by the fertilization or simplification heuristics;
3. The lemma is generalized as much as possible;
4. Generalized lemmas are filtered through a conjecture disprover;
5. If several lemmas are suggested, the critic deletes any that are subsumed.

**Speculation of transverse wave rules.**

The preconditions and postconditions can be easily generalised to include multiple and nested annotations. The critic also uses an additional cancellation heuristic to generalize transverse wave rules. This heuristic attempts to cancel equal outermost functors where possible. For example, consider the theorem,

$$half(x + x) = x$$

From SPIKE's diverging proof attempt, the critic suggests the lemma,

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

The cancellation heuristic deletes the equal outermost function. This gives the more general lemma,

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

## 7 Results

The critic described in the previous sections has been implemented in Prolog. It is successful at identifying divergence and proposing appropriate lemmas and generalizations for a large number of theorems. A few examples are given in Table 1. The times given are in seconds for the average of 10 runs on a Sun 4 running Quintus 3.1.1. For brevity, :: is written for infix cons, <> for infix append, [] for the empty list nil, and $[x]$ for the list cons($x$,nil). In addition, + is defined recursively on its second argument, even is defined by a s(s(x)) recursion, $\text{even}_m$ is defined by a mutual recursion with $\text{odd}_m$, and $rot(n, l)$ rotates a list $l$ by $n$ elements.

SPIKE's proof attempt diverges on each example when given the definitions alone. In each of the 25 cases, however, the critic is quickly able to suggest a lemma which overcomes divergence. When multiple lemmas are proposed (with the exception of 15) any one on its own is sufficient to fix divergence. In every case (except 9 and 19) the lemmas proposed are sufficiently simple to be proved automatically without introducing fresh divergence. In many cases, the lemmas proposed are optimal; that is, they are the simplest possible lemmas which fix divergence. In the cases when the lemma is not optimal, it is close to optimal.

Example 2 is a simple program verification problem taken from [7]. The second lemmas proposed in examples 3 and 5 are somewhat surprising; they are nevertheless just as good at fixing divergence as the first lemmas. Examples 7 and 8 demonstrate that the critic can cope with divergence in theories involving mutual recursion. In example 9, the proposed lemma is too difficult to be proved automatically. However, the divergence critic is able to identify the cause of this difficulty and propose a lemma which allows the proof to go through (example 11). In example 15, the critic identifies two separate divergence patterns. To overcome divergence, the first lemma plus one or other of the second and third are therefore needed. Example 19 is the only disappointment; the lemma proposed fixes divergence but is too difficult to be proved automatically, even with the assistance of the divergence critic. The problem seems to be that the example needs the introduction of a derived function like append which does not occur in the specification of the theorem. Examples 24 and 25 demonstrate that the critic can cope with divergence in theories containing conditional equations.

Divergence analysis is very quick in each case. The divergence pattern is recognized usually in less than a second. Most of the time is spent looking for generalizations and refuting over-generalizations using the conjecture disprover. Indeed, in the slowest example, less than 1% of the time is spent performing divergence analysis. Additional heuristics for preventing over-generalization and a more efficient implementation of the conjecture disprover would thus speed up the slower examples considerably.

The results are very pleasing. Using the divergence critic, the theorems listed (with the exception of 19) can all be proved from the definitions alone. For comparison, the NQTHM system [5] (perhaps the best known explicit induction theorem prover) when given just the definitions was unable to prove more

| No | Theorem | Lemmas speculated | Time/s |
|---|---|---|---|
| 1 | s(x)+x=s(x+x) | s(x)+y=s(x+y)<br>s(x)+y=x+s(y) | 7.8 |
| 2 | dbl(x)=x+x ↔<br>dbl(0)=0, dbl(s(x))=s(s(dbl(x))) | s(x)+y=s(x+y)<br>s(x)+y=x+s(y) | 8.2 |
| 3 | len(x <> y)=len(y <> x) | len(x <> z :: y)=s(len(x <> y))<br>len(x <> z :: y)=len(w :: x <> y) | 3.6 |
| 4 | len(x <> y)=len(x)+len(y) | s(x)+y=s(x+y), s(x)+y=x+s(y) | 7.2 |
| 5 | len(x <> x)=dbl(len(x)) | len(x <> z :: y)=s(len(x <> y))<br>len(x <> w :: z :: y)=s(len(x <> w :: y)) | 11.6 |
| 6 | even(x+x) | even(s(s(x))+y)=even(x+y) | 5.4 |
| 7 | $even_m$(x+x) | $even_m$(s(s(x))+y)=$even_m$(x+y)<br>$odd_m$(s(s(x))+y)=$odd_m$(x+y) | 28.4 |
| 8 | $even_m$(x) → half(x)+half(x)=x | s(x)+y=s(x+y), s(x)+y=x+s(y) | 6.0 |
| 9 | rot(len(x),x)=x | rot(len(x),x <> [y])=y :: rot(len(x),x) | 2.4 |
| 10 | len(rot(len(x),x))=len(x) | len(rot(x,z <> [y]))=s(len(rot(x,z))) | 4.8 |
| 11 | rot(len(x),x <> [y])=y :: rot(len(x),x) | (x <> [y]) <> z=x <> y :: z<br>rot(len(x),(x <> [y]) <> z)=y :: rot(len(x),x <> z) | 86.3 |
| 12 | len(rev(x))=len(x) | len(x <> [y])=s(len(x)) | 2.0 |
| 13 | rev(rev(x))=x | rev(x <> [y])=y :: rev(x) | 1.2 |
| 14 | rev(rev(x) <> [y])=y :: x | rev(x <> [y])=y :: rev(x) | 16.0 |
| 15 | len(rev(x <> y))=len(x)+len(y) | len(x <> [y])=s(len(x))<br>s(x)+y=s(x+y), s(x)+y=s(x+y) | 10.0 |
| 16 | len(qrev(x,[]))=len(x) | len(qrev(x,z :: y))=s(len(qrev(x,y))) | 2.2 |
| 17 | qrev(x,y)=rev(x) <> y | (x <> [y]) <> z=x <> y :: z | 3.4 |
| 18 | len(qrev(x,y))=len(x)+len(y) | s(x)+y=s(x+y), s(x)+y=x+s(y) | 12.0 |
| 19 | qrev(qrev(x,[]),[])=x | qrev(qrev(x,[y]),z)=y :: qrev(qrev(x,[]),z) | 5.0 |
| 20 | rev(qrev(x,[]))=x | rev(qrev(x,[y]))=y :: rev(qrev(x,[])) | 5.8 |
| 21 | qrev(rev(x),[])=x | qrev(x <> [y],z)=y :: qrev(x,z) | 5.2 |
| 22 | nth(i,nth(j,x))=nth(j,nth(i,x)) | nth(s(i),nth(j,y :: x))=nth(i,nth(j,x)) | 7.4 |
| 23 | nth(i,nth(j,nth(k,x)))=nth(k,nth(j,nth(i,x))) | nth(s(i),nth(j,y :: x))=nth(i,nth(j,x)) | 7.6 |
| 24 | len(isort(x))=len(x) | len(insert(y,x))=s(len(x)) | 2.0 |
| 25 | sorted(isort(x)) | sorted(insert(y,x))=sorted(x)<br>sorted(insert(y,insert(z,x)))=sorted(x) | 114 |

**Table 1.** Some lemmas speculated by the Divergence Critic.

than half these theorems.[1] Of course, with the addition of some simple lemmas, NQTHM is able to prove all these theorems. Indeed, in many cases, NQTHM needs the same lemmas as those proposed by the divergence critic and required by SPIKE. This suggests that the divergence critic is not especially tied to the particular prover used nor even to the implicit induction setting. To test this hypothesis, I presented the output of a diverging proof attempt from NQTHM to the critic. I chose the commutativity of times as this is perhaps the simplest theorem which causes NQTHM to diverge. The critic proposed the lemma,

```
(EQUAL (TIMES Y (ADD1 X)) (PLUS Y (TIMES Y X))))
```

where `TIMES` and `PLUS` are primitives of NQTHM's logic recursively defined on their first arguments. This is exactly the lemma needed by NQTHM to prove the commutativity of times.

## 8 Related Work

Critics for monitoring the construction of proofs were first proposed in [9] for "proof planning". In this framework, failure of one of the proof methods automatically invokes a critic. Various critics for explicit induction have been developed that speculate missing lemmas, perform generalizations, look for suitable case splits, *etc* using heuristics based upon rippling similar to the ones described here [10]. There are, however, several significant differences. First, the divergence critic described here works in an implicit (and not an explicit) induction setting. Second, the divergence critic is not automatically invoked but must identify when the proof is failing. Third, the divergence critic is less specialized. These last two differences reflect the fact that critics in proof planning are usually associated with the failure of a particular precondition to a heuristic. The same divergence pattern can, by comparison, arise for many different reasons: the need to generalize variables apart, to generalize common subterms, to add a lemma, *etc.* Fourth, the divergence critic must use difference matching to annotate terms; in proof planning, terms are often already appropriately annotated. Finally, the divergence critic is less tightly coupled to the the theorem prover's heuristics. The critic can therefore exploit the strengths of the prover without needing to reason about the complex heuristics being used. For instance, the divergence critic has no difficulty identifying divergence in complex situations like nested or mutual inductions.

Divergence has been studied quite extensively in completion procedures. Two of the main novelties of the critic described here are the use of difference matching to identify divergence, and the use of rippling in the speculation of lemmas to overcome divergence. Dershowitz and Pinchover, by comparison, use most specific generalization to identify divergence patterns in the critical pairs produced by completion [7]. Kirchner uses generalization modulo an equivalence relation to recognise such divergence patterns [11]; meta-rules are then synthesized to describe infinite families of rules with some common structure. Thomas and Jantke

[1] To be precise, NQTHM failed on 5, 6, 7, 9, 10, 11, 14, 16, 17, 19, 20, 21, 22, and 23.

use generalization and inductive inference to recognize divergence patterns and to replace infinite sequences of critical pairs by a finite number of generalizations [13]. Thomas and Watson use generalization to replace an infinite set of rules by a finite complete set with an enriched signature [14].

Generalization modulo an equivalence enables complex divergence patterns to be identified. However, it is in general undecidable. Most specific generalization, by comparison, is more limited. It cannot recognize divergence patterns which give nested wavefronts like,

$$\boxed{s^n(\boxed{s^n(\underline{x})} + x)}.$$

In addition, most specific generalization cannot identify term structure in wave-holes. For example, consider the divergence pattern of example (20),

$$\begin{aligned} rev(qrev(x, nil)) &= x \\ rev(qrev(x, \boxed{cons(y, \underline{nil})})) &= \boxed{cons(y, \underline{x})} \\ rev(qrev(x, \boxed{cons(z, \underline{cons(y, nil)})})) &= \boxed{cons(z, \underline{cons(y, x)})} \\ &\vdots \end{aligned}$$

Most specific generalization of the lefthand side of this sequence gives the term $rev(qrev(x, z))$ (or, ignoring the first term in the sequence, $rev(qrev(x, cons(y, z)))$). Most specific generalization cannot, however, identify the more useful pattern, $rev(qrev(x, cons(y, nil))))$ which suggests the simpler lemma,

$$rev(qrev(x, cons(y, nil))) = cons(y, rev(qrev(x, nil)))$$

## 9 Future Work

There are many other types of divergence which could be incorporated into the divergence critic. Further research is needed to identify such divergence patterns, isolate their causes and propose ways of fixing them. This research may take advantage of the close links between divergence patterns and particular types of generalization. For instance, I am currently attempting to incorportate two other divergence patterns into the critic which arise when variables need to be renamed apart, and common subterms generalized in the theorem being proved. (Note that the current heuristics already rename variables apart and generalize common subterms in the speculated lemma.)

To illustrate a divergence pattern associated with the need to rename variables apart, consider the theorem,

$$app(x, app(x, x)) = app(app(x, x), x)$$

SPIKE's attempt to prove this theorem diverges giving the equations,

$$app(x, app(x, x)) = app(app(x, x), x)$$

$$app(x, \boxed{cons(y, \underline{app(x, \boxed{cons(y, \underline{x})})})}) = app(app(x, \boxed{cons(y, \underline{x})}), \boxed{cons(y, \underline{x})})$$

$$\vdots$$

As before difference matching identifies the accumulating term structure. However, rather than try to speculate a lemma to move this difference out of the way, the critic generalizes the original theorem so that the difference is not introduced in the first place. In this case, it is sufficient to rename the first variable apart,

$$app(z, app(x, x)) = app(app(z, x), x)$$

SPIKE is able to prove this theorem without difficulty. The task of divergence analysis here is to isolate the variables to be renamed apart.

To illustrate a divergence pattern associated with the need to generalize common subterms consider the theorem,

$$len(rev(rev(x)) = len(rev(x))$$

SPIKE's attempt to prove this theorem diverges giving the equations,

$$len(rev(rev(x)) = len(rev(x))$$

$$len(rev(\boxed{app(\underline{rev(x)}, cons(y, nil))})) = len(\boxed{app(\underline{rev(x)}, cons(y, nil))})$$

$$\vdots$$

As before difference matching identifies the accumulating term structure. However, rather than try to speculate some lemma for moving this difference out of the way, we generalize the common subterm $rev(x)$. This has the effect of changing the wavefront from, $\boxed{app(\underline{\cdots}, cons(y, nil))}$ to $\boxed{cons(y, \underline{\cdots})}$. The equation can then be simplified using the definitions of *app* and *len*.

## 10 Conclusions

This paper has described a critic which attempts to identify diverging proof attempts and to propose lemmas and generalizations which overcome the divergence. The critic has proved very successful; it enables the system SPIKE to prove many theorems from the definitions alone. The critic's success can be largely attributed to the power of the rippling heuristic. This heuristic was originally developed for proofs using explicit induction but has since found several other applications. To apply the rippling heuristic to the divergence of proofs using implicit induction required the addition of the difference matching procedure. This identifies accumulating term strucure which is causing the divergence. Lemmas and generalizations are then proposed to move this term structure out of the way.

## Acknowledgments

This research was supported by a Human Captial and Mobility Postdoctoral Fellowship. I wish to thank: Adel Bouhoula and Michael Rusinowitch for their invaluable assistance with SPIKE; Pierre Lescanne for inviting me to visit Nancy; David Basin, Alan Bundy, Miki Hermann, Andrew Ireland, and Michael Rusinowitch for their helpful comments and questions; the members of the Eureca and Protheo groups at INRIA; and the members of the DReaM group at Edinburgh.

## References

1. R. Aubin. *Mechanizing Structural Induction*. PhD thesis, University of Edinburgh, 1976.
2. D. Basin and T. Walsh. Difference matching. In D. Kapur, editor, *11th Conference on Automated Deduction*, pages 295–309. Springer Verlag, 1992. Lecture Notes in Computer Science No. 607.
3. D. Basin and T. Walsh. Difference unification. In *Proceedings of the 13th IJCAI*. International Joint Conference on Artificial Intelligence, Chambery, France, 1993.
4. A. Bouhoula, and M. Rusinowitch. Automatic Case Analysis in Proof by Induction. In *Proceedings of the 13th IJCAI*. International Joint Conference on Artificial Intelligence, Chambery, France, 1993.
5. R.S. Boyer and J.S. Moore. *A Computational Logic*. Academic Press, 1979. ACM monograph series.
6. A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993.
7. N. Dershowitz and E. Pinchover. Inductive Synthesis of Equational Programs. In *Proceedings of the 8th National Conference on AI*, pages 234–239. American Association for Artificial Intelligence, 1990.
8. M. Hermann. Crossed term rewriting systems. CRIN Report 89-R-003, Centre de Recherche en Informatique de Nancy, 1989.
9. A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proof. In *Proceedings of LPAR'92*. Springer-Verlag, 1992. Lecture Notes in Artificial Intelligence 624.
10. A. Ireland and A. Bundy. Using failure to guide inductive proof. Technical report 613, Dept. of Artificial Intelligence, University of Edinburgh, 1992.
11. H. Kirchner. Schematization of infinite sets of rewrite rules. Application to the divergence of completion processes. In *Proceedings of RTA'87*, pages 180–191, 1987.
12. M. Protzen. Disproving conjectures. In D. Kapur, editor, *11th Conference on Automated Deduction*, pages 340–354. Springer Verlag, 1992. Lecture Notes in Computer Science No. 607.
13. M. Thomas and K.P. Jantke. Inductive Inference for Solving Divergence in Knuth-Bendix Completion. In *Proceedings of International Workshop AII'89*, pages 288–303, 1989.
14. M. Thomas and P. Watson. Solving divergence in Knuth-Bendix completion by enriching signatures. *Theoretical Computer Science*, 112:145–185, 1993.
15. T. Walsh, A. Nunes, and A. Bundy. The use of proof plans to sum series. In D. Kapur, editor, *11th Conference on Automated Deduction*, pages 325–339. Springer Verlag, 1992. Lecture Notes in Computer Science No. 607.

16. T. Yoshida, A. Bundy, I. Green, T. Walsh, and D. Basin. Coloured rippling: the extension of a theorem proving heuristic. Technical Report, Dept. of Artificial Intelligence, University of Edinburgh, 1993.