**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

**Research
Report**

RR 97-03

Using Rippling to Prove the Termination of Algorithms

Dieter Hutter

February 1997

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Using Rippling to Prove the Termination of Algorithms

Dieter Hutter

DFKI-RR 97-03

This work has been supported by a grant from The Federal Ministry of Education, Science, Research and Technology (FKZ ITWM-9600).

ISSN 0946-008X

Using Rippling to Prove the Termination of Algorithms

Dieter Hutter

Abstract

When proving theorems by explicit induction the used induction orderings are synthesized from the recursion orderings underlying the definition principles for functions and predicates. In order to guarantee the soundness of a generated induction scheme the well-foundedness of the used recursion orderings has to be proved.

In this paper we present a method to synthesize appropriate measure functions in order to prove the termination of algorithms. We use Walthers' estimation-calculus as a "black-box procedure" in these explicit proofs. Thus, we inherit both, the flexibility of an explicit representation of the termination proof as well as the in-built knowledge concerning the count ordering.

1 Introduction

Proving theorems by induction, the selection of an appropriate induction ordering is a source of infinite branching. In practice (e.g. NQTHM [BM79] or INKA [BHHW86, HS96]) an induction scheme is synthesized from existing orderings which are implicitly given by the user while specifying the behavior of functions and predicates by so-called *algorithmic* definitions. The recursion ordering underlying such an algorithmic specification of a function f is used to formulate an appropriate induction scheme for properties of f. Yet, the proof that the underlying recursion ordering is well-founded is a necessary precondition for the soundness of the generated induction scheme.

While in theory this *halting-problem* is known to be undecidable, a lot of effort has been spent to develop methods which can prove a reasonable subset of algorithms to be terminating. In the field of Knuth-Bendix-based approaches orderings on terms — like KB-orderings, recursive path-orderings — are used to guarantee the termination of rewriting. In the framework of explicit induction we are more interested in orderings $\prec$ on the set $\mathcal{A}(\mathcal{T}(\Sigma))$ of objects denoted by the set of ground terms. Defining $\prec'$ by $s \prec' t$ iff $\mathcal{A}(s) \prec \mathcal{A}(t)$, we obtain a partial ordering on ground terms which we extend to non-ground terms by $s[x] \prec' t[x]$ iff $\sigma(s[x]) \prec' \sigma(t[x])$ for all variable assignments σ of x.[1]

[1] Throughout this paper we will not distinguish between $\prec$ and its implied ordering $\prec'$, instead we will use $\prec$ to denote both orderings.

Walther presented an approach [Wal94] to prove the termination of algorithms which are specified on freely generated datatypes. These datatypes possess the *unique-factorization property*, i.e. – roughly speaking — the term algebra on the constructor functions is the intended model of the axiomatization. Thus, each object has a unique constructor representation and the *size* of an object is uniquely determined by the number of (reflexive) constructors used to denote the object (see [GTW78, Wal94] for details). Walthers' approach is based on the so-called *count ordering* $\prec_{\#}$ which compares this size by $<_N$. Based on a so-called *estimation*-calculus Γ he introduces a rewrite system[2] which can efficiently deal with proof obligations of the form $\forall x^*\ \Phi(x^*) \rightarrow s(x^*) \prec_{\#} t(x^*)$. Roughly speaking, given two terms $s(x^*)$ and $t(x^*)$ it derives - if possible - a tuple $\langle s(x^*) \prec_{\#} t(x^*), \Delta(x^*)\rangle$ denoting a theorem $\forall x^*\ \Delta(x^*) \rightarrow s(x^*) \prec_{\#} t(x^*)$. Thus, we are left with a proof of $\forall x^*\ \Phi(x^*) \rightarrow \Delta(x^*)$ within a first-order theorem prover in order to establish $\Phi(x^*) \rightarrow s(x^*) \prec_{\#} t(x^*)$.
Walthers' approach has severe limitations. Since it is based on the definition of $\prec_{\#}$, firstly, it is restricted to datatypes possessing the unique-factorization property and secondly, it is only able to detect the termination of algorithms whose recursive calls decrease according the count-ordering $\prec_{\#}$.
Now consider the following example from a case study in specification and verification of an access control system of a nuclear power plant which has been done within the verification support environment VSE [HLS+96]. The area of this plant is divided into several sections and each of these sections possesses a specific security level. In addition each member of the staff has a specific clearance denoted by the security level of rooms he is allowed to enter. The relation of persons to their clearance is specified by a look-up table *rights*. Applying *max* to *rights* returns the highest security-level adjoined to anyone in *rights*, and $levelp(x, l, z)$ implements the look-up whether a person x has a specific clearance l wrt. the look-up table z.
Once a person has clearance for a security-level n he also has automatically admittance to all rooms with security-levels less than n. Thus, we define the function *accp* denoting whether a person x may enter a room with security-level l wrt. the look-up table z as follows:

$$\begin{aligned}&\mathsf{function}\ accp(x\!:\!person, l\!:\!nat, z\!:\!rights) : bool \equiv \\ &\qquad \mathsf{if}\ s(max(z)) - l = 0\ \mathsf{then}\ false \\ &\qquad \mathsf{if}\ s(max(z)) - l \neq 0\ \mathsf{then}\ levelp(x, l, z) \vee accp(x, s(l), z)\end{aligned} \tag{1}$$

Neither of the arguments of *accp* becomes $\prec_{\#}$-smaller in the recursive call. Thus, Walthers' approach is not able to detect the termination of *accp*. In NQTHM [BM79] the idea that is used to prove the termination of *accp* is to introduce a so-called *measure function* $\mathbf{m}$ and to prove that the measure of the recursive call $\mathbf{m}(x, s(l), z)$ is $\prec_{\#}$-smaller than the measure $\mathbf{m}(x, l, z)$ of the original call. By choice of $\mathbf{m} = \lambda u, v, w.s(max(w)) - v$ the termination of *accp* is guaranteed by a proof of

$$s(max(z)) - l \neq 0 \rightarrow s(max(z)) - s(l) \prec_{\#} s(max(z)) - l \tag{2}$$

[2] which we call $\mathcal{R}_\Gamma$ throughout the paper

But problems arise, how to find an appropriate measure $\mathbf{m}$ automatically, and how to prove the termination of the algorithm wrt. this measure.
While in [BM79] the selection of appropriate measures has to be done by a user, we propose an extension of Walters' approach that synthesizes measure-functions for termination proofs automatically and that integrates the estimation-calculus as a black-box procedure.
Walthers' approach does not rely on an explicit representation of the ordering $\prec_{\#}$. Yet the property $a \prec_{\#} b$ can be reformulated in a higher-order setting to:

$$\exists \mathbf{F}\ a = \mathbf{F}(b) \wedge \Delta_{\mathbf{F},1}(b) \tag{3}$$

with the additional condition that $\vdash_{\Gamma} \langle a \preceq_{\#} \mathbf{F}(b), \Delta_{\mathbf{F},1}(b)\rangle$ holds, i.e. we are able to deduce this tuple within the estimation calculus Γ. Thus, the termination of *accp* can be expressed by

$$\begin{aligned}
&\exists \mathbf{m}\ \exists \mathbf{F}\ \forall x : person\ \forall l : nat\ \forall z : rights \\
&\qquad s(max(z)) - l \neq 0 \rightarrow \mathbf{m}(x, s(l), z) = \mathbf{F}(\mathbf{m}(x, l, z), x, l, z) \wedge \\
&\qquad s(max(z)) - l \neq 0 \rightarrow \Delta_{\mathbf{F},1}(\mathbf{m}(x, l, z), x, l, z) \\
&\quad \text{where} \quad \vdash_{\Gamma} \langle \mathbf{F}(\mathbf{m}(x, l, z), x, l, z) \prec_{\#} \mathbf{m}(x, l, z), \Delta_{\mathbf{F},1}(\mathbf{m}(x, l, z), x, l, z)\rangle
\end{aligned}$$

In the following we present a method to deal with these kind of proof obligations that is based on rippling technics. During the automated proof of the above termination formula, appropriate instantiations of the variables $\mathbf{m}$, $\mathbf{F}$ and $\Delta_{\mathbf{F},1}$ that guarantee the corresponding algorithm to be terminating will be computed.

2 The Estimation Calculus

As already mentioned, Walthers' approach [Wal94] is based on the count-ordering $\prec_{\#}$ on freely generated datatypes. The so-called *estimation calculus* Γ is used to incorporate knowledge on $\prec_{\#}$ and depends on the actual axiomatization and deals with tuples $\langle s[x^*] \preceq_{\#} t[x^*], \Delta[x^*]\rangle$. If such a tuple can be derived within the estimation calculus, i.e.

$$\vdash_{\Gamma} \langle q[x^*] \preceq_{\#} r[x^*], \Delta[x^*]\rangle \tag{4}$$

then it holds that

$$\forall x^*\ q[x^*] \preceq_{\#} r[x^*] \wedge \forall x^*\ \Delta[x^*] \leftrightarrow q[x^*] \prec_{\#} r[x^*]. \tag{5}$$

Using the calculus-rules in reverse direction, we obtain a rewrite-system $\mathcal{R}_{\Gamma}$ which, given a problem $q[x^*] \prec_{\#} r[x^*]$, either computes a predicate $\Delta[x^*]$ satisfying (5) (wrt. the actual axiomatization) or it fails, denoting that the relation $q[x^*] \preceq_{\#} r[x^*]$cannot be established between both terms. This rewrite-system possesses some nice properties: it is locally finite and Noetherian which implies that $\vdash_{\Gamma} \langle q[x^*] \prec_{\#} r[x^*], \Delta[x^*]\rangle$ is decidable.
The estimation-calculus relies on the notion of *p-bounded* functions. A function is p-bounded if $\forall x_1, \ldots, x_n\ f(x_1, \ldots, x_n) \preceq_{\#} x_p$ holds. In [Wal94] a technique is presented to inspect an algorithmic function definitions for f in order to analyze

whether f is p-bounded. In case the method recognizes f to be p-bounded, a so-called *p-difference predicate* $\Delta_{f,p}$ is automatically synthesized with the property that

$$\forall x_1, \ldots, x_n \; \Delta_{f,p}(x_1, \ldots, x_n) \leftrightarrow f(x_1, \ldots, x_n) \prec_{\#} x_p \tag{6}$$

holds. Thus, the estimation calculus is especially effective for proofs of $\vdash_\Gamma \langle q[t] \preceq_{\#} t, \Delta[x] \rangle$ where the second argument t is a subterm of the first argument $q[t]$. In this case it has to be checked whether all the functions occurring on the way between to level of $q[t]$ to some occurrence of t are bounded in the respective arguments.
In order to illustrate the use of this approach when analyzing the termination of algorithms consider the following example. Suppose we define subtraction of two natural numbers by:

$$\begin{aligned} &\mathsf{function} - (X : nat, Y : nat) : nat \equiv \\ &\quad \mathsf{if}\ Y = 0\ \mathsf{then}\ X \\ &\quad \mathsf{if}\ Y \neq 0\ \mathsf{then}\ p(X - p(Y)) \end{aligned} \tag{7}$$

In order to prove its termination we have to ensure that either the first or the second argument of $-$ within the recursive call is $\prec_{\#}$-smaller than the corresponding formal parameter. Using the estimation calculus we obtain $\langle p(Y) \preceq_{\#} Y, \Delta_{p,1}(Y) \rangle$ for the second argument. Thus, we have to prove that $\forall Y : nat \, Y \neq 0 \rightarrow \Delta_{p,1}(Y)$ which is true since $\Delta_{p,1}$ is defined by $\Delta_{p,1}(X) \leftrightarrow X \neq 0$.
Walthers' approach is a black-box procedure. There is no explicit representation of $\prec_{\#}$ on the object level, i.e. within the logical database of the underlying theorem prover. The reasoning about $\prec_{\#}$ is done completely within the estimation-calculus and not within the inference machine of the theorem prover which makes it difficult to incorporate measure functions into the basic inference-mechanism.

3 Computing Measure-functions

Rather than integrating specific measure functions into the meta-level algorithm - i.e. estimation-calculus - we want to obtain a maximal flexibility by defining measure-functions inside the database of the theorem-prover. This allows a user to specify and to prove properties about measures explicitly within the system. But doing so, we also want to make use of the implicit knowledge about $\prec_{\#}$ as it is built implicitly into the estimation-calculus.
Suppose we define a function f by case analysis and recursion and let

$$\mathsf{if}\ \Phi(x^*)\ \mathsf{then}\ h(\ldots f(t^*[x^*]) \ldots)$$

be one of these cases. Then, an appropriate measure $\mathbf{m}$ has to be found such that the recursive call $t^*[x^*]$ is less than x^* with respect to $\mathbf{m}$ and $\prec_{\#}$, i.e. $\Phi(x^*) \rightarrow \mathbf{m}(t^*[x^*]) \prec_{\#} \mathbf{m}(x^*)$. In a next step we will eliminate the explicit occurrence of $\prec_{\#}$ since Walthers' approach does not require an explicit representation of $\prec_{\#}$ and thus, $\prec_{\#}$ is not explicitly defined within the system. In order to prove a property $a \preceq_{\#} b$ we search for an appropriate function $\mathbf{F}$ such that $a = \mathbf{F}(b)$ and $\mathbf{F}$ is 1-bounded. Thus, we obtain a higher-order equality problem $\exists \mathbf{F} \; a = \mathbf{F}(b)$. Since $\mathbf{F}$

has to be 1-bounded we have to check the property that $\vdash_\Gamma \langle \mathbf{F}(y) \prec_\# y, \Delta_{\mathbf{F},1}(y)\rangle$ holds which is done with the help of the system $\mathcal{R}_\Gamma$.
In case we want to prove $a \prec_\# b$, additionally $\Delta_{\mathbf{F},1}(b)$ has to be established under the given conditions. Thus, in general we have to prove the following property[3] to guarantee that the recursive call $t^*[x^*]$ is less than x^* in case of $\Phi(x^*)$:

$$\begin{aligned}
&\exists \mathbf{m}\ \exists \mathbf{F}\ \forall x^* \\
&\qquad \Phi(x^*) \rightarrow \mathbf{m}(t^*[x^*]) = \mathbf{F}(\mathbf{m}(x^*), x^*) \wedge && (8)\\
&\qquad \Phi(x^*) \rightarrow \Delta_{\mathbf{F},1}(\mathbf{m}(x^*), x^*) && (9)\\
&\quad \text{where} \quad \vdash_\Gamma \langle \mathbf{F}(\mathbf{m}(x^*), x^*) \prec_\# \mathbf{m}(x^*), \Delta_{\mathbf{F},1}(\mathbf{m}(x^*), x^*)\rangle && (10)
\end{aligned}$$

For example, when proving the termination of *accp* we obtain the following proof obligation:

$$\begin{aligned}
&\exists \mathbf{m}\ \exists \mathbf{F}\ \forall x : \mathit{person}\ \forall l : \mathit{nat}\ \forall z : \mathit{rights} \\
&\qquad s(\mathit{max}(z)) - l \neq 0 \rightarrow \mathbf{m}(x, s(l), z) = \mathbf{F}(\mathbf{m}(x, l, z), x, l, z) \wedge && (11)\\
&\qquad s(\mathit{max}(z)) - l \neq 0 \rightarrow \Delta_{\mathbf{F},1}(\mathbf{m}(x, l, z), x, l, z) && (12)\\
&\quad \text{where} \quad \vdash_\Gamma \langle \mathbf{F}(\mathbf{m}(x, l, z), x, l, z) \prec_\# \mathbf{m}(x, l, z), \Delta_{\mathbf{F},1}(\mathbf{m}(x, l, z), x, l, z)\rangle && (13)
\end{aligned}$$

Proving these kind of theorems will result in appropriate instantiations of $\mathbf{F}$, $\Delta_{\mathbf{F},1}$ and $\mathbf{m}$ which guarantee the algorithm under consideration to be terminating. Therefore we split the proof of the termination-theorem into three parts which will be solved successively:

1. In a first step we tackle part (8) with rippling-techniques in order to obtain an appropriate instantiation of $\mathbf{F}$. This process will also unveil constraints on possible instantiations of $\mathbf{m}$

2. Secondly, we use the instantiation of $\mathbf{F}$ to solve (10) with the help of the system $\mathcal{R}_\Gamma$ which results in an instantiation of $\Delta_{\mathbf{F},1}$.

3. Finally, we use the information computed in the steps before to instantiate (9) and to compute appropriate instantiations of $\mathbf{m}$.

3.1 Instantiation of F

In order to prove part (8) we have to solve the equation

$$\mathbf{m}(t^*[x^*]) = \mathbf{F}(\mathbf{m}(x^*), x^*)$$

Shading the differences between left- and right-hand side results in the following colored equation

$$\mathbf{m}(\boxed{t^*[x^*]}) = \boxed{\mathbf{F}(\mathbf{m}(x^*), x^*)}$$

In order to prove this equation we have to manipulate the left-hand side until the differences - the so-called wave-fronts - occur on the top of the non-shaded

[3]Note that $\mathbf{F}$ has additional parameters corresponding to the all-quantified variables in the context of its occurrence

parts of the left-hand side (the so-called *skeleton*). This can be done by applying rippling-out equations [BSvH+93, Hut90] of the form

$$f(\ldots g(\ldots x \ldots) \ldots) = h(\ldots f(\ldots x \ldots) \ldots) \tag{14}$$

which will step by step move the wave-fronts towards the top-level. The techniques to perform these manipulations are known as rippling and are typically used within inductive proofs to enable the use of induction hypothesis inside the induction conclusion. Once the wave-fronts have reached the top-level, the left-hand side is an instance of the right-hand side and we may instantiate $\mathbf{F}$ to the respective top-level wave-front.
However, the problem of instantiating $\mathbf{F}$ can be solved by *rippling* the wave-front t^* in front of $\mathbf{m}$. In addition to standard rippling, we have to obey the constraints given by (10), i.e. that $\mathbf{F}$ has to be 1-bounded. Thus, we admit only rippling-out equations of form (14) which have the property that the right-hand side of the equation is less or equal wrt. $\preceq_\#$ than its skeleton, i.e.

$$\vdash_\Gamma \langle h(\ldots f(\ldots x \ldots) \ldots) \prec_\# f(\ldots x \ldots), \Psi \rangle$$

holds for some Ψ.[4]
In the example of proving the termination of *accp* we have to prove

$$\mathbf{m}(x, s(l), z) \quad = \quad \mathbf{F}(\mathbf{m}(x, l, z), x, l, z) \tag{15}$$

A rippling-out equation has to deal with a wave-front $s(\ldots)$ on the left-hand side, and the wave-front on the right-hand side has to obey the restrictions concerning argument-bounded functions. Thus, for instance the following annotated equations from the definitions of $+$ and $\times$ are *not* appropriate

$$s(X) + Y = s(X + Y) \quad , \qquad s(X) \times Y = Y + (X \times Y)$$

since for both rippling-out equations the top-level wave-front functions s and $+$ are not argument-bounded and thus, the system $\mathcal{R}_\Gamma$ fails. However the rippling-out equation created from the definition of $-$

$$X - s(Y) \quad = \quad p(X - Y) \tag{16}$$

satisfies the restrictions to admissible top-level wave-fronts, since using the system $\mathcal{R}_\Gamma$ we can establish:

$$\vdash_\Gamma \langle p(x) \prec_\# x, x \neq 0 \rangle. \tag{17}$$

Thus, this very selective procedure gives rise to the approach of instantiating an appropriate $\mathbf{m}$ and $\mathbf{F}$ "on the fly", i.e. during the rippling process. In our problem of proving the equation (15) we use (16) to ripple-out one of the wave-fronts on the left-hand side. Unification of the left-hand side of (16) and the subterm $\mathbf{m}(x, s(l), z)$ results in a single colored unifier

$$\{\mathbf{m} \leftarrow \lambda u, v, w.\mathbf{m}_1(u, v, w) - v, X \leftarrow \mathbf{m}_1(x, s(l), z), Y \leftarrow l\}.$$

[4]This property reduces significantly the branching rates during the rippling process.

Applying the instantiated colored equation on (15) we obtain the modified equation

$$p(\mathbf{m}_1(x, s(l), z) - l) = \mathbf{F}(\mathbf{m}_1(x, l, z) - l, x, l, z) \tag{18}$$

At this point the rippling-out process stops because one of the following criteria is satisfied:

1. There is a non-empty, top-level wave-front on the left-hand side which can be used as a possible instantiation of $\mathbf{F}$. If non-instantiated measure-functions, like $\mathbf{m}$ or $\mathbf{m}_1$, still occur on the left-hand side, there is at least one of its arguments without any wave-front occurring inside. In the example above both, the first and the third arguments of $\mathbf{m}_1$ contain no wave-fronts.

2. There is no top-level wave-front on the left-hand side but the function $\mathbf{m}$ has an argument $t_i[x_i]$ with $\vdash_\Gamma \langle t_i[x_i] \prec_\# x_i, \Psi(x_i)\rangle$. E.g. when proving the termination of $-$ we have to solve the equation

$$\mathbf{m}(x, p(y)) = \mathbf{F}(\mathbf{m}(x, y), x, y) \tag{19}$$

   In this case no rippling-process is required at all, since due to (17) the criterion is satisfied wrt. to the second argument of $\mathbf{m}$.

3.2 Instantiation of $\Delta_{\mathbf{F},1}$

Once we have finished the rippling process, we use the obtained equation in order to determine an appropriate instantiation of $\mathbf{F}$.
In case we finished the rippling process because of the first criterion, we identify $\mathbf{F}$ with the non-empty, top-level wave-front. Thus, we obtain $\mathbf{F} \leftarrow \lambda u, v, w, z.p(u)$ as the appropriate instantiation from (18).
In case the second criterion was responsible for the rippling process to stop, we identify the measure function with the projection to the argument in which the argument-bounded wave-front occurs. Thus, in case of (19) we choose $\mathbf{m} \leftarrow \lambda u, v.v$ which simplifies this equation to $p(y) = \mathbf{F}(y, x, y)$. Now, analogously to the first case, $\mathbf{F}$ is identified with the non-empty, top-level wave-front which in this example results again in $\mathbf{F} \leftarrow \lambda u, v, w, z.p(u)$.
In both cases we obtain an instantiation of $\mathbf{F}$ which is used to determine the corresponding $\Delta_{\mathbf{F},1}$ with the help of the system $\mathcal{R}_\Gamma$. In both examples $\mathcal{R}_\Gamma$ results in the derivation

$$\vdash_\Gamma \langle p(x) \prec_\# x, x \neq 0\rangle.$$

Thus, we instantiate $\Delta_{\mathbf{F},1}$ by $\lambda x.x \neq 0$ and obtain in case of *accp* the following simplified proof-obligation:

$$\exists \mathbf{m}_1\ \forall x : person\ \forall l : nat\ \forall z : rights$$
$$s(max(z)) - l \neq 0 \rightarrow p(\mathbf{m}_1(x, s(l), z) - l) = p(\mathbf{m}_1(x, l, z) - l) \wedge \tag{20}$$
$$s(max(z)) - l \neq 0 \rightarrow \mathbf{m}_1(x, l, z) - l \neq 0 \tag{21}$$

3.3 Instantiation of m

In order to select an appropriate instantiation of the remaining measure function(s) we concentrate on the second part (9) of the proof obligation, i.e. that the conditions of the recursive case have to imply the instantiated difference predicate $\Delta_{\mathbf{F},1}$. For instance, in case of *accp* we have to prove the formula (21). Possible instantiations are computed by unifying $\mathbf{m}_1(x,l,z) \neq 0$ with some appropriate axiom or condition of the definition case. This suggests to use the condition $s(max(z)) - l \neq 0$ which results in an instantiation $\{\mathbf{m}_1 \leftarrow \lambda u, v, w.s(max(w))\}$. The set of possible instantiations is restricted by equation (20): the instantiation of $\mathbf{m}_1$ must not use any of its arguments, in which still wave-fronts occur. Thus, in our example any instantiation of $\mathbf{m}_1$ that makes use of its second argument is not admissible since in this case (20) is not trivially true after the instantiation of $\mathbf{m}$.
Summing up, we use rippling-out techniques to move the wave-fronts inside the left-hand side of the equation to the top-level. During this process we allow only wave-fronts at top-level which dominate the skeleton by argument-bounded functions. According to different criteria the rippling process stops and $\mathbf{F}$ is instantiated by the top-level wave-front on the left-hand side. Using this instantiation we are able to compute $\Delta_{\mathbf{F},1}$ with the help of the estimation-calculus. Finally, we instantiate the measure function $\mathbf{m}$ by projections to the "wave-front free" arguments. Inspecting the above proofs, there is almost no search. While in inductive proofs rippling-out already turned out to be a very restrictive search strategy, in this application this process is further restricted by the constraints concerning the admissibility of top-level wave-fronts.

4 Lexicographical Orderings

In this section we consider algorithms with multiple recursive calls.
As an example in our scenario, suppose there are visitor-groups within the nuclear power plant. Hence, persons are either visitors or members of the staff. We define a function *visitorp* on persons which returns 1[5] if the person is a visitor and 0 else. A visitor as a member of a guided tour inherits the access rights from the guide which is a member of the staff. Thus, we specify a function *guide* on *person* which yields the guide of the person in case he is a visitor, while members of the staff are their own guides

$$\forall X : person\ visitorp(X) = 0 \rightarrow guide(X) = X \tag{22}$$

$$\forall X : person\ visitorp(guide(X)) = 0$$

Incorporating visitor tours to the function *accp* results in the following specification:

$$\mathsf{function}\ accp(x : person, l : nat, z : rights) : bool \equiv$$
$$\mathsf{if}\ s(max(z)) - l = 0\ \mathsf{then}\ false$$

[5] which is denoted by $s(0)$

$$\begin{array}{l}\mathsf{if}\ s(max(z)) - l \neq 0 \wedge visitorp(x) = 0\\ \quad \mathsf{then}\ levelp(x,l,z) \vee accp(x, s(l), z)\\ \mathsf{if}\ s(max(z)) - l \neq 0 \wedge visitorp(x) = 1\\ \quad \mathsf{then}\ accp(guide(x), l, z)\end{array}$$

In contrast to the previous sections, the definition of *accp* contains two recursive calls $accp(x, s(l), z)$ and $accp(guide(x), l, z)$. In order to prove the termination of *accp* we will use a lexicographical ordering on different measure functions tailored to the different calls.
Suppose, $t_1^*[x^*], \ldots, t_n^*[x^*]$ are the recursive calls of an algorithm which are governed by the corresponding conditions $\Phi_1(x^*), \ldots, \Phi_n(x^*)$, then we have to find an appropriate sequence of measure functions $\mathbf{m}^1, \ldots, \mathbf{m}^k$ and a mapping τ which relates each recursive call $t_i^*[x^*]$ to an appropriate measure function $\mathbf{m}^{\tau(i)}$. Then, we have to prove the following theorem for each recursive call $t_i^*[x^*]$:

$$\begin{array}{ll}\exists \mathbf{m}^j\ \exists \mathbf{F}^i\ \forall x^* & \\ \qquad \Phi_i(x^*) \rightarrow \mathbf{m}^j(t_i^*[x^*]) = \mathbf{F}^i(\mathbf{m}^j(x^*), x^*) \wedge & \\ \qquad \Phi_i(x^*) \rightarrow \Delta_{\mathbf{F}^i,1}(\mathbf{m}^j(x^*), x^*) & \\ \quad \text{with} \quad \vdash_\Gamma \langle \mathbf{F}^i(\mathbf{m}^j(x^*), x^*) \prec_\# \mathbf{m}^j(x^*), \Delta_{\mathbf{F}^i,1}(\mathbf{m}^j(x^*), x^*)\rangle & \end{array}$$

In the example of *accp* we are looking for appropriate measure functions $\mathbf{m}^1$ and $\mathbf{m}^2$ and argument-bounded functions $\mathbf{F}^1$ and $\mathbf{F}^2$ such that the following formulas hold:

$$\begin{array}{ll}\exists \mathbf{m}^1\ \exists \mathbf{F}^1\ \forall x\!:\!person\ \forall l\!:\!nat\ \forall z\!:\!rights & (23)\\ \qquad s(max(z)) - l \neq 0 \wedge visitorp(x) = 0 & \\ \qquad\qquad \rightarrow \mathbf{m}^1(x, s(l), z) = \mathbf{F}^1(\mathbf{m}^1(x,l,z), x, l, z) \wedge & \\ \qquad s(max(z)) - l \neq 0 \wedge visitorp(x) = 0 \rightarrow \Delta_{\mathbf{F}^1}(\mathbf{m}^1(x,l,z), x, l, z) & \\ \quad with \quad \vdash_\Gamma \langle \mathbf{F}^1(\mathbf{m}^1(x,l,z), x, l, z) \prec_\# \mathbf{m}^1(x,l,z), \Delta_{\mathbf{F}^1,1}(\mathbf{m}^1(x,l,z), x, l, z)\rangle & \end{array}$$

and

$$\begin{array}{ll}\exists \mathbf{m}^2\ \exists \mathbf{F}^2\ \forall x\!:\!person\ \forall l\!:\!nat\ \forall z\!:\!rights & (24)\\ \qquad s(max(z)) - l \neq 0 \wedge visitorp(x) = 1 & \\ \qquad\qquad \rightarrow \mathbf{m}^2(guide(x), l, z) = \mathbf{F}^2(\mathbf{m}^2(x,l,z), x, l, z) \wedge & \\ \qquad s(max(z)) - l \neq 0 \wedge visitorp(x) = 1 \rightarrow \Delta_{\mathbf{F}^2}(\mathbf{m}^2(x,l,z), x, l, z) & \\ \quad with \quad \vdash_\Gamma \langle \mathbf{F}^2(\mathbf{m}^2(x,l,z), x, l, z) \prec_\# \mathbf{m}^2(x,l,z), \Delta_{\mathbf{F}^2,1}(\mathbf{m}^2(x,l,z), x, l, z)\rangle & \end{array}$$

In order to determine an appropriate measure function $\mathbf{m}^1$ we start with formula (23) and analogously to the previous section we obtain the solution

$$\{\mathbf{m}^1 \leftarrow \lambda u, v, w.s(max(w)) - v, \mathbf{F}^1 \leftarrow \lambda u, v, w.p(u), \Delta_{\mathbf{F}^1,1} \leftarrow \lambda u, v, w.u \neq 0\}$$

In order to keep the set of different measure functions as small as possible we now test whether some other recursive calls decrease according to the measure function $\mathbf{m}_1$. Thus, we instantiate $\mathbf{m}^2$ by $\lambda u, v, w.s(max(w)) - v$ and try to

prove the theorems according to the procedure described in section 3. But $\mathbf{m}_1$ is not appropriate to establish the second theorem since $\mathbf{m}_1(guide(x), l, z) = max(z) - l = \mathbf{m}_1(x, l, z)$ holds. Thus, we have to find another measure according to which the second recursive call is getting smaller.
Again, we use our procedure described in section 3 and try to prove the second formula (24). We start with the rippling process of the colored equation:

$$\mathbf{m}^2(guide(x), l, z) = \mathbf{F}^2(\mathbf{m}^2(x, l, z), x, l, z) \tag{25}$$

In order to move the wave-front to the top-level we need a rippling-out equation with *guide* occurring in the wave-front of the left-hand side and some argument-bounded functions occurring in the wave-front of the right-hand side. The last condition rules out the colored version of equation (22) which does not possess any wave-front on the right-hand side. Instead we use the following colored equation

$$\forall X : person\ visitorp(X) = 1 \rightarrow visitorp(guide(X)) = p(visitorp(X)) \tag{26}$$

Unifying the left-hand side of (26) with the left-hand side of (25) we obtain the following substitution:

$$\{\mathbf{m}_2 \leftarrow \lambda u, v, w.visitorp(u), X \leftarrow x\}$$

and applying the instantiated equation results in the following formula

$$p(visitorp(x)) = \mathbf{F}^2(visitorp(x), x, l, z) \tag{27}$$

At this point the rippling process stops according to the first criterion given in paragraph 3.1 and $\mathbf{F}^2$ is instantiated by $\lambda u, v, w, z.p(u)$ which solves the colored equation (27). Using the system $\mathcal{R}_\Gamma$ results in an instantiation of $\Delta_{\mathbf{F}^2,1}$ to $\lambda u, v, w, z.u \neq 0$ such that we are left with the trivial problem of proving:

$$\begin{aligned}&\forall x : person\ \forall l : nat\ \forall z : rights\\ &\quad s(max(z)) - l \neq 0 \wedge visitorp(x) = 1 \rightarrow visitorp(x) \neq 0\end{aligned}$$

Thus, we have proved that the first recursive call of *accp* decreases according to the measure-function $\mathbf{m}^1$ while the second call decreases wrt. $\mathbf{m}^2$. In order to obtain a lexicographical ordering using both measure functions, we have to prove either that the first call stays invariant wrt. $\mathbf{m}^2$ or that the third call stays invariant wrt. to $\mathbf{m}_1$. Hence, we formulate the following proof obligation:

$$\begin{aligned}&\forall x : person\ \forall l : nat\ \forall z : rights\\ &\quad s(max(z)) - l \neq 0 \wedge visitorp(x) = 0 \rightarrow \mathbf{m}^2(x, s(l), z) = \mathbf{m}^2(x, l, z) \vee\\ &\quad s(max(z)) - l \neq 0 \wedge visitorp(x) = 1 \rightarrow \mathbf{m}^1(guide(x), l, z) = \mathbf{m}^1(x, l, z)\end{aligned}$$

Unfolding the definitions of $\mathbf{m}^1$ and $\mathbf{m}^2$ the formula is easy to prove since $\mathbf{m}^1$ does not depend on its first argument and thus, $\mathbf{m}^1(guide(x), l, z) = \mathbf{m}^1(x, l, z)$ is trivially true.

5 The Tautology Rule

As illustrated in the above examples, our approach to synthesize appropriate measure functions $\mathbf{m}$ is based on the rippling-out technique. In order to support the ripple-out process there is a need for a specific inference rule, the so-called *tautology-rule* which is used to re-annotate the colors of a term without changing the skeleton. A typical application of the tautology-rule is to manipulate a term $s(\boxed{f(s(a))})$ into $\boxed{s(f(s(a)))}$. The erasure of the equation $s(\boxed{f(s(X))}) = \boxed{s(f(s(X)))}$ needed to perform this manipulation is a tautology.
In a higher-order setting, as it is used in our approach, the tautology-rule interferes with the the use of higher-order variables like $\mathbf{m}$. Given for example a term $\mathbf{m}(\boxed{f(s(a))})$ we have to instantiate $\mathbf{m}$ by $\lambda u.\mathbf{m}_1(s(u))$ in order to apply the tautology rule on $\mathbf{m}_1(s(\boxed{f(s(a))}))$ which yields $\mathbf{m}_1(\boxed{s(f(s(a)))})$. Thus the tautology rule has to be generalized to cover the instantiation of higher-order variables by a wave-front occurring just right before the skeleton.
In order to demonstrate the use of this generalized tautology-rule we return to our example of the access-control system. Suppose, our look-up table *rights* is implemented as a set of security-levels. Each security-level l is a record which consists of a natural number $level(l)$, specifying a specific clearance, and a list of persons which possess this clearance. This fact is denoted by the following axioms:

$$\begin{aligned}
&\forall l : seclevel\ l = mksec(level(l), pers(l)) \\
&\forall n : nat\ \forall p : plist\ level(mksec(n,p)) = n \\
&\forall n : nat\ \forall p : plist\ pers(mksec(n,p)) = p
\end{aligned} \tag{28}$$

Now, the following function *remove* deletes a person from a security-level (assuming that each person occurs at most once in person-list):

$$\begin{aligned}
&\mathsf{function}\ remove(x : person, l : seclevel) : seclevel \\
&\quad \mathsf{if}\ pers(l) = nil\ \mathsf{then}\ l \\
&\quad \mathsf{if}\ pers(l) \neq nil \wedge car(pers(l)) = x \\
&\qquad \mathsf{then}\ mksec(level(l), cdr(pers(l))) \\
&\quad \mathsf{if}\ pers(l) \neq nil \wedge car(pers(l)) \neq x \\
&\qquad \mathsf{then}\ mksec(level(l), \\
&\qquad\qquad cons(car(pers(l)), \\
&\qquad\qquad\quad pers(remove(x, mksec(level(l), cdr(pers(l))))))))
\end{aligned} \tag{29}$$

In order to prove the termination of *remove* we have to prove the following theorem:

$$\begin{aligned}
&\exists \mathbf{m}\ \exists \mathbf{F}\ \forall x : person\ \forall l : seclevel \\
&\qquad pers(l) \neq nil \wedge car(pers(l)) \neq x \\
&\qquad\quad \rightarrow \mathbf{m}(x, \boxed{mksec(level(l), cdr(pers(l)))}) = \boxed{\mathbf{F}(\mathbf{m}(x,l), x, l)} \wedge \\
&\qquad pers(l) \neq nil \wedge car(pers(l)) \neq x \rightarrow \Delta_{\mathbf{F},1}(\mathbf{m}(x,l), x, l) \\
&\quad \text{with} \quad \vdash_\Gamma \langle \mathbf{F}(\mathbf{m}(x,l), x, l) \prec_\# \mathbf{m}(x,l), \Delta_{\mathbf{F},1}(\mathbf{m}(x,l), x, l) \rangle
\end{aligned}$$

Again we start with the annotated equation and try to ripple-out the wave-front on the left-hand side:

$$\mathbf{m}(x, \mathit{mksec}(\mathit{level}(l), \mathit{cdr}(\mathit{pers}(l)))) = \mathbf{F}(\mathbf{m}(x,l), x, l)$$

In order to push the rippling-out process we use the tautology-rule as described above and obtain by instantiating $\mathbf{m}$ to $\lambda u, v.\mathbf{m}_1(u, \mathit{pers}(v))$:

$$\mathbf{m}_1(x, \mathit{pers}(\mathit{mksec}(\mathit{level}(l), \mathit{cdr}(\mathit{pers}(l))))) = \mathbf{F}(\mathbf{m}_1(x, \mathit{pers}(l)), x, l)$$

However, the usage of this tautology-rule must be restricted to specific situations: either there is a colored equation applicable, which removes the wave-fronts moved towards the top-level by this rule; or the wave-front moved outside is itself argument-bounded. In our example we can eliminate the wave-front by using a colored version of (28):

$$\mathbf{m}_1(x, \mathit{cdr}(\mathit{pers}(l))) = \mathbf{F}(\mathbf{m}_1(x, \mathit{pers}(l)), x, l)$$

According to the second condition given in section 3 the rippling-process stops and $\mathbf{m}_1$ is instantiated to $\lambda u, v.v$ which results in:

$$\mathit{cdr}(\mathit{pers}(l)) = \mathbf{F}(\mathit{pers}(l), x, l)$$

As usual, we instantiate $\mathbf{F}$ by the wave-front of the left-hand side $\lambda u.\mathit{cdr}(u)$ and using $\mathcal{R}_\Gamma$ we obtain $\vdash_\Gamma \langle \mathit{cdr}(x) \prec_\# x, x \neq \mathit{nil} \rangle$ Therefore we instantiate $\Delta_{\mathbf{F},1}$ by $\lambda u.u \neq \mathit{nil}$ and obtain the trivial task of proving:

$$\forall x : \mathit{person}\ \forall l : \mathit{seclevel}\ (\mathit{pers}(l) \neq \mathit{nil} \wedge \mathit{car}(\mathit{pers}(l)) \neq x) \rightarrow \mathit{pers}(l) \neq \mathit{nil}$$

which finishes the termination proof of *remove*.

6 Practical Results

This approach depends on the existence of appropriate rippling-out equations and of course on the existence of user defined functions necessary to synthesize appropriate measure functions (by higher-order unification). Hence, the user may improve the behavior of the system by specifying additional functions or lemmata giving a kind of hint how to synthesize an appropriate measure function.
The approach described in this paper has been successfully implemented in the inductive theorem proving system INKA [HS96]. INKA is itself integrated into an environment for the formal development of software, "Verification Support Environment" (VSE), [HLS+96], in order to handle first-order proof obligations. During the execution of various industrial case-studies many algorithms had to be proved to be terminating. However, for some of these algorithms Walthers approach failed since they were either specified on non-freely generated datatypes or descend according to some non-standard ordering. In these cases we could successfully prove their termination using our approach either completely automatically or with the help of the user who submitted the system appropriate rippling-out equations (as lemmata) in order to guide the instantiation of $\mathbf{m}$. As it turned out, performing these case-studies the time spent to prove the termination of the algorithms is usually less than a second. This includes also the backtracking of the rippling-out process in case the instantiation of measure functions fails.

7 Related Works

Based on Walthers estimation-calculus, Sengler [Sen96] has recently extended this approach to non-freely generated datatypes. Since these datatypes do not possess the unique factorization property he defines the size of an object as the minimal number of (reflexive) constructor-symbols necessary to denote an object. His approach is mainly limited to datatypes which only consist of so-called size increasing constructors c, i.e. $x_p \preceq_{\#} c(x_1, \ldots, x_n)$ holds for all reflexive argument positions of c. He introduces an estimation-calculus similar to [Wal94] which is also used to recognize p-bounded algorithms. Similar to [Wal94] his approach is only able to prove the termination of algorithm which terminate according to the count-ordering.
Notice, that our approach does not rely on a specific ordering but is based on a "black-box procedure" detecting whether $s \prec t$ wrt. some well-founded ordering $\prec$ holds. Thus, we are free to use the approach of [Sen96] as the underlying mechanism instead of [Wal94] if we want to prove the termination of algorithms operating on non-freely generated datatypes.
In NQTHM [BM79] there is an explicit representation of $\prec_{\#}$ within the database (using the size function # and $<_{Nat}$). In order to prove the termination of an algorithm the user has to support the system with appropriate induction lemmata $\Phi(x^*) \rightarrow m(t[x^*]) \prec_{\#} m(x^*)$ which introduce possible measure functions m. Hence, the user has explicitly to specify an appropriate measure function in order to prove the termination of the algorithm under consideration.
In the field of Knuth-Bendix based approaches there has been innumerable work (e.g. [Der87, Ste95]) defining well-founded orderings on terms. These orderings are used to direct equations of the axiomatization, and thus to guide the rewriting of a term t to a term t' such that t is equal to t' under the given theory and t' is a minimal wrt. the ordering. In our setting we are interested in orderings $\prec$ on the semantic objects and thus, $\prec$ is independent of a specific syntactic representation of a term, i.e. $\forall a, b, c, d\ a = b \wedge c = d \rightarrow (a \prec c) \rightarrow (b \prec d)$ holds.
Selecting a unique syntactic representation for each semantic object, a syntactical ordering $<$ can be used to order their representations. Then, two terms are compared by comparing their representatives wrt. $<$. For instance in case of freely-generated datatypes the constructor-terms can be used as representatives. Following this idea, Giesl [Gie95] uses arbitrary polynomial term-orderings to prove the termination of algorithmic definitions which overcomes the limitations of the fixed count-ordering used in [Wal94]. The problem of proving the termination is encoded into a set of inequalities, the solution of which specifies the appropriate measure function. The approach uses Collins' cylindric algebraic decomposition algorithm to solve the set of in-equations. Since the size of the set of inequalities to be solved corresponds to the size of the axiomatization, it is not clear how Collins' algorithm behaves on large axiomatizations obtained for instance by industrial case-studies. Furthermore, Giesl' approach is restricted to freely-generated datatypes and polynomial orderings.

8 Conclusion

In this paper we have presented a method to incorporate Walthers' approach of proving the termination of algorithms as a "black-box procedure" into explicit proofs about termination using measure functions. Since we do not use internal knowledge of the estimation calculus, we can also use Senglers' approach (or even other approaches based on any well-founded orderings) as an underlying procedure to deal with problems like, $\Phi(x^*) \rightarrow s[x^*] \prec_{\#} t[x*]$. Thus, we inherit both, the flexibility of an explicit representation of the termination proof and the in-built knowledge about the count-ordering. The practical results obtained so far are very promising.

References

[BHHW86] Susanne Biundo, Birgit Hummel, Dieter Hutter, and Christoph Walther. The Karlsruhe Induction Theorem Proving System. In Jörg H. Siekmann, editor, *Proceedings 8th International Conference on Automated Deduction (CADE), LNCS 230*, Oxford, England, 1986. Springer-Verlag.

[BM79] R. S. Boyer and J Strother Moore. *A Computational Logic*. Academic Press, London, England, 1979.

[BSvH+93] Alan Bundy, Andrew Stevens, Frank van Harmelen, Andrew Ireland, and Alan Smaill. Rippling: a heuristic for guiding inductive proofs. *Artificial Intelligence, North Holland*, 62:185–253, 1993.

[Der87] Nachum Dershowitz. Termination of rewriting. *J. Symbolic Computation*, 3(1&2):69–115, February/April 1987. Corrigendum: *4*, 3 (December 1987), 409–410; reprinted in *Rewriting Techniques and Applications*, J.-P. Jouannaud, ed., pp. 69—115, Academic Press, 1987.

[Gie95] Jürgen Giesl. *Automatsierung von Terminierungsbeweisen für rekursiv definierte Algorithmen*. PhD thesis, TH Darmstadt, 1995. Infix-Verlag, Bd. 96.

[GTW78] J.A. Goguen, J.W. Thatcher, and E.G. Wagner. An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types. In R.T. Yeh, editor, *Current Trends in Programming Methodology*. Prentice Hall, 1978.

[HLS+96] D. Hutter, B. Langenstein, C. Sengler, J. Siekmann, W. Stephan, and A. Wolpers. Deduction in verification support environment (vse). In *Proceedings Formal Method Europe, LNCS 1051*, Oxford, Great Britain, 1996. Springer-Verlag.

[HS96] Dieter Hutter and Claus Sengler. INKA - The Next Generation. In M.McRobbie J.Slaney, editor, *13th International Conference on Automated Deduction (CADE), LNCS* , 1996.

[Hut90] Dieter Hutter. Guiding induction proofs. In Mark E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction (CADE), LNAI 449*, pages 147–161, Kaiserslautern, Germany, July 1990. Springer-Verlag.

[Sen96] Claus Sengler. Termination of algorithms over non-freely generated data types. In J.K.Slaney M.A. Mc Robbie, editor, *Proceedings 13th International Conference on Automated Deduction (CADE), LNAI 1104*, pages 121–135, New Brunswick, USA, July 1996. Springer-Verlag.

[Ste95] Joachim Steinbach. Simplification orderings — history of results. *Fundamenta Informaticae*, September 1995.

[Wal94] C. Walther. on proving the termination of algorithms by machine. *Artificial Intelligence, North Holland*, 71:101–157, 1994.

Using Rippling to Prove the Termination of Algorithms

Dieter Hutter

RR 97-03
Research Repor