Extensions to the Estimation Calculus

Jeremy Gow, Alan Bundy and Ian Green

Institute for Representation and Reasoning,
Division of Informatics, University of Edinburgh,
80 South Bridge, Edinburgh EH1 1HN, Scotland
jeremygo@dai.ed.ac.uk {A.Bundy,I.Green}@ed.ac.uk

Abstract. Walther's estimation calculus was designed to prove the termination of functional programs, and can also be used to solve the similar problem of proving the well-foundedness of induction rules. However, there are certain features of the goal formulae which are more common to the problem of induction rule well-foundedness than the problem of termination, and which the calculus cannot handle. We present a sound extension of the calculus that is capable of dealing with these features. The extension develops Walther's concept of an argument bounded function in two ways: firstly, so that the function may be bounded *below* by its argument, and secondly, so that a bound may exist between two arguments of a predicate. Our calculus enables automatic proofs of the well-foundedness of a large class of induction rules not captured by the original calculus.

1 Introduction

An induction rule is *well-founded* iff there is a well-founded order such that each *step case* of the rule the inductive hypotheses are less in that order than the inductive conclusion. A standard technique for showing validity of an induction rule involves showing the rule to be well-founded, and so automatic techniques for establishing well-foundedness are of interest to the inductive theorem proving community.

The problem of proving an induction rule well-founded is similar to that of proving the termination of a recursive functional program. The current state of the art techniques in automated termination analysis of functional programs are based upon Walther's estimation calculus [10]. Likewise, these techniques currently represent the most powerful approach to automatically proving the well-foundedness of induction rules.

Both termination and well-foundedness proofs involve finding a well-founded relation $\prec$ that satisfies formulae of the form

$$\varphi \to s \prec t \tag{1}$$

In a termination proof of a function[1] f, there is a goal (1), known as a *termination formula*, for each recursive call in a defining equation of f

[1] We do not consider functions defined by mutual or nested recursion

$$\varphi \to f(t) = \cdots f(s) \cdots \tag{2}$$

In a well-foundedness proof, there is goal (1), known as a *well-foundedness formulae*, for each induction hypothesis in a step case of the induction rule

$$\varphi, \ldots, \psi(s), \ldots \vdash \psi(t) \tag{3}$$

However, there are two common features of the induction step case (3) which are appear less often in (2). Firstly, the term t in (3) can contain defined function symbols, whereas the t in (2) is often a pattern (i.e., a linear constructor term) – some languages (e.g. ML) demand this is the case. Secondly, the terms s and t in (3) may be related by a predicate in the step case conditions φ. Although this can occur in (2), it is not a common style of programming. Hence well-foundedness formulae have features whose analogues appear less frequently in termination formulae:

(i) the appearance of non-constructor functions on the right of the inequality, and,
(ii) the two sides of the inequality are related by a predicate that appears in the preconditions.

As the original estimation calculus was designed to prove termination formulae, it does not take account of either of these features, and so fails on well-foundedness formulae when these features are relevant to the solution (several examples are given below).

In this paper we present a sound extension of the estimation calculus which can handle both of these features of well-foundedness formulae. Furthermore, this extended calculus is readily automated in just the same way as Walther's original calculus. Thus the extended calculus enables automatic proofs of the well-foundedness of a strictly larger class of induction rules not captured by Walther's approach. (We discuss below other extensions of the original calculus.) Likewise, it can prove the termination of a larger class of functions, given some formalisms may allow functions with features analogous to (i) and (ii).

The extension is achieved by developing the concept of *argument bounds*. In the original calculus, an argument bounded function is one whose result is bounded above by one of its arguments under the size order. The size order $<_\#$ orders free data types by their value under the size measure $\#$, e.g., natural numbers are ordered by magnitude, and lists by length.

We extend the concept of argument bounds to functions which are bounded below by their arguments, and to predicates in which one argument bounds another. Using these concepts, the calculus is extended in order to deal with features (i) and (ii) described above. For simplicity in this paper, we concentrate on extending Walther's original calculus [10], although our techniques could be combined with some of the other extensions described in §2.3. We also ignore nested or mutually recursive functions, although the extended calculus could be applied to such functions [4].

The features particular to well-foundedness formulae and our extensions to estimation calculus are illustrated by the following two examples. Firstly, consider (4) below as an example of an induction rule whose well-foundedness formulae have feature (i):

$$\frac{\begin{array}{r}\vdash \psi(0)\\ \vdash \psi(s(0))\\ x \neq 0 \wedge y \neq 0,\, \psi(x),\, \psi(y) \vdash \psi(plus(x,y))\end{array}}{\vdash \forall x{:}nat.\, \psi(x)} \tag{4}$$

where *plus* sums two natural numbers. If we attempt to use the size order # to prove this well-founded, we must show that

$$x \neq 0 \wedge y \neq 0 \rightarrow \#(x) < \#(plus(x,y)) \tag{5}$$

$$x \neq 0 \wedge y \neq 0 \rightarrow \#(y) < \#(plus(x,y)) \tag{6}$$

These well-foundedness formulae both display feature (i): non-constructor functions appear on the right of the inequality. If we know that *plus* is bounded below by its first argument, relative to #, and that this bound is strict when the second argument is non-zero, i.e.,

$$v \neq 0 \rightarrow \#(u) < \#(plus(u,v)) \tag{7}$$

then we can easily discharge (5). This is the basic approach taken by the estimation calculus: find an argument bound, synthesise lemmas giving conditions on the strictness of this bound (like (7)) and then show that these conditions hold. Formula (6) can be discharged with a similar insight about the second argument of *plus*.

However, this example cannot be solved by the estimation calculus. Because the termination formulae it was designed to solve rarely display feature (i), it only reasons with functions which are bounded *above* by one of their arguments. The crucial part of this proof is to recognise the *lower* argument bound on *plus*. Our extended calculus can solve such well-foundedness conditions by reasoning about lower argument bounds.

Our second example (8) has well-foundedness formulae which illustrate feature (ii) described above. Here *shorter* is a predicate that holds only when its first argument is a shorter list than its second argument.

$$\frac{\vdash \psi(nil) \qquad shorter(x,y),\, \psi(x) \vdash \psi(y)}{\vdash \forall x{:}list(\tau).\, \psi(x)} \tag{8}$$

To establish well-foundedness using the size order, we need to discharge

$$shorter(x,y) \rightarrow \#(x) < \#(y) \tag{9}$$

This well-foundedness formula displays feature (ii): the two sides of the inequality are related by a predicate that appears in the preconditions. If we know that when *shorter* holds, its first argument is bounded above by the second argument, relative to $\#$, and that this bound is *always* strict, then we can discharge (9). Notice we have taken the estimation calculus approach again: find an argument bound, synthesise a lemma giving conditions on the strictness of this bound and show these conditions hold – in this example the conditions are trivially true.

The original estimation calculus cannot solve this example, as the crucial part of the proof is recognising the relevant argument bound holds between the first and second arguments of *shorter*. The calculus can only reason about argument bounded functions, and not argument bounded *predicates* that appear in the conditions on the inequality. This is because these rarely appear in the termination formulae the calculus was designed to prove. Our extended calculus can solve such well-foundedness conditions by reasoning about bounds between arguments of predicates.

Although there exist more powerful techniques which can reason about features (i) and (ii), i.e. [3] and [1], our calculus has advantages over these. The main contribution of this paper is that such reasoning can be 'built in' to Walther's calculus in a way analogous to the original, and which retains its simplicity. The method is simpler and easier to implement than comparable techniques, and although less powerful, is capable of coping with many common examples.

The remainder of this paper is organised as follows: we provide some background on the estimation calculus in §2. The extension for handling the occurrence of non-constructor functions in the conclusion of a step case is presented in §3, and the extension for formulae where the two sides of the inequality are related by a predicate that appears in the conditions is described in §4. Refinements and possible developments of our approach are discussed in §5, and in §6 we draw our conclusions.

Conventions We use $i \in [n]$ to denote $1 \leq i \leq n$, and $\overrightarrow{s_n}$ to denote $s_1, \ldots, s_n$. Each n-ary constructor c has n associated destructor functions $d_c^1, \ldots, d_c^n$ which return the arguments of c, defined as $d_c^i(c(\overrightarrow{t_n})) = t_i$, a everywhere else, where a is an arbitrary nullary constructor of the appropriate type. It is assumed that such a constructor exists for each type.

2 Background

Proving induction rules well-founded, and functional programs terminating (excluding nested and mutually recursive programs), requires us to find a well-founded relation[2] $\prec$ which satisfies a set of formulae of the form

$$\varphi \to (\overrightarrow{s_n}) \prec (\overrightarrow{t_n}) \tag{10}$$

There is a well-foundedness formula of this form for each inductive hypothesis, where the s_i are values of the induction variables in the hypothesis, the t_i are the values in the conclusion of this step case and φ are the conditions on this case. In the case of termination proofs, there is a termination formula (10) for each

[2] A relation is well-founded if it does not contain any infinite descending chains.

recursive call – the s_i are the arguments of this call, the t_i are the arguments of the head of this defining case and φ are the case conditions.

If a relation $\prec$ is well-founded on β, a measure functions $m : \alpha \to \beta$ can be used to induce a well-founded relation $\prec_m$, defined by

$$\forall x, y{:}\alpha.\ \big(x \prec_m y \leftrightarrow m(x) \prec m(y)\big)$$

The estimation calculus [10] attempts to prove sets of well-foundedness formulae using the well-founded size order $<_\#$. The size measure $\# : \tau \to nat$ counts the number of reflexive[3] type τ constructors in a type τ data-structure, where substructures of other types are ignored. The rest of this section gives a brief summary of the estimation calculus – for more details see [10].

2.1 Argument Bounds and Difference Predicates

Walther defines an argument bounded function as one whose result is smaller under $\leq_\#$ than one of its arguments. In order to avoid confusion later, we refer to these as *upper* argument bounded functions, because the argument is an upper bound on the function. Formally:

Definition 1 (Upper Argument Bounded Function). *A function* $f : \tau_1 \times \cdots \times \tau_n \to \tau$ *is* upper p-bounded *iff* $p \in [n]$ *and*

$$\forall t_1{:}\tau_1 \ldots t_n{:}\tau_n.\ f(\overrightarrow{t_n}) \leq_\# t_p$$

A function is upper argument bounded *iff it is upper p-bounded for some p.*

For each upper argument bounded position p of a function f, there is a difference predicate which is true only when the upper bound is strict. Formally:

Definition 2 (Difference Predicate). *If f is upper p-bounded, the* difference predicate Δ_f^p *is defined by*

$$\Delta_f^p(\overrightarrow{t_n}) = \big(f(\overrightarrow{t_n}) <_\# t_p\big)$$

Note that predicates are treated as functions with the range $\{\text{TRUE}, \text{FALSE}\}$. For an n-ary predicate P we write $P(\overrightarrow{x_n}) = \text{TRUE}$ as $P(\overrightarrow{x_n})$ (see [10] for further details).

2.2 The Estimation Calculus

Walther's calculus is given in Fig. 1, which we have recast as a sequent-style system. The measured data type has k reflexive constructors $\overrightarrow{r_k}$, and l irreflexive constructors $\overrightarrow{ir_l}$. Each r_i is reflexive on the set of argument positions R_i.

The calculus is used to derive sequents of the form $\langle s \leq_\# t, \Delta\rangle$, and is sound in that $\vdash_E \langle s \leq_\# t, \Delta\rangle$ implies both $s \leq_\# t$ and $\Delta \leftrightarrow s <_\# t$. Well-foundedness

[3] A function is reflexive if its range type is one of its domain types.

Assumption

$$\frac{}{\Gamma \vdash_E A} \text{ if } A \in \Gamma$$

Identity

$$\frac{}{\Gamma \vdash_E \langle t \leq_{\#} t,\ \text{FALSE}\rangle}$$

Equivalence

$$\frac{}{\Gamma \vdash_E \langle ir_i(\overrightarrow{s_n}) \leq_{\#} ir_j(\overrightarrow{t_m}),\ \text{FALSE}\rangle} \text{ if } i, j \in [l]$$

Strong Estimation

$$\frac{}{\Gamma \vdash_E \langle ir_i(\overrightarrow{s_n}) \leq_{\#} r_j(\overrightarrow{t_m}),\ \text{TRUE}\rangle} \text{ if } i \in [l], j \in [k]$$

Minimum

$$\frac{}{\begin{array}{r}\Gamma \vdash_E \langle ir_i(\overrightarrow{s_n}) \leq_{\#} t\ ,\ t = r_1(d^1_{r_1}(t), \ldots, d^{n_1}_{r_1}(t)) \vee \cdots \\ \cdots \vee t = r_k(d^1_{r_k}(t), \ldots, d^{n_k}_{r_k}(t))\rangle\end{array}} \text{ if } i \in [l]$$

Upper Bound Estimation

$$\frac{\Gamma \vdash_E \langle s_p \leq_{\#} t,\ \Delta\rangle}{\Gamma \vdash_E \langle f(\overrightarrow{s_n}) \leq_{\#} t,\ \Delta \vee \Delta^p_f(\overrightarrow{s_n})\rangle} \text{ if } f \text{ is upper } p\text{-bounded}$$

Strong Embedding

$$\frac{\Gamma \vdash_E \langle s \leq_{\#} t_j,\ \Delta\rangle}{\Gamma \vdash_E \langle s \leq_{\#} r_i(\overrightarrow{t_m}),\ \text{TRUE}\rangle} \text{ if } i \in [k], j \in R_i$$

Weak Embedding

$$\frac{\Gamma \vdash_E \langle s_{j_1} \leq_{\#} t_{j_1}, \Delta_1\rangle, \ldots, \Gamma \vdash_E \langle s_{j_m} \leq_{\#} t_{j_m}, \Delta_n\rangle}{\Gamma \vdash_E \langle r_i(\overrightarrow{s_n}) \leq_{\#} r_i(\overrightarrow{t_n}),\ \Delta_1 \vee \cdots \vee \Delta_n\rangle} \quad \begin{array}{l}\text{if } i \in [k], \\ R_i = \{\overrightarrow{j_m}\}\end{array}$$

Fig. 1. The estimation calculus

conditions of the form (10) are proved by showing $\vdash_E \langle s_i \leq_{\#} t_i, \Delta\rangle$ for some $i \in [n]$ and then using a theorem prover to establish $\varphi \to \Delta$.

The calculus rules can be used in reverse to decompose the goal formula $\langle s \leq_{\#} t, \Delta\rangle$, where the identity of Δ is initially unknown. If we represent this unknown as a meta-variable which can be instantiated by rule applications, then the difference formula Δ can be constructed during the analysis[4].

[4] Walther's original approach to using the calculus was to recast it as a production rule system whose rules constructed Δ as they decomposed the inequality. The approaches are trivially equivalent.

Recognising argument bounded functions and synthesising difference predicates is done automatically using the estimation calculus. An upper p-bounded function f is recognised by performing a meta-induction proof that demonstrates that each defining case of f returns a value no larger under $<_\#$ than the pth argument (see [10] for details). If it exists, the corresponding difference predicate is synthesised as a by-product of this analysis.

2.3 Related Techniques

Based on the estimation calculus, Giesl developed a similar calculus that works with arbitrary measure functions based on polynomial norms [2]. As it is not restricted to using the size measure, it is a much more powerful approach. The method still had the drawback that the user must supply the appropriate measure function. To overcome this Giesl adapted the approach to automatically synthesise these measure functions, using techniques from termination analysis of term rewriting systems [3]. This latter technique is quite different from the estimation calculus, and does not use argument bounded functions. A good overview of this research can be found in [5].

The estimation calculus has also been extended to work with certain non-free data types [8], and has been used as the basis for Walther recursive programs [6], a class of functional programs for which termination is decidable.

3 Lower Argument Bounded Functions

In this section we describe our extension for feature (i): the occurrence of non-constructor functions on the right of the inequality. If a well-foundedness formula has this feature, then proving it requires us to show $\vdash_E \langle s \leq_\# t, \Delta\rangle$, where t contains non-constructor functions. The calculus fails in these situations because it has no rules which can derive theorems of this form.

We can extend the estimation calculus to allow non-constructor functions f to be added to t, providing that they do not decrease the value of this term under the size measure. In other words, the value of $f(\ldots,t,\ldots)$ is bounded below by the value of t. We call these functions *lower* argument bounded functions, and define them as follows:

Definition 3 (Lower Argument Bounded Function). *A function $f : \tau_1 \times \cdots \times \tau_n \to \tau$ is* lower p-bounded *iff $p \in [n]$ and*

$$\forall t_1{:}\tau_1 \cdots t_n{:}\tau_n .\, t_p \leq_\# f(\overrightarrow{t_n})$$

A function is lower argument bounded *iff it is lower p-bounded for some p.*

Before we can extend the calculus to use lower argument bounded functions, we need to be able to synthesise a difference predicate that is true iff the lower argument bound is strict. The process is exactly analogous to the upper bound case – the difference predicate Δ_f^p is synthesised while verifying that f is lower

p-bounded – and is described in §3.1. We can now extend the estimation calculus by adding the following inference rule (11) to handle lower argument bounded functions.

Lower Bound Estimation

$$\frac{\Gamma \vdash_E \langle s \leq_\# t_p,\, \Delta\rangle}{\Gamma \vdash_E \langle s \leq_\# f(\overrightarrow{t_n}), \Delta \vee \Delta^p_f(\overrightarrow{t_n})\rangle} \text{ if } f \text{ is lower } p\text{-bounded} \tag{11}$$

Because all constructor functions are argument bounded on their reflexive argument positions, the strong embedding rule (see Fig. 1) is now redundant, being subsumed rule (11). Below we use $\vdash_E$ to denote the estimation calculus extended with our new rule (11).

Theorem 1. *Rule (11) is sound.*

Proof. Assume f is lower p-bounded and $\langle s \leq_\# t_p, \Delta\rangle$. By definition $s \leq_\# t_p$ and $\Delta \leftrightarrow s <_\# t_p$, and $t_p \leq_\# f(\overrightarrow{t_n})$ and $\Delta^p_f(\overrightarrow{t_n}) \leftrightarrow t_p <_\# f(\overrightarrow{t_n})$. Now:

(a) $s \leq_\# f(\overrightarrow{t_n})$, by $s \leq_\# t_p$ and $t_p \leq_\# f(\overrightarrow{t_n})$.
(b) $\Delta \vee \Delta^p_f(\overrightarrow{t_n}) \rightarrow s <_\# f(\overrightarrow{t_n})$, as $\Delta \rightarrow s <_\# f(\overrightarrow{t_n})$ and $\Delta^p_f(\overrightarrow{t_n}) \rightarrow s <_\# f(\overrightarrow{t_n})$ by (a).
(c) $s <_\# f(\overrightarrow{t_n}) \rightarrow \Delta \vee \Delta^p_f(\overrightarrow{t_n})$, because $s \leq_\# t_p \leq_\# f(\overrightarrow{t_n})$, so $s <_\# f(\overrightarrow{t_n}) \rightarrow t_p \neq s \vee t_p \neq f(\overrightarrow{t_n})$. Hence $s <_\# f(\overrightarrow{t_n}) \rightarrow s <_\# t_p \vee t_p <_\# f(\overrightarrow{t_n})$.

Therefore $\langle s \leq_\# f(\overrightarrow{t_n}), \Delta \vee \Delta^p_f(\overrightarrow{t_n})\rangle$ as required. □

Given the original estimation calculus and the new rule (11) are both sound, our extended calculus $\vdash_E$ is also sound.

As an example of rule (11) in operation, consider the following induction rule, taken from [7]:

$$\frac{\vdash \psi(nil) \qquad\qquad \psi(l) \vdash \psi(app(l, cons(x, nil)))}{\vdash \forall l{:}list(\tau).\, \psi(l)} \tag{12}$$

Here *nil* and *cons* are the list constructors and *app* appends two lists, defined as

$$app(nil, l) = l \tag{13}$$
$$app(cons(h,t), l) = cons(h, app(t,l)) \tag{14}$$

We can verify that *app* is lower 1-bounded, with the associated difference predicate Δ^1_{app} (see §3.1 for details), defined as

$$\Delta^1_{app}(nil, l) = (l = cons(hd(l), tl(l))) \tag{15}$$
$$\Delta^1_{app}(cons(h,t), l) = \Delta^1_{app}(t, l) \tag{16}$$

We can use the size measure to prove (12) well-founded: $\vdash_E \langle l \leq_\# l,\ \text{FALSE}\rangle$ by the identity rule, and then by lower bound estimation

$$\vdash_E \langle l \leq_\# app(l, cons(x, nil)),\ \text{FALSE} \vee \Delta^1_{app}(l, cons(x, nil))\rangle$$

It is within the power of current automatic inductive theorem provers (e.g., Clam [9]) to show that the difference formulae $\text{FALSE} \vee \Delta^1_{app}(l, cons(x, nil))$ is true, and so the inequality is strict. Hence the induction rule (12) is well-founded. Note this cannot be established using the original calculus, because of the non-constructor function *app* appearing on the right hand side of the inequality.

In [3] termination/well-foundedness formulae are converted into a set of constraints on a polynomial measure, and a suitable measure is generated. This relieves the user of having to provide suitable measures for the proof. It is also general enough to handle goal formulae with feature (i), and so could be used as an alternative to the estimation calculus extended with our rule (11). However, our approach is considerably simpler and easier to implement. Of course, it can only be used in situations where the size measure is sufficient, but this includes many common induction rules/functions.

3.1 Recognising Lower Argument Bounded Functions

When an n-ary function is defined, we attempt to prove it is lower p-bounded for each $p \in n$. We assume it has been shown terminating, and has a set of mutually exclusive and exhaustive defining equations. Recall that we do not consider nested or mutually recursive functions here. To verify that f is lower p-bounded for some p we must show

$$\vdash_E \langle t_p \leq_\# f(\overrightarrow{t_n}),\ \Delta^p_f(\overrightarrow{t_n})\rangle \tag{17}$$

for some difference predicate Δ^p_f. As in the upper argument bounded case (for details see [10]), we prove this property by a meta-induction over the estimation calculus which corresponds to the recursive structure of f. The difference predicate Δ^p_f is synthesised during this process – each case of the meta-induction adds an equation to its definition.

So for each defining equation of f

$$\varphi \rightarrow f(\overrightarrow{t_n}) = b \tag{18}$$

where b contains k recursive calls $f(s_{1,1}, \ldots, s_{1,n}), \ldots, f(s_{k,1}, \ldots, s_{k,n})$, we must verify a case of our meta-induction corresponding to (18)

$$\begin{array}{l} \langle s_{1,p} \leq_\# f(s_{1,1}, \ldots, s_{1,n}),\ \Delta^p_f(s_{1,1}, \ldots, s_{1,n})\rangle, \\ \qquad\qquad\vdots \\ \langle s_{k,p} \leq_\# f(s_{k,1}, \ldots, s_{k,n}),\ \Delta^p_f(s_{k,1}, \ldots, s_{k,n})\rangle \vdash_E \langle t_p \leq_\# b,\ \Delta\rangle \end{array} \tag{19}$$

for some Δ. Note there may be no recursive calls in b, and so they will be no inductive hypotheses.

The corresponding difference predicate is synthesised as a by-product: for each case of our meta-induction (19), we add the following equation to the definition of Δ_f^p

$$\varphi \rightarrow \Delta_f^p(\overrightarrow{t_n}) = \Delta \tag{20}$$

The above meta-induction is guaranteed valid, because we demand f is terminating and has a set of mutually exclusive and exhaustive defining equations. If we use this scheme to prove (17), then for each case (18) there is a meta-induction case

$$\varphi,\, h_1,\, \ldots,\, h_k \vdash \langle t_p \leq_{\#} f(\overrightarrow{t_n}),\, \Delta_f^p(\overrightarrow{t_n})\rangle$$

where $h_1, \ldots, h_k$ are the inductive hypotheses of (19). By the definitions of f (18) and Δ_f^p (20) it is sufficient to prove (19). Hence the meta-induction proves (17).

Furthermore, as $\vdash_E$ is sound, (17) implies $t_p \leq_{\#} f(\overrightarrow{t_n})$ and $\Delta_f^p(\overrightarrow{t_n}) \leftrightarrow t_p \leq_{\#} f(\overrightarrow{t_n})$. So by definition 3, the meta-induction verifies that f is lower p-bounded and has difference predicate Δ_f^p.

The process of recognising lower argument bounded functions is illustrated by the verification *app* (see §3) is a lower 1-bounded function. For defining equation (13) we use the minimum rule to show

$$\vdash_E \langle nil \leq_{\#} l, (l = cons(hd(l), tl(l)))\rangle$$

(15) is extracted from this. For the recursive equation (14) we can use the weak embedding rule to show

$$\langle t \leq_{\#} app(t,l), \Delta_{app}^1(t,l)\rangle \vdash_E \langle cons(h,t) \leq_{\#} cons(h, app(t,l)), \Delta_{app}^1(t,l)\rangle$$

from which (15) is extracted. Hence *app* is lower 1-bounded, with the difference predicate defined by (15) and (16).

4 Argument Bounded Predicates

We now describe our extension for feature (ii): the two sides of the inequality are related by a predicate that appears in the preconditions. A well-foundedness formula with this feature requires us to show $\vdash_E \langle s \leq_{\#} t,\, \Delta\rangle$, where s is less than t because of the preconditions. This is not possible in the original calculus, which ignores these conditions.

Although the conditions φ may entail $s \leq_{\#} t$, it may require arbitrarily hard theorem proving to establish this – and we would still be left with the problem of synthesising the appropriate difference predicate. We adopt a restricted but more practical approach in which $\varphi \rightarrow w(\overrightarrow{t_n})$ is tested using a decision procedure[5],

[5] For example, that the formula is a tautology.

such that $s = t_p$ and $t = t_q$, where w is a predicate that is mentioned in w and whose pth argument is never greater under the size measure than its qth argument. In other words, w ensures t is bounded below by s. We call w an *argument bounded predicate*, defined as follows:

Definition 4 (Argument Bounded Predicate). *A predicate* $w : \tau_1 \times \cdots \times \tau_n \to bool$ *is* (p, q)-bounded *iff* $1 \leq p, q \leq n$, $p \neq q$ *and*

$$\forall t_1{:}\tau_1 \cdots t_n{:}\tau_n .\, w(\overrightarrow{t_n}) \to t_p \leq_{\#} t_q$$

A predicate is argument bounded *iff it is* (p, q)*-bounded for some* p, q.

As with argument bounded functions, there is a difference predicate $\Delta_w^{(p,q)}$ that is equivalent to this bound being strict, i.e. $w(\overrightarrow{t_n}) \to (\Delta_w^{(p,q)}(\overrightarrow{t_n}) \leftrightarrow t_p \leq_{\#} t_q)$, and which is synthesised while verifying w is (p, q)-bounded. This is described in §4.1. We can now extend the estimation calculus by adding an inference rule (21) to handle argument bounded predicates in the conditions.

Condition Bound

$$\frac{}{\Gamma \vdash_E \langle t_p \leq_{\#} t_q,\, \Delta_w^{(p,q)}(\overrightarrow{t_n})\rangle} \tag{21}$$

Providing (p, q)-bounded w in φ and $\varphi \to w(\overrightarrow{t_n})$ is a tautology.

Theorem 2. *Rule (21) is sound.*

Proof. Assume w is (p, q)-bounded and $\varphi \to w(\overrightarrow{t_n})$ is a tautology. As φ is the current condition, $w(\overrightarrow{t_n})$ holds. By definition 4, $w(\overrightarrow{t_n}) \to t_p \leq_{\#} t_q$, so $t_p \leq_{\#} t_q$. Also, $w(\overrightarrow{t_n}) \to (\Delta_w^{(p,q)}(\overrightarrow{t_n}) \leftrightarrow t_p <_{\#} t_q)$, so $\Delta_w^{(p,q)}(\overrightarrow{t_n}) \leftrightarrow t_p <_{\#} t_q$. Hence $\langle t_p \leq_{\#} t_q,\, \Delta_w^{(p,q)}(\overrightarrow{t_n})\rangle$ as required. □

Extending $\vdash_E$ with (21) preserves soundness; henceforth we shall refer to this system (i.e., $\vdash_E$ with the addition of rule (21)) as $\vdash_E$.

As an example of the use of rule (21), consider the follow induction rule:

$$\frac{\vdash \psi(nil) \qquad leqlen(l, m),\, \psi(l) \vdash \psi(cons(x, m))}{\vdash \forall l{:}list(\tau).\, \psi(l)} \tag{22}$$

Here *leqlen* is a predicate that holds when its first argument is a list not longer than its second argument, and is defined as

$$leqlen(nil, m) = \text{TRUE} \tag{23}$$
$$leqlen(cons(g, s), nil) = \text{FALSE} \tag{24}$$
$$leqlen(cons(g, s), cons(h, t)) = leqlen(s, t) \tag{25}$$

We can show that *leqlen* is $(1, 2)$-bounded, and has the difference predicate $\Delta_{leqlen}^{(1,2)}$ (see §4.1 for details), defined as

$$\Delta^{(1,2)}_{leqlen}(nil, m) = (m = cons(hd(m), tl(m))) \tag{26}$$

$$\Delta^{(1,2)}_{leqlen}(cons(g, s), nil) = \text{FALSE} \tag{27}$$

$$\Delta^{(1,2)}_{leqlen}(cons(g, s), cons(h, t)) = \Delta^{(1,2)}_{leqlen}(s, t) \tag{28}$$

To establish the well-foundedness of (22) using the size order, we can use the condition bound rule (21) to derive $\vdash_E \langle l \leq_\# m,\ \Delta^{(1,2)}_{leqlen}(l, m)\rangle$, followed by lower bound estimation, given that $cons$ is lower 2-bounded.

$$\vdash_E \langle l \leq_\# cons(x, m),\ \Delta^{(1,2)}_{leqlen}(l, m) \vee \Delta^2_{cons}(x, m)\rangle$$

The difference formula is true, as $\Delta^2_{cons}(x, m)$ is defined as TRUE. Hence induction rule (22) is well-founded. Note that this example cannot be solved using the original estimation calculus, as it does not consider the conditions on the well-foundedness formulae.

Brauburger and Giesl use inductive evaluation to exploit the conditions on the inequality in termination formulae [1], and so their method could also be used as an alternative to the condition bound rule (21). However, this requires an inductive theorem prover to solve subgoals that correspond to proving the predicate is strictly argument bounded. Our approach performs this analysis when the predicate is first defined, and so requires less theorem proving support during execution. It is simpler to identify argument bounded predicates when they are defined, and to use the condition bound rule when possible. Of course, there are many situations where rule (21) is not relevant and inductive evaluation is required.

4.1 Recognising Argument Bounded Predicates

When an n-ary predicate is defined, we attempt to prove it is (p, q)-bounded for each $p \neq q$, $1 \leq p, q \leq n$. We assume it has been shown terminating (recall our predicates are functions onto $\{\text{TRUE}, \text{FALSE}\}$) and has a set of mutually exclusive and exhaustive defining equations. To verify that w is (p, q)-bounded for some p and q we must show that

$$\vdash_E \langle t_p \leq_\# t_q,\ \Delta^{(p,q)}_w(\vec{t_n})\rangle \tag{29}$$

when $w(\vec{t_n})$ holds, for some difference predicate $\Delta^{(p,q)}_w$. We proceed as in the argument bounded function case (see §3.1 and [10]), by a meta-induction over the estimation calculus according to the recursive structure of w. Again each case of the meta-induction adds an equation to the definition of the difference predicate $\Delta^{(p,q)}_f$.

However, because we have the extra assumption $w(\vec{t_n})$, the details of the meta-induction are somewhat different from the functional case. For each defining equation of w

$$\varphi \rightarrow \quad w(\overrightarrow{t_n}) = b \tag{30}$$

we require that b is a quantifier-free formula over the free variables of $w(\overrightarrow{t_n})$. This formula is converted into disjunctive normal form $b' = d_1 \vee \cdots \vee d_m$. Recall that we only want to establish (29) when $w(\overrightarrow{t_n})$ holds, so if $b =$ FALSE we can ignore the case (30) and do not care what value $\Delta_w^{(p,q)}(\overrightarrow{t_n})$ takes – a case assigning it FALSE under the condition φ is added.

Otherwise, we must prove a case of the meta-induction corresponding to (30) when $w(\overrightarrow{t_n}) =$ TRUE. The latter implies at least one of the disjuncts d_i must hold. If d_i holds and contains the set of positive literals p_i, we can make the following assumptions

1. For each $w(\overrightarrow{s_n})$ in p_i we can assume $\langle s_p \leq_{\#} s_q,\ \Delta_w^{(p,q)}(\overrightarrow{s_n})\rangle$.
2. For each $z(\overrightarrow{s_m})$ is in p_i, such that z is a (u, v)-bounded predicate, we can assume $\langle s_p \leq_{\#} s_q,\ \Delta_z^{(u,v)}(\overrightarrow{s_n})\rangle$.

For each d_i we collect such a set of assumptions $h_1, \ldots, h_a$ and verify the following meta-induction case

$$h_1, \ldots, h_a \ \vdash_E \ \langle t_p \leq_{\#} t_q,\ \Delta\rangle \tag{31}$$

If this proof is successful we create the following defining equation for $\Delta_w^{(p,q)}$:

$$\varphi \rightarrow \quad \Delta_w^{(p,q)}(\overrightarrow{t_n}) = \Delta$$

Compare our meta-induction with the induction based upon the recursive structure of w. Ours has the same case structure, with extra cases splits on the disjuncts $d_1 \vee \cdots \vee d_m$, and only uses inductive hypotheses which would also appear in the latter induction. The meta-induction is valid since w is terminating and has a set of mutually exclusive and exhaustive defining equations. So if the meta-induction succeeds, then (29) is established under the assumption $w(\overrightarrow{t_n})$.

Given $w(\overrightarrow{t_n})$ implies (29), the soundness of $\vdash_E$ yields $w(\overrightarrow{t_n}) \rightarrow t_p \leq_{\#} t_q$ and $w(\overrightarrow{t_n}) \rightarrow (\Delta_w^{(p,q)}(\overrightarrow{t_n}) \leftrightarrow t_p \leq_{\#} t_q)$. So by definition 4, the meta-induction correctly verifies that w is (p, q)-bounded and has difference predicate $\Delta_w^{(p,q)}$.

Our approach to recognising argument bounded predicates is illustrated by the verification of $leqlen$ (see §4) as a $(1, 2)$-bounded predicate. Consider defining equation (23) of $leqlen$: we use the minimum rule to show

$$\vdash_E \langle nil \leq_{\#} m, (m = cons(hd(m), tl(m)))\rangle$$

which gives us (26). The defining equation (24) has FALSE on the right, so this case is ignored, and $\Delta_{leqlen}^{(1,2)}(cons(g, s), nil)$ set to FALSE. For the third defining equation (28) there is a single disjunct containing a single positive literal $leqlen(s, t)$. Hence we use the weak embedding rule to show

$$\langle s \leq_{\#} t, \Delta^{(1,2)}_{leqlen}(s,t)\rangle$$
$$\vdash_E \langle cons(g,s) \leq_{\#} cons(h,t), \Delta^{(1,2)}_{leqlen}(cons(g,s), cons(h,t))\rangle$$

from which (28) is extracted. Hence *leqlen* is $(1,2)$-bounded, with the difference predicate defined by (26), (27) and (28).

5 Further Work

Our extended calculus consists of lower bound estimation and condition bound rules added to the original estimation calculus, minus the strong embedding rule – which is subsumed by lower bound estimation. There are a number of refinements that could be made to improve its performance. Many of those suggested by Walther for his original calculus [10] would be similarly applicable to our work, e.g., the optimisation of difference algorithms.

Argument bounded predicates can give us useful information even when their bound arguments are not simply the terms of the inequality we want to derive. For instance, consider the following induction rule:

$$\frac{\begin{array}{r}\vdash \psi(nil)\\ less(len(l), len(m)),\, \psi(l) \vdash \psi(m)\end{array}}{\vdash \forall l{:}list(\tau).\, \psi(l)} \tag{32}$$

Here *less* is less than on natural numbers, and *len* returns the length of a list. *less* is also $(1,2)$-bounded, so we can use the condition bound rule to derive

$$\langle len(l) \leq_{\#} len(m),\, \Delta^{(1,2)}_{less}(len(l), len(m))\rangle$$

This can be used to prove induction rule (32) well-founded, providing we know the following properties of *len*:

$$\forall x, y{:}list(\tau).\, len(x) \leq_{\#} len(y) \rightarrow x \leq_{\#} y \tag{33}$$
$$\forall x, y{:}list(\tau).\, len(x) <_{\#} len(y) \rightarrow x <_{\#} y \tag{34}$$

Such reasoning could be included in the extended calculus, where properties like (33) and (34) are established when the functions are initially defined.

The use of lower argument bounded functions and argument bounded predicates could be incorporated into Giesl's calculus for polynomial norm measure functions [2], given that it works on similar principles to the estimation calculus. This would give our benefits for well-foundedness proofs, without the restriction of using only the size measure.

6 Conclusions

We have presented a fully automatic technique for proving that induction rules are well-founded. It is an sound extension of the estimation calculus designed to handle two common features of well-foundedness formulae for induction rules. These features are i) non-constructor functions on the right of the inequality and ii) a predicate in the preconditions which relates the two sides of the inequality. The original estimation calculus did not take account of either of these features, as they rarely appear in the termination formulae it was designed to solve. Consequently, our calculus is more powerful.

Although both features could be tackled using alternative techniques our approach is simpler and will either require less theorem proving support during execution, or will be easier to implement than comparable methods.

Acknowledgements Thanks to Jürgen Brauburger, Simon Colton, Stephen Cresswell, Ben Curry, Jürgen Giesl, Andrew Ireland and Christoph Walther for many helpful comments. This research was supported by the EPSRC grant GR/J/80702.

References

1. J. Brauburger and J. Giesl. Termination analysis by inductive evaluation. In C. Kirchner and H. Kirchner, editors, *15th International Conference on Automated Deduction*, pages 254–269. LNAI 1421, Springer-Verlag, 1998.
2. J. Giesl. Automated termination proofs with measure functions. In I. Wachsmuth, C. Rollinger, and W. Brauer, editors, *19th Annual German Conference on Artificial Intelligence*, pages 149–160. LNAI 981, Springer-Verlag, 1995.
3. J. Giesl. Termination analysis for functional programs using term orderings. In *2nd International Static Analysis Symposium*. LNCS 983, Springer-Verlag, 1995.
4. J. Giesl. Termination of nested and mutually recursive algorithms. *Journal of Automated Reasoning*, 19:1–29, 1997.
5. J. Giesl, C. Walther, and J. Brauburger. Termination analysis for functional programs. In W. Bibel and P.H. Schmitt, editors, *Automated Deduction – A Basis for Applications, Vol III: Applications*, volume 10 of *Applied Logic Series*, chapter 6, pages 135–164. Kluwer Academic, 1998.
6. David McAllester and Kostas Arkoudas. Walther recursion. In M.A. McRobbie and J.K. Slaney, editors, *13th International Conference on Automated Deduction*, pages 643–657. LNAI 1104, Springer Verlag, July 1996.
7. L.C. Paulson. *ML for the Working Programmer*. Cambridge University Press, 1991.
8. C. Sengler. Termination of algorithms over non-freely generated datatypes. In M.A. McRobbie and J.K. Slaney, editors, *13th International Conference on Automated Deduction*, pages 121–135. LNAI 1104, Springer Verlag, July 1996.
9. Frank van Harmelen, Andrew Ireland, Andrew Stevens, Santiago Negrete, and Alan Smaill. The Clam proof planner, user manual and programmer manual (version 2.2). Technical report, Department of Artificial Intelligence, 1993.
10. C. Walther. On proving termination of algorithms by machine. *Artificial Intelligence*, 71(1):101–157, 1994.