On Automating Inductive and Non-Inductive Termination Methods

Fairouz Kamareddine and François Monin

Department of Computing and Electrical Engineering,
Heriot-Watt University,
Edinburgh EH14 4AS, Scotland,
fairouz@cee.hw.ac.uk, monin@cee.hw.ac.uk

Abstract. The *Coq* and *ProPre* systems show the automated termination of a recursive function by first constructing a tree associated with the specification of the function which satisfies a notion of *terminal property* and then verifying that this construction process is formally correct. However, those two steps strongly depend on inductive principles and hence *Coq* and *ProPre* can only deal with the termination proofs that are inductive. There are however many functions for which the termination proofs are non-inductive. In this article, we attempt to extend the class of functions whose proofs can be done automatically à la *Coq* and *ProPre* to a larger class including functions whose termination proofs are not inductive. We do this by extending the terminal property notion and replacing the verification step above by one that searches for a decreasing measure which can be used to establish the termination of the function.

1 Introduction

Termination is an important property in the verification of programs defined on recursive data structure that use automated deduction. While the problem is undecidable, several proof methods of termination of functions have been developed using for instance formal proof methods in functional programming or orderings of rewriting systems. For instance we mention polynomial interpretations [3, 9, 21], recursive path orderings [14] and Knuth-Bendix orderings [7, 12]. The latter methods are characterized by orderings called simplification orderings [5, 18] and deal with the termination of functions called *simply terminating functions*. Some functions that are *non-simply terminating* can be proven to terminate with methods based on structural inductive proofs because they focus on recursive functions which can be viewed as *sorted constructor systems* that allow reasoning on the orderings' structure of the data objects.

But there are other recursive functions that are non-simply terminating for which inductive methods fail to prove the termination since precisely the structural orderings on the terms of the algebra cannot be used. These functions, which we call, *non-inductive-simply terminating functions* form an important class of functions used in recursive data structures and hence, automatically establishing their termination is an important property. This is witnessed by the

recent literature where techniques coming from rewriting [6, 1, 2, 8] have been proposed to automate the termination of such functions.

We are interested in automating the termination (inductive or non-inductive) of recursive functions in a theorem proving framework. We choose the framework of the system called *ProPre* developed in [17, 15, 16] which is also the one used in Coq [4]. *ProPre* is devoted to the termination of recursively defined functions. The *Coq* and *ProPre* systems show the automated termination of a recursive function by first constructing a tree associated with the specification of the function which satisfies a notion of *terminal property* and then verifying that the process of constructing such a tree is formally correct. The search of such trees, from which it is also possible to extract decreasing measures [19, 10] through the recursive call of the function, relies in particular on the structure of multi-sorted algebras and hence automated termination proofs in *Coq* and *ProPre* strongly depend on inductive principles. This means that *Coq* and *ProPre* can only deal with the termination proofs that are inductive.

Our aim is to extend the *Coq* and *ProPre* approaches to deal with automated non-inductive termination proofs. To do this, we introduce a new notion of terminal state property which has an algorithmic content that enables the method to be automated as an inductive one. From each tree that enjoys the terminal state property we associate an ordinal measure that couldn't be previously obtained from the *ProPre* system [19, 10] and we show the decreasing property that ensures in this way the termination of the recursive function. As a consequence, the technique allows inductive methods to go further in the proof search when *natural* structural orderings are not enough to achieve the proof.

The paper is divided as follows: In Section 2, we set out the formal machinery. In Section 3, we introduce *ProPre* notion of terminal state property and our own extension of it. We show that our extension strictly includes the *ProPre* notion and establish in Theorem 1 that if a distributing tree $\mathcal{A}$ has the terminal state property in the system *ProPre*, then $\mathcal{A}$ has the new terminal state property and that the opposite does not hold. In Section 4, we explain how it is possible to define ordinal measures against trees of functions where if the ordinal measure decreases in the recursive call of the function, then this function terminates. We recall the ramified measures that come from the analysis of the *ProPre* system and we give new measures which will help in establishing terminations of functions where the proofs of terminations are non-inductive. Our main theorem of this section (Theorem 2) establishes that our new notion of terminal state and our extended notion of measures, enable us to establish the termination of functions (inductive and non inductive).

2 Preliminaries

2.1 Constructor systems

In this paper we deal with constructor systems and more precisely with sorted constructor systems. The following standard definitions are needed.

Definition 2.1. We assume a set $\mathcal{F}$ of *function symbols*, called signature, and a set S of *sorts*. To each function $f \in \mathcal{F}$ we associate a natural number n that denotes its *arity* and a *type* $s_1, \ldots, s_n \to s$ with $s, s_1, \ldots, s_n \in S$. A function is called constant if its arity is 0.
We assume that the set of functions $\mathcal{F}$ is divided in two disjoint sets $\mathcal{F}_c$ and $\mathcal{F}_d$. Functions in $\mathcal{F}_c$ (which also include the constants) are called *constructor symbols* or *constructors* and those in $\mathcal{F}_d$ are called *defined symbols* or *defined functions*.

Definition 2.2. Let $\mathcal{X}$ be a set of *variables* disjoint from $\mathcal{F}$. We assume that each variable of $\mathcal{X}$ has a unique sort and that for each sort s there is a countable number of variables in $\mathcal{X}$ of sort s. If s is a sort, F and X are respectively sets included in $\mathcal{F}_c \cup \mathcal{F}_d$ and $\mathcal{X}$, then $\mathcal{T}(F, X)_s$ is the smallest set such that:

1. every element of X of sort s is a term of sort s,
2. if $t_1, \ldots, t_n$ are terms of sorts $s_1, \ldots, s_n$ respectively, and if f is a function of type $s_1, \ldots, s_n \to s$, then $f(t_1, \ldots, t_n)$ is a term of sort s.

If X is empty, we denote $\mathcal{T}(F, X)_s$ by $\mathcal{T}(F)_s$ whose elements are called ground terms. If the arity of c is 0, the *constant term* $c()$ is also denoted c. $\mathcal{V}ar(t)$ denotes the set of variables that occur in the term t, and $\mathcal{P}os(t)$ is the set of positions of t. If s and t are terms and q is a position of t, then the term $t[s]_q$ is the term t in which the term s is now at position q.

Definition 2.3. A *(sorted) equation* is a pair $(l, r)_s$ of terms l and r of a sort s, which is also called *rewrite rule* and written $l \to r$. A set of (sorted) equations is *non overlapping* iff no left-hand sides unify each other.

Definition 2.4. A *specification* or *constructor system* $\mathcal{E}$ of a function $f : s_1, \ldots, s_n \to s$ in $\mathcal{F}_d$ is a non overlapping set of left-linear equations $\{(e_1, e'_1)_s, \ldots, (e_p, e'_p)_s\}$ such that for all $1 \leq i \leq p$, e_i is of the form $f(t_1, \ldots, t_n)$ with $t_j \in \mathcal{T}(\mathcal{F}_c, \mathcal{X})_{s_j}$, $j = 1, \ldots, n$, and $e'_i \in \mathcal{T}(\mathcal{F}_c \cup \mathcal{F}_d, \mathcal{X})_s$.

Definition 2.5. Let $\mathcal{E}$ be a specification of a function f with type $s_1, \ldots, s_n \to s$. A *recursive call* of f is a pair $(f(t_1, \ldots, t_n), f(u_1, \ldots, u_n))$ where $f(t_1, \ldots, t_n)$ is a left-hand side of an equation of $\mathcal{E}$ and $f(u_1, \ldots, u_n)$ is a subterm of the corresponding right-hand side.

2.2 The term distributing trees

We give some ingredients that will be needed in the next sections. A term distributing tree of a specification is a tree whose root can be seen as an uplet of distinct variables, each node matches its children and each leaf corresponds to a left-hand side of an equation. More precisely we have:

Definition 2.6. Let $\mathcal{E}$ be a specification of a function $f : s_1, \ldots, s_n \to s$. $\mathcal{A}$ is a *term distributing tree* for $\mathcal{E}$ iff it is a tree such that:

1. its root is of the form $f(x_1, \ldots, x_n)$ where x_i is a variable of sort s_i, $i \leq n$,

2. each left-hand side of an equation of $\mathcal{E}$ is a leaf of $\mathcal{A}$ (up to variable renaming)
3. each node $f(t_1, \ldots, t_n)$ of $\mathcal{A}$ admits one variable x' of a sort s' such that the set of children of the node is (for $x'_1, \ldots, x'_r$ are not in $t_1, \ldots t_n$): $\{f(t_1, \ldots t_n)[C(x'_1, \ldots x'_r)/x'], C : s'_1, \ldots, s'_r \rightarrow s' \in \mathcal{F}_c\}$.

Notation 2.7. Let $\mathcal{A}$ be a term distributing tree. A branch $\mathcal{B}$ from the root θ_1 to a leaf θ_k is denoted by $(\theta_1, x'_1), \ldots, (\theta_{k-1}, x'_{k-1}), \theta_k$ where θ_1 is the root, θ_k is the leaf, and for each $i \leq k-1$, x'_i is the variable x' for the node θ_i in the third clause of Definition 2.6.

It can be easily seen, according to Definition 2.6, that we have the following:

Fact 2.8. Let $\mathcal{E}$ be a specification of a function f of type $s_1, \ldots, s_n \rightarrow s$ and $\mathcal{A}$ be a term distributing tree for $\mathcal{E}$.

1. For each $(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}_c)_{s_1} * \ldots * \mathcal{T}(\mathcal{F}_c)_{s_n}$ there exists one and only one leaf θ of $\mathcal{A}$ and a ground constructor substitution ρ such that $\rho(\theta) = f(t_1, \ldots, t_n)$.
2. For every branch of $\mathcal{A}$ from the root to a leaf $(\theta_1, x_1), \ldots, (\theta_{k-1}, x_{k-1}), \theta_k$ and for all $i \leq j \leq k$, there exists a constructor substitution $\sigma_{j,i}$ such that $\sigma_{j,i}(\theta_i) = \theta_j$. The substitutions $\sigma_{j,i}$ may also be written as $\sigma_{\theta_j, \theta_i}$

We introduce here a well-founded ordering relation on the terms.

Definition 2.9. Assume a function m on the terms ranging over natural number that is closed under substitutions, i.e. $m(u) > m(v)$ implies $m(\sigma(u)) > m(\sigma(v))$ for all ground substitution σ. Let $u, v \in \mathcal{T}(\mathcal{F}, \mathcal{X})_s$ for a given sort s. We say that $u \sqsubset v$ iff u is linear and $m(u) < m(v)$ with $\mathcal{V}ar(u) \subseteq \mathcal{V}ar(v)$.

3 Generalizing the Coq termination procedure and the ProPre system

The analysis of the termination proofs using the `Recursive Definition` of the Coq assistant and the *ProPre* system shows that writing a formal proof in these systems can be regarded as the search of a term distributing tree that enjoys a terminal state property. That is to say, if a formal tree for a specification of a function can be built having a terminal state property then the function terminates. We define in this section a new terminal state property for a term distributing tree generalizing that of `Recursive Definition` procedure and *ProPre*. We first give some notations that will be used in the rest of the paper.

Notation 3.1. Let $\mathcal{A}$ be a term distributing tree for a specification. If t is the left-hand side of an equation, $b(t)$ will denote the branch in the term distributing tree that leads to the term t. If b is a branch, then L_b will denote the leaf of the branch b. Note that b and $b(t)$ may denote two distinct branches.
If a node θ matches a term u of a recursive call (t, u), then the substitution will be denoted by $\rho_{\theta,u}$.

As every function that terminates with the procedure of the Coq assistant also terminates with the *ProPre* system, we do not give here the property in the setting of the `Recursive Definition` but only for the extended version of the *ProPre* system. Note that it is actually devised in a different way from below in [16]. However it has been shown [11] that for each distributing tree defined in [16] that enjoys the terminal state property of [16] there is a corresponding term distributing tree of Definition 2.6 that has the following property (Definition 3.2) which is more convenient for our purpose.

Definition 3.2. Let $\mathcal{A}$ be a term distributing tree for a specification. We say that $\mathcal{A}$ has the *terminal state property* (*tsp*) if there is an application $\mu : \mathcal{A} \to \{0,1\}$ on the nodes of $\mathcal{A}$ such that if L is a leaf, $\mu(L) = 0$, and for all recursive call (t,u), there is a node (θ, x) in the branch $b(t)$ with $\mu(\theta) = 1$ such that θ matches u with $\rho_{\theta,u}(x) \sqsubset \sigma_{L_{b(t)},\theta}(x)$ and for all ancestor (θ', x') of θ in $b(t)$ with $\mu(\theta') = 1$, we have $\rho_{\theta',u}(x') \sqsubseteq \sigma_{L_{b(t)},\theta'}(x')$.

As already mentioned, if a specification $\mathcal{E}$ of a function admits a term distributing tree that has the terminal state property, then $\mathcal{E}$ is terminating [17].
The rest of this section is devoted to define a new terminal state property generalizing the previous one. We first need to introduce fresh variables as follows. For each position q and sort s, we will assume there is a new variable of sort s indexed by q and distinct from those of $\mathcal{X}$.

Definition 3.3. Let t be a term and q be a position. The term $[\![t]\!]_q$ is defined as follows: $[\![x]\!]_q = x$ if x is a variable, $[\![C(t_1,\ldots,t_n)]\!]_q = C([\![t_1]\!]_{q\cdot 1},\ldots,[\![t_n]\!]_{q\cdot n})$ if $C \in F_c$, and $[\![g(t_1,\ldots,t_n)]\!]_q = x_q$ if $g \in F_d$.

For a term $u = g(u_1,\ldots,u_n)$ and a substitution φ, $g(\varphi[\![u]\!])$ will denote the term $g(\varphi([\![u]\!]_1),\ldots,\varphi([\![u]\!]_n))$.

We introduce the following relations: For u, v in $\mathcal{T}(\mathcal{F}_c, \mathcal{X})_s$, we will say that $u \trianglerighteq v$ if $u \not\sqsubset v$ with $\neg$(b) or $\neg$(c) or $m(v) < m(u)$ in Definition 2.9; and we will say that $u \preccurlyeq v$ if $u \not\sqsubset v$ with (b) and (c) and $m(u) = m(v)$. We will use the so-called size measure $|\,.\,|_\#$ for the mentioned measure m. In the following definitions of the section we will consider a function $f : s_1,\ldots,s_n \to s$, a specification or a split specification $\mathcal{E}$, and a term distributing tree $\mathcal{A}$ of $\mathcal{E}$.

Definition 3.4. For each node θ, C_θ will denote $\{b \in \mathcal{A}, \theta \in b\}$ and $\mathcal{R}_\theta$ the set of recursive call (t,u) such that $b(t) \in C_\theta$. If (t,u) is a recursive call, then $\mathcal{M}_\mathcal{A}(u) = \{b \in \mathcal{A}, \exists\varphi,\varphi' \text{ such that } f(\varphi[\![u]\!]) = \varphi'(f(L_b))\}$ and $\mathcal{Q}_\mathcal{A}(t,u) = \{\theta \in b(t), \exists\sigma, \sigma(f(\theta)) = u\}$.

Note that the set $\mathcal{Q}_\mathcal{A}(t,u)$ is not empty since the root node belongs to $\mathcal{Q}_\mathcal{A}(t,u)$. Let b be a branch and two nodes $\theta, \theta' \in b$, we say that $\theta < \theta'$ if θ is *closer* than θ' to the root (i.e. if θ is an ancestor of θ'). So we can write $\mathcal{N}_\mathcal{A}(t,u) = \max\mathcal{Q}_\mathcal{A}(t,u)$.

For each node θ of $\mathcal{A}$ we assume an associated subset $\mathcal{G}_\theta$ of $\mathcal{R}_\theta$ which will be made explicit in Definition 3.7. Notice that the definitions below should be given simultaneously but are introduced separately to ease the readability.

Definition 3.5. Let (θ, x) be a node of $\mathcal{A}$ and $\mathcal{G}_\theta$ be a subset of $\mathcal{R}_\theta$. For each recursive call (t, u) of $\mathcal{G}_\theta$ such that $\theta \in \mathcal{Q}_\mathcal{A}(t, u)$, we assume that one of the two following cases below holds and we define $\xi^\theta_{(t,u)}$, as follows:

1. If $\rho_{\theta,u}(x) \sqsubset \sigma_{L_{b(t)},\theta}(x)$ or $\rho_{\theta,u}(x) \trianglerighteq \sigma_{L_{b(t)},\theta}(x)$, then $\xi^\theta_{(t,u)} = 1$,
2. If $\rho_{\theta,u}(x) \preccurlyeq \sigma_{L_{b(t)},\theta}(x)$, then $\xi^\theta_{(t,u)} = 0$.

The meaning of the above definition and the following one is to give decreasing criteria extending those of Definitions 2.9 and 3.2. It relies in particular on the *hierarchical structure* of the trees.

Definition 3.6. Let (θ, x) be a node of $\mathcal{A}$ and $\mathcal{G}_\theta$ be a subset of $\mathcal{R}_\theta$. For each recursive call (t, u) of $\mathcal{G}_\theta$ such that $\theta \in \mathcal{Q}_\mathcal{A}(t, u)$ and for each branch $b \in \mathcal{C}_\theta$, we will define $\eta^\theta_{(t,u),b}$ in the following way:

1. First take all (t, u) such that $\rho_{\theta,u}(x) \trianglerighteq \sigma_{L_{b(t)},\theta}(x)$, and let for all $b \in \mathcal{C}_\theta$:

$$\eta^\theta_{(t,u),b} = \begin{cases} 0 & \text{if } b \in \mathcal{M}_\mathcal{A}(u), \\ 1 & \text{if not.} \end{cases}$$

2. Next, consider each (t, u) in $\mathcal{G}_\theta$ such that there is a (t', u') with $\eta^\theta_{(t',u'),b(t)} = 0$, and for which no $\eta^\theta_{(t,u),b'}$ is already defined for any $b' \in \mathcal{C}_\theta$. Then also take

$$\eta^\theta_{(t,u),b} = \begin{cases} 0 & \text{if } b \in \mathcal{M}_\mathcal{A}(u), \\ 1 & \text{if not.} \end{cases}$$

3. Finally if item 2 cannot be applied, put $\eta^\theta_{(t,u),b} = 1$ for each $b \in \mathcal{C}_\theta$.

Notice that the cases 1 and 2 are made distinct in the above definition as the value $\xi^\theta_{(t,u)}$ is algorithmically defined; namely case 1 is the initial case.

We define, for each node θ of $\mathcal{A}$ and each left-hand side t of an equation where θ with $b(t) \in C_\theta$, $\eta^\theta_t = \prod_{\substack{(t',u') \in \mathcal{G}_\theta \\ \theta \in \mathcal{Q}_\mathcal{A}(t',u')}} \eta^\theta_{(t',u'),b(t)}$ if $\mathcal{G}_\theta \neq \emptyset$, and $\eta^\theta_t = 0$ if not.

We now explicit the subset $\mathcal{G}_\theta$ of $\mathcal{R}_\theta$ for a node θ. The following states whether from each node, a recursive call can be eliminated from a set of recursive calls:

Definition 3.7. Let θ_1 be the root of the recursive distributing tree $\mathcal{A}$. We first put $\mathcal{G}_{\theta_1} = \mathcal{R}_{\theta_1}$. Now assume that $\mathcal{G}_\theta$ is defined for a node θ of $\mathcal{A}$ and let θ' be a child of θ with θ' in $\mathcal{A}$. The set $\mathcal{G}_{\theta'}$ is then defined as follows: $(t, u) \in \mathcal{G}_{\theta'}$ iff $(t, u) \in \mathcal{R}_{\theta'} \cap \mathcal{G}_\theta$ and $(\xi^\theta_{(t,u)}, \eta^\theta_t) \neq (1, 1)$.

Now, we define F which is a necessary condition for the termination statement.

Definition 3.8. Let θ be a node of associated tree $\mathcal{A}$ of the recursive distributing tree $\mathcal{A}$ distinct from a leaf. We put $F(\theta) = 0$ if there is a child θ' of θ and (t, u) in $\mathcal{G}_{\theta'}$ such that $\theta > \mathcal{N}_\mathcal{A}(t, u)$; and we put $F(\theta) = 1$ if not.

Now the new terminal state property can be defined below.

Definition 3.9. The recursive distributing tree $\mathcal{A}$ is said to have the new terminal state property if for each node θ of $\mathcal{A}$ distinct from a leaf we have $F(\theta) = 1$ and for each branch b there is node θ' in b such that $\mathcal{G}_{\theta'} = \emptyset$.

We now come to Theorem 1 that states the above definition of the new terminal state property strictly includes the *ProPre* notion of terminal state property.

Theorem 1. *Let $\mathcal{E}$ be a specification of a function with a distributing tree $\mathcal{A}$. If $\mathcal{A}$ has the terminal state property in the system ProPre, then $\mathcal{A}$ has the new terminal state property. The opposite does not hold.*

A crucial point is of course to make sure that the new terminal state property leads a function to terminate. We prove this result in the next section by showing the existence of measures decreasing through the recursive calls of the functions. Note that there exist decreasing measures coming from the formal termination proofs in Coq or in *ProPre*. But in contrast with these measures, the new one, with the new terminal state property, will allow one to prove the termination functions that usually cannot be done with inductive methods.

4 Dealing with a non inductive method

A close notion to term distributing trees of a specification that has the terminal state property is the ramified measures. The measures coming from *Coq* or *ProPre* characterize in some sense the induction proofs made in the systems. We recall the ramified measures and explain why we need to introduce other measures to deal with termination that usually cannot be proven with inductive methods. Among these measures, a particular class is defined that is related to term distributing trees enjoying the new terminal state property and we show that they have the decreasing property. This, therefore, implies that the corresponding functions terminate. As a consequence, this provides a method of reasoning about termination of recursive functions where the underlying proofs rely on non-inductive as well as inductive axioms.

4.1 The ramified measures and the *ProPre* system

Definition 4.1. Let $\mathcal{A}$ be a tree and θ a node of $\mathcal{A}$. The *height* of θ in $\mathcal{A}$, denoted by $\mathcal{H}(\theta, \mathcal{A})$, is the height of the subtree of $\mathcal{A}$ whose root is θ minus one.

For a term distributing tree $\mathcal{A}$, we assume that for each node θ_i different from a leaf there is an application m_i that maps on natural numbers. The general form of ordinal measures introduced in [19] is given by the following

Definition 4.2. Let $\mathcal{E}$ be a specification of a function $f : s_1, \ldots, s_n \to s$, $\mathcal{A}$ be a term distributing tree for a specification of $\mathcal{E}$ and ω be the least infinite ordinal. The ramified measure $\Omega_{\mathcal{A}} : \mathcal{T}(F_c)_{s_1} * \ldots * \mathcal{T}(F_c)_{s_n} \to \omega^{\omega}$ is defined by: Let $t = (t_1, \ldots, t_n)$ be an element of the domain and θ be the leaf of $\mathcal{A}$ such that there is a substitution ρ with $\rho(\theta) = f(t)$ (Fact 2.8). Let $\mathcal{B}$ be the branch

$(\theta_1, x_1), \ldots, (\theta_{k-1}, x_{k-1}), \theta$ of $\mathcal{A}$ from the root to θ, let $\sigma_{r,s}$ be the substitutions of Fact 2.8 and for each θ_i the associated application m_i, $i \leq k-1$. Then

$$\Omega_{\mathcal{A}}(t) = \sum_{i=1}^{k-1} \omega^{\mathcal{H}(\theta_i, \mathcal{A})} * m_i(\rho(\sigma_{k,i}(x_i))) \ .$$

The ramified measures can be illustrated by Figures 1 and 4. An interesting subclass of the above measures is the class of R-*measures*. It has been shown that to each formal termination proof of recursive functions made with the `Recursive Definition` procedure of the Coq assistant, there is a distributing tree that has the terminal state property implying the decreasing property of the R-measure associated to the distributing tree [19]. This class of measures could be enlarged with I-*measures* [10] that can be related to a more efficient version of *ProPre* [16].

The functions m_i that occur in the definition of these measures belonging to the class of Definition 4.2 are directly supplied from formal proofs made in the system. This can be illustrated by Figures 2 and 3, where m is the parameterized function of Definition 2.9.

Definition 4.3. The *recursive length* lg of a term t of sort s is defined by:

1. if t is a constant or a variable, then $lg(t) = 1$,
2. if $t = C(t_1, \ldots, t_n)$ with $C : s_1, \ldots, s_n \to s$ then $lg(t) = 1 + \sum_{s_j = s} lg(t_j)$.

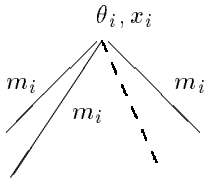

Fig.1. Ramified measures

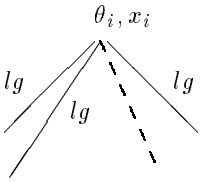

Fig.2. R-measures

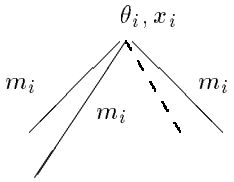

Fig.3. I-measures, $m_i \in \{m, \tilde{0}\}$

4.2 Extended ordinal measures

We motivate here the definition of new ordinals illustrated with some examples.

It is well known that the structural ordering is the most used among the well-founded orderings on natural numbers. As it is claimed in [20], other well-orderings on natural number are difficult to find automatically. As a simple illustration the constant function with value 0 is given in [20] that can be defined with the following specification $\mathcal{E}_1$.

$$f(0) \to f(s(0)); \quad f(s(0)) \to f(s(s(0)); \quad f(s(s(x)) \to 0. \tag{1}$$

Though a well-founded ordering is of course easy to find by a human in this case, it is however difficult to obtain one in an automated way since it is a non-simply terminating function and not suited to inductive methods. Note that proving at the same time the correctness of the specification (i.e. $f(x) = 0$)

and the termination seems not really relevant here as this is usually done with an ordering. Moreover the specification of the *quot* function given in this paper clearly shows that the correctness cannot be helpful in that case.

Note that there is no term distributing tree of $\mathcal{E}_1$ which has the terminal state property, but there is one that satisfies the new terminal state property.

The following example $\mathcal{E}_2$ of the function $evenodd : nat, nat \to Bool$ is borrowed from [1]. As mentioned in [1] the modifications of mutually recursive functions to obtain a function without mutual recursion leads to such specifications as that of *evenodd* below. We assume that *not* is already defined.

$$\begin{aligned} &evenodd(x, 0) \to not(evenodd(x, s(0))) \\ &evenodd(0, s(0)) \to false \\ &evenodd(s(x), s(0)) \to evenodd(x, 0). \end{aligned} \tag{2}$$

The second argument is used as a *flag* that enables *evenodd* to compute either the *even* function or the *odd* function. This function, which is a non simply terminating function, cannot be proven with usual inductive methods since precisely there is no *natural* orderings that can be used.

Consider the next example of specification of the function $quot : nat, nat, nat \to nat$, borrowed from T. Kolbe [13] and that can be found in [2].

The value of $quot(x, y, z)$ corresponds to $1 + \lfloor \frac{x-y}{z} \rfloor$ when $z \neq 0$ and $y \leq x$, that is to say $quot(x, y, y)$ computes $\lfloor \frac{x}{y} \rfloor$.

$$\begin{aligned} &quot(0, s(y), s(z)) \to 0 \\ &quot(s(x), s(y), z) \to quot(x, y, z) \\ &quot(x, 0, s(z)) \to s(quot(x, s(z), s(z)). \end{aligned} \tag{3}$$

The last rule shows that the specification is not simply terminating. The same rule also shows that the termination cannot be proven by usual inductions proofs.

It turns out that specification functions such as (1), the *evenodd* function (2) or the *quot* function (3) cannot be proven by the system *ProPre* and no R-measures neither I-measures [19, 10] have the decreasing property for any of these specifications. However, the following ordinal function

$\Omega(u, 0) = \omega * |u|_{\#} + 1, \quad \Omega(u, s(v)) = \omega * |u|_{\#},$

where $|\cdot|_{\#}$ is the *size* function, i.e. $|0|_{\#} = 1$, $|s(u)|_{\#} = 1 + |u|_{\#}$, reflects the specification of *evenodd* in the sense that it decreases in the recursive call of the function. There is also an ordinal measure below that has the decreasing property for the specification of the *quot* function

$\Omega(u, s(v), w) = \omega * |u|_{\#}, \quad \Omega(u, 0, w) = \omega * |u|_{\#} + 1.$

It would be possible to find a decreasing measure in the class of Definition 4.2 for (1), (2) or (3), but the choice of the m_i is difficult to obtain in an automated way. In particular we want to have m_i functions that are as simple as possible, such as for instance the size functions that are found in the above ordinal. Furthermore we would like to relate decreasing measures to term distributing trees that satisfy the new notion of terminal state property generalizing those

of Coq and *ProPre*. It turns out that such suitable measures actually belong to the extended ordinal measures defined below. We first introduce the following

Definition 4.4. Let $\mathcal{E}$ be a specification of a function $f : s_1, \ldots, s_n \to s$ such that there exists a term distributing tree $\mathcal{A}$ for $\mathcal{E}$. For each node θ_i of $\mathcal{A}$, we will assume that there are associated applications $m_{i,1}, \ldots, m_{i,j_i}$ that map on natural numbers whose number is equal to the number of the sub-branches starting from the node θ_i. Note that this number may be distinct from the number of the children of the node. These applications will be called *node measures*. If θ is a leaf of a branch where θ_i appears, we will also use $m_{\theta_i,\theta}$ to make explicit one of the node measures of θ_i when necessary.

Definition 4.5. Let $\mathcal{E}$ be a specification of a function $f : s_1, \ldots, s_n \to s$ such that there exists a term distributing tree $\mathcal{A}$ for $\mathcal{E}$. The extended measure $\Omega_{\mathcal{A}} : \mathcal{T}(F_c)_{s_1} * \ldots * \mathcal{T}(F_c)_{s_n} \to \omega^\omega$, is defined as follows:
Let $t = (t_1, \ldots, t_n)$ be an element of the domain and θ be the leaf of $\mathcal{A}$ such that there is a substitution ρ with $\rho(\theta) = f(t)$ (Fact 2.8). Let $\mathcal{B}$ be the branch $(\theta_1, x_1), \ldots, (\theta_{k-1}, x_{k-1}), \theta$ of $\mathcal{A}$ from the root to θ, let $\sigma_{r,s}$ be the substitutions of Fact 2.8. Then

$$\Omega_{\mathcal{A}}(t) = \sum_{i=1}^{k-1} \omega^{\mathcal{H}(\theta_i,\mathcal{A})} * m_{\theta_i,\theta}(\rho(\sigma_{k,i}(x_i))) \ .$$

An extended measure can be illustrated by Figures 5 and 6.

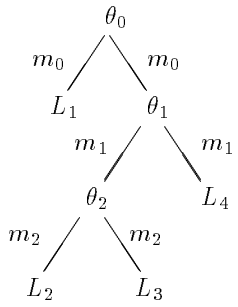

Fig.4. Term distributing with ramified measure

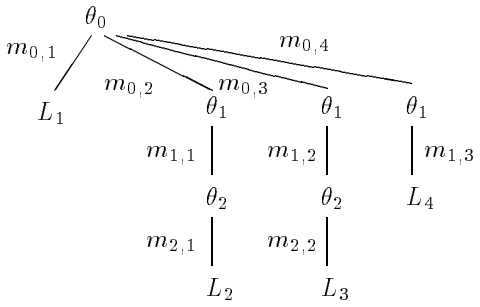

Fig.5. Term distributing with extended measure

For instance the specification $\mathcal{E}_1$ with the equations (1) in Section 4.2 admits a term distributing tree with an extended measure defined as follows

$\Omega_{\mathcal{A}}(0) = \omega * |0|_{\#}, \quad \Omega_{\mathcal{A}}(s(0)) = |0|_{\#}, \quad \Omega_{\mathcal{A}}(s(s(u))) = 0.$

This measure has obviously the decreasing property in the recursive calls of $\mathcal{E}_1$.

One may wonder whether the automation of decreasing measures belonging to Definition 4.5 is possible since we have to take account of the $m_{i,j}$ applications. We will show that it will be enough to consider a subclass of measures, called *hole-measures*, generalizing R- and I-measures. These measures will be associated to term distributing trees enjoying the new terminal state property. We will show that they have the decreasing property and that functions which admit a term distributing tree with the new terminal state property, are therefore terminating.

4.3 The hole-measures

Definition 4.6. Let $\mathcal{E}$ be a specification of a function $f : s_1, \ldots, s_n \to s$ such that there exists a term distributing tree $\mathcal{A}$ for $\mathcal{E}$. The *hole measure* $\Omega_{\mathcal{A}} : \mathcal{T}(F_c)_{s_1} * \ldots * \mathcal{T}(F_c)_{s_n} \to \omega^{\omega}$, is defined as follows:
Let $t = (t_1, \ldots, t_n)$ be an element of the domain and θ be the leaf of $\mathcal{A}$ such that there is a substitution ρ with $\rho(\theta) = f(t)$ (Fact 2.8). Let $\mathcal{B}$ be the branch $(\theta_1, x_1), \ldots, (\theta_{k-1}, x_{k-1}), \theta$ of $\mathcal{A}$ from the root to θ, let $\sigma_{r,s}$ be the substitutions of Fact 2.8. Then

$$\Omega_{\mathcal{A}}(t) = \sum_{i=1}^{k-1} \omega^{\mathcal{H}(\theta_i, \mathcal{A})} * (\eta_t^{\theta_i} * |(\rho(\sigma_{k,i}(x_i)))|_{\#}) \ .$$

That is to say $m_{\theta_i,\theta} = \eta_t^{\theta_i} * | \ . \ |_{\#}$.

Note that, due to the relation between the leaf θ and the term t, $\eta_t^{\theta_i} * | \ . \ |_{\#}$ depends both on θ_i and θ in the above definition.

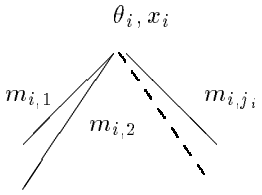

Fig.6. Extended measures

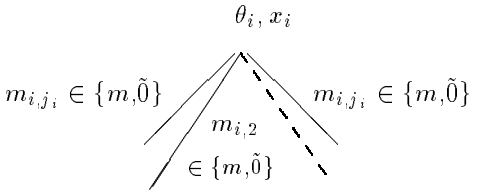

Fig.7. Hole-measures with $m_{i,j} \in \{m,\tilde{0}\}$

Now we come to the main theorem of this paper. It states that our new notion of new terminal state property and our extended notion of measures enable us to establish the inductive and non inductive termination of functions.

Theorem 2. *Let $\mathcal{E}$ be a specification of a function $f : s_1, \ldots, s_n \to s$ and $\mathcal{A}$ be a distributing tree $\mathcal{A}$ for $\mathcal{E}$ having the new terminal state property. The associated measure $\Omega_{\mathcal{A}}$ satisfies the decreasing property. I.e., for each recursive call $(f(t_1, \ldots, t_n), f(u_1, \ldots, u_n))$ of $\mathcal{E}$ and ground constructor substitution φ we have: $\Omega_{\mathcal{A}}(\varphi(t_1), \ldots, \varphi(t_n)) > \Omega_{\mathcal{A}}(\varphi([\![u_1]\!]_1), \ldots, \varphi([\![u_n]\!]_n))$.*

5 Conclusion

In this paper we have proposed a method that extends the automation of the proofs of termination of recursive functions used in *ProPre* and Coq. Whereas Coq and *ProPre* could only deal with the automation of inductive proofs, the method allows the automation of a larger class of recursive functions because non structural orderings can be handled by the method. The method is also a good vehicle for extending the automation of termination proofs of recursive functions to deal with issues not yet incorporated in theorem provers.

References

[1] T. Arts and J. Giesl. *Automatically proving termination where simplification orderings fail* in Proceeding TAPSOFT'97, LNCS, volume 1214, 261-272.

[2] T. Arts and J. Giesl. *Proving innermost normalisation automatically*in Proceeding RTA'97, LNCS, volume 1232, 157-171.

[3] A. Ben Cherifa and P. Lescanne. *Termination of rewriting systems by polynomial interpretations and its implementation.* Science of Computer Programming 9(2), 137-159, 1987.

[4] C. Cornes *et al.*. The Coq proof assistant reference manual version 5.10. *Technical Report 077*, INRIA, 1995.

[5] N. Dershowitz. Termination of rewriting. *Theoretical Computer Science 17*, 279-301, 1982.

[6] N. Dershowitz and C. Hoot. Natural termination. *Theoretical Computer Science* 142(2), 179-207, 1995.

[7] J. Dick, J. Kalmus and U. Martin. Automating the Knuth Bendix ordering. *Acta Informatica 28*, 95-119, 1990.

[8] T. Genet and I. Gnaedig. Termination proofs using gpo ordering constraints. *TAPSOFT*, LNCS 1214, 249-260, 1997.

[9] J. Giesl. *Generating polynomial orderings for termination proofs.* Proceedings of the 6th International Conference on Rewriting Techniques and Application, Kaiserlautern, LNCS, volume 914, 1995.

[10] F. Kamareddine and F. Monin. On formalised proofs of termination of recursive functions. In *Proc. International Conference on Principles and Practice of Declarative Programming*, Paris, France, 1999.

[11] F. Kamareddine and F. Monin. Induction Lemmas for Termination of Recursive Functions in a Typed System *Proc. submitted.*

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

[13] T. Kolbe. Challenge problems for automated termination proofs of term rewriting systems. Technical Report IBN96-42, Technische Hochshule Darmstadt, Alexanderstr. 10, 64283 Darmstadt, Germany, 1996.

[14] P. Lescanne. On the recursive decomposition ordering with lexicographical status and other related orderings. *Journal of Automated Reasoning* 6(1) 39-49, 1990.

[15] P. Manoury. *A User's friendly syntax to define recursive functions as typed lambda-terms.* Proceedings of Type for Proofs and Programs TYPES'94, LNCS, volume 996, 1994.

[16] P. Manoury and M. Simonot. *Des preuves de totalité de fonctions comme synthèse de programmes.* PhD thesis, University Paris 7, 1992.

[17] P. Manoury and M. Simonot. Automatizing termination proofs of recursively defined functions. *Theoretical Computer Science* 135(2) 319-343, 1994.

[18] A. Middeldorp and H. Zantema. Simple termination of rewrite systems. *Theoretical Computer Science 175*, 127-158, 1997.

[19] F. Monin and M. Simonot. An ordinal measure based procedure for termination of functions. To appear in *Theoretical Computer Science.*

[20] M. Parigot. Recursive programming with proofs. *Theoretical Computer Science* 94(2) 335-356, 1992.

[21] J. Steinbach. Generating polynomial orderings. *Information Processing Letters 49*, 85-93, 1994.