On Formalised Proofs of Termination of Recursive Functions *

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. In proof checkers and theorem provers (e.g. Coq [4] and *ProPre* [13]) recursive definitions of functions are shown to terminate automatically. In standard non-formalised termination proofs of recursive functions, a decreasing measure is sometimes used. Such a decreasing measure is usually difficult to find.
By observing the proof trees of the proofs of termination of recursive functions in *ProPre* (the system used in Coq's proofs of termination), [14] finds a decreasing measure which could be used to show termination in the standard non-formalised way. This is important because it establishes a method to find decreasing measures that help in showing termination.
As the *ProPre* system made heavy use of structural rather than inductive rules, an extended more powerful version has been built with new proof trees based on new rules.
In this article, we show that the ordinal measures found in [14] lose the decreasing property in the extended *ProPre* system and then, set out to show that the extended *ProPre* system will still be suitable for finding measures required by other systems (e.g. NQTHM). We do this by showing that exist other measures that can be associated to the proof trees developed in the extended *ProPre* system that respect the decreasing property. We also show that the new parameterised measure functions preserve the decreasing property up to a simple condition.

1 Introduction

In the verification of programs defined on recursive data structures, that use automated deduction, an important property is that of termination. A recursively defined function terminates if there is a well-founded order such that each recursive call of the function decreases with respect to this order. Though the termination problem is undecidable, several methods have been proposed for studying the termination of functional programs. For example, measures are used in the well-known NQTHM system of Boyer-Moore [2, 3], and in [6] the system can deal with measures based on polynomial norms. Though efficient,

* Supported by EPSRC GR/L15685.

these methods need however the measures to be given by the user. Other automated systems [18, 15, 19] have been developed, these are fully automated but they use only fixed ordering or a lexicographic combinations of the ordering.

Another approach has been developed in the termination procedure of the Coq prover [4] implemented in the *ProPre* system [11]. The method is automated and builds formal proofs because it is based on the Curry-Howard isomorphism from which lambda-terms are extracted which compute the algorithms. In contrast with other methods as for instance in [3, 6], a notion of right terminal state property for proof trees is introduced in the procedure instead of measures. It has been shown in [14] that once a termination proof is made, it is then possible to find a decreasing measure related to each proof tree. The measures characterize in some sense the orders found by the *ProPre* system (called *ramified measure*) which differ from the lexicographic combinations of one single fixed ordering. Moreover it has been shown that these measures could be automatically given for the NQTHM system.

However a difficult task for the system of [11] is to be able to establish the termination of the automated construction of the proof trees. A drawback of that system is that it is not easy to derive efficient rules in a formal context. More particularly, the method in [11] is restricted to one general structural rule and this implies the right terminal state property of proof trees to be limited.

To circumvent these drawbacks, the formal logical framework behind the method in [11], has been extended to give rise to a new system [12] using other rules and accommodated with a generalized induction principle. Furthermore an order decision procedure on terms has been introduced outside the proof trees that alleviate the search of right terminal state properties. As a consequence, the termination method can be used by the system in a far more efficient way and the class of formal termination proofs made in the system has been considerably enlarged.

The measures coming from the previous system can be also defined in the new system. But unfortunately they do not enjoy the decreasing property anymore. Therefore, the method of [14] cannot be used in the new *ProPre* system [12] to find suitable measures required by other systems such as NQTHM. We solve this problem in this paper by showing that there exist other measures that can be associated to the proof trees developed in the system respecting the decreasing property.

Moreover, the order decision procedure mentioned above, that is external to the formal proofs in the *ProPre* system, is based on the so-called size measure. So, this measure function could be easily changed or parameterised in the extended *ProPre* system. We also show that, up to a simple condition (Property 4.11), the decreasing property of measures will still hold.

Our work has the following advantages:

- We establish a method to find the measures needed to establish termination for recursive functions. We extend the system to a more powerful version while retaining the decreasing property of measures. This is important

because non-formalised termination proofs usually rely on the decreasing property.
- As the extended version of *ProPre* used the advantageous order decision procedure which was isolated from the formal proofs (in contrast to being intertwined with them as in the earlier version of *ProPre*), this implied that the measure functions could be easily parameterised or changed. In this paper we show that those measure functions preserve the decreasing property up to a simple condition. This means that the measures (now in a larger class) found by the method of this paper can be used by systems such as NQTHM.

2 Preliminaries

We assume familiarity with basic notions of type theory and term rewriting. The following definition contains some basic notions needed throughout the paper.

Definition 2.1.

1. **Sorts, Functions, Sorted Signature** We assume a set S of *sorts* and a finite set $\mathcal{F}$ of *function symbols* (or *functions*). We use s, s_1, s_2, ..., s', s'', ... to range over sorts and $f, f_1, f_2, \ldots, f', f'', \ldots$ to range over functions. A *sorted signature* is a finite set $\mathcal{F}$ of functions and a set S of sorts.
2. **Types, Arities of functions and Constants** For every function $f \in \mathcal{F}$, we associate a *type* $s_1, \ldots, s_n \to s$ with $s, s_1, \ldots, s_n \in S$. The number $n \geq 0$ denotes the arity of f. A function is called *constant* if its arity is 0.
3. **Defined and Constructor Symbols** 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*.
4. **Variables** Let $\mathcal{X}$ be a countable set of variables disjoint from $\mathcal{F}$. We assume that for every variable is associated a sort.
5. **Terms over F and $\mathcal{X}$ of sort s: $\mathcal{T}(F,\mathcal{X})_s$** If s is a sort, F is a subset of $\mathcal{F}$ and $\mathcal{X}$ is a certain set of variables, then the set of terms over F and $\mathcal{X}$ (simply called terms) of sort s denoted $\mathcal{T}(F,\mathcal{X})_s$, is the smallest set where:
   (a) every element of $\mathcal{X}$ of sort s is a term of sort s,
   (b) if $t_1, \ldots, t_n$ are terms of sort $s_1, \ldots, s_n$ respectively, and if f is a function of type $s_1, \ldots, s_n \to s$ in F, then $f(t_1, \ldots, t_n)$ is a term of sort s.

   We use $t, l, r, u, v, t_1, l_1, r_1, t_2, \ldots, t', l', r', t'', \ldots$ to range over $\mathcal{T}(F,\mathcal{X})_s$. If $\mathcal{X}$ is empty, we denote $\mathcal{T}(F,\mathcal{X})_s$ by $\mathcal{T}(F)_s$. $\mathcal{T}(F,\mathcal{X}) = \bigcup_{s\in S} \mathcal{T}(F,\mathcal{X})_s$.
6. **Constructor Terms, Ground terms and Ground Constructor Terms** Recall the set of variables $\mathcal{X}$ and the set of functions $\mathcal{F} = \mathcal{F}_c \cup \mathcal{F}_d$.
   (a) Elements of $\mathcal{T}(\mathcal{F}_c,\mathcal{X})_s$, i.e., terms such that every function symbol which occurs in them is a constructor symbol, are called *constructor terms*.
   (b) Elements of $\mathcal{T}(\mathcal{F}_c \cup \mathcal{F}_d)_s$, i.e., terms in which no variable occurs, are called *ground terms*.
   (c) Elements of $\mathcal{T}(\mathcal{F}_c)_s$ i.e., terms which do not have any variables and where every function symbol which occurs in them is a constructor symbol, are called *ground constructor terms*.

7. **(Sorted) Equations** A *sorted equation* is a pair $(l, r)_s$ of terms l and r of a sort s. We always assume that the equation is sorted and hence, we may drop the term *sorted* and speak only of equations. An equation $(l, r)_s$ gives rise to a *rewrite rule* $l \to r$. Although a pair $(l, r)_s$ is oriented it will also be written $l =_s r$. When no confusion occurs, the sort may be discarded from the equation and we write (l, r), $l \to r$ and $l = r$. l (resp. r) are called the left (resp. right) hand side of the equation.
8. **Left-Linear Equations** An equation is *left-linear* iff each variable occurs only once in the left-hand side of the equation.
9. **Non-Overlapping Equations** A set of equations is *non overlapping* iff no left-hand sides unify each other.
10. **Specification or Constructor System** A *specification* 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 $e'_i \in \mathcal{T}(\mathcal{F}_c \cup \mathcal{F}_d, \mathcal{X})_s$. We use $\mathcal{E}$, $\mathcal{E}'$, ... to range over specifications.
11. **{Constructor, Ground, Ground Constructor} Substitution** A substitution σ is a mapping from the set $\mathcal{X}$ of variables to the set of terms $\mathcal{T}(\mathcal{F}, \mathcal{X})$, such that for every variable x, $\sigma(x)$ and x are of the same sort. A substitution σ is called a *constructor substitution* (respectively *ground substitution*, *ground constructor substitution*) if $\sigma(x)$ is a constructor term (respectively ground term, ground constructor term) for any variable x.
12. **Recursive Call** 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 f and $f(u_1, \ldots, u_n)$ is a subterm of the corresponding right-hand side.

3 The extended *ProPre* system

The extended *ProPre* system deals with inductive types that are defined with second order formulas using first and second order universal quantification, implication and a general least fixed point operator on predicate variables. The last connective aims at improving the efficiency of the extracted programs (see [16]).

Unlike the previous system [11], a connector symbol $\restriction$ is added whose meaning is a connective conjunction used with some restrictions but without any algorithmic counterpart. The last property is interesting because it first allows the programs not to carry out some unnecessary computations, and secondly it can easily support inductive methods (which was not the case in the previous system). Combined with the connector $\restriction$, a binary relation symbol $\prec$ is added. It corresponds to a well-founded ordering on terms which is used for the inductive rule defined in the section.

Definition 3.1. The language is defined as follows:

1. **Terms** The terms of Definition 2.1.6 constitute the first order part.

2. **Data Symbols** For each sort s_i is associated a unary second order predicate said also data symbol and denoted by D_{s_i} or D_i, whose meaning is: $t \in \mathcal{T}(\mathcal{F}_c)_{s_i}$ iff $D_{s_i}(t)$ holds.
3. **Formulae** A formula is built as follows:
   (a) if D is a data symbol and t is a term then $D(t)$ is a formula,
   (b) if A is a formula and x is a variable, then $\forall x A$ is a formula,
   (c) if A is a formula and u, v are terms, then $A \restriction (u \prec v)$ is a formula,
   (d) if A and B are formulas, then $A \to B$ is a formula.
   We use $A, B, P, F, F_1, F_2, \ldots$ to range over formulae.

Notation 3.2. We will use some convenient conventions:

1. $Du_{\preceq t}$ is a shorthand for $D(u) \restriction (u \prec t)$,
2. $\forall x A \to B$ denotes $\forall x(A \to B)$.
3. $F_1, \ldots, F_n \to F$ denotes $F_1 \to (F_2 \to \ldots \to (F_n \to F)) \ldots)$.
4. Let $P = F_1, \ldots, F_k, \forall x D'(x), F_{k+1}, \ldots, F_m \to D(t)$ be a formula, then $P_{-D'(x)}$ denotes the formula $F_1, \ldots, F_k, F_{k+1}, \ldots, F_m \to D(t)$.

Note that the later notation is correct as it will be used with Definition 3.4.

Definition 3.3. Let $f : s_1, \ldots, s_n \to s \in \mathcal{F}_d$. The *termination statement for* f is the formula: $\forall x_1(D_{s_1}(x_1) \to \ldots \to \forall x_n(D_{s_n}(x_n) \to D_s(f(x_1, \ldots, x_n))) \ldots)$, also written by Notation 3.2 as: $\forall x_1 D_1(x_1), \ldots, \forall x_n D_n(x_n) \to D(f(x_1, \ldots, x_n))$.

In the previous *ProPre* system, the proofs relied on two fundamental notions: the distributing trees and the right terminal state property. In the extended version, the distributing trees now include two new rules, said *Struct* and *Ind* rules defined in the section. The definition of the right terminal state property (Definition 3.8) is now more sophisticated due to the introduction of these rules.

The *ProPre* prover makes termination proofs, said I-proofs, with the help of some macro-rules (or *tactics*, or *derived rules*) of Natural Deduction for Predicate Calculus (see [9]). The set of the rules and the definition of I-proofs is described in [12]. Due to Proposition 3.9 below, we will only need here to define the *Struct*-rule and the *Ind*-rule which constitute the distributing trees in *ProPre*.

Although the earlier *ProPre* system can prove the termination of many algorithms, there are numerous interesting algorithms for whose there exist no proof trees. For instance, the example below illustrates that the use *Rec*-rule defined in [11] can lead to loss of efficiency. Let Tr be the sort *tree*, with the *leave* constant $le : Tr$ and the *branch* constructor $br : Tr, Tr \to Tr$. Consider the specification of the *flatten* function $flat : Tr \to Tr$ given by the following equations:

$$flat(le) = le$$
$$flat(br(le, a)) = br(le, flat(a))$$
$$flat(br(br(a_1, a_2), a)) = flat(br(a_1, br(a_2, a))).$$

While the specification cannot be proven to terminate in the previous system [11], the termination proof is now easily done in the extended system due to the new rules presented below. Note that a single ordering using for instance the size measure [18] is not sufficient for the termination proof because of the presence of the second recursive call. The $flatten$ can be proved to terminate using polynomial ordering [10], but these have to be given by the user [1]. Therefore methods have been developed in [5, 17] that aim at synthesising polynomial orderings.

We now introduce the rules that are used in the extended system. Let be given a sort s. We then consider all the constants $c_1, \ldots, c_p$ of type $:\rightarrow s$, and all the constructor functions $C_i : s_{i_1}, \ldots, s_{i_k} \rightarrow s$, $(i_k \geq 1)$, $i \leq q$, whose range is s. Note that the above distinction between constants and the other constructors just corresponds to a question of presentation. Let also $F(x)$ be a formula where x, of sort s, is free in F. Then:

1. $\Phi_{c_i}(F)$ denotes $F[c_i/x]$, $i \leq p$,
2. $\Phi_{C_i}(F)$ denotes $\forall x_{i_1} D_{i_1}(x_{i_1}), \ldots, \forall x_{i_k} D_{i_k}(x_{i_k}) \rightarrow F[C_i(x_{i_1}, \ldots, x_{i_k})/x]$, $i \leq q$, where $x_{i_1}, \ldots, x_{i_k}$ are not in F,
3. $\Psi_{C_i}(F)$ denotes $\forall x_{i_1} D_{i_1}(x_{i_1}), \ldots, \forall x_{i_k} D_{i_k}(x_{i_k}), \forall z(D z_{\prec C_i(x_{i_1}, \ldots, x_{i_k})} \rightarrow F[z/x]) \rightarrow F[C_i(x_{i_1}, \ldots, x_{i_k})/x])$, $i \leq q$, where $z, x_{i_1}, \ldots, x_{i_k}$ are not in F.

Definition 3.4. Let P be of the form $F_1, \ldots, F_k, \forall x D(x), F_{k+1}, \ldots, F_m \rightarrow D'(t)$. The induction rule for the sort s is a choice between the two following rules:

$$\frac{\Gamma \vdash \Phi_{c_i}(P_{-D(x)})\ i \leq p,\ \Gamma \vdash \Phi_{C_j}(P_{-D(x)})\ j \leq q}{\Gamma \vdash P} \quad Struct(x)$$

$$\frac{\Gamma \vdash \Phi_{c_i}(P_{-D(x)})\ i \leq p,\ \Gamma \vdash \Psi_{C_j}(P_{-D(x)})\ j \leq q}{\Gamma \vdash P} \quad Ind(x)$$

For instance the induction rule *Ind* on integers is:

$$\frac{\Gamma \vdash P_{-N(x)}(0) \quad \Gamma \vdash \forall y N(y), \forall z(N z_{\prec s y} \rightarrow P_{-N(x)}(z)) \rightarrow P_{-N(x)}(sy)}{\Gamma \vdash P} \quad Ind(x)$$

The *Struct* has to be considered as a reasoning by cases. The above rules lead the following

Definition 3.5. A formula F is called an *I-formula* iff F is of the form $H_1, \ldots, H_m \rightarrow D(f(t_1, \ldots, t_n))$ with D a data symbol and $f \in \mathcal{F}_d$ such that for all $i = 1, \ldots, m$, H_i is of the form either $\forall x D'(x)$ or $\forall z(D' z_{\prec u} \rightarrow F')$, with D' a data symbol, F' an I-formula and u a term.
Furthermore a formula of the above form $H_i = \forall z(D' z_{\prec u} \rightarrow F')$ is called *a restrictive hypothesis* of F.

Note that the above definition is a recursive definition whose initial case can be obtained with "$H_i = \forall x D'(x)$". The *heart* $C(F)$ of the formula F will denote the term $f(t_1, \ldots, t_n)$.

Though a restrictive hypothesis is not an I-formula, we will also say that H' is a restrictive hypothesis of another restrictive hypothesis $\forall z(D'z_{\prec s} \rightarrow F')$ if H' is a restrictive hypothesis of the I-formula F'. Finally $C(\forall z(D'z_{\prec s} \rightarrow F'))$ will be $C(F')$.

Definition 3.6. Let $\mathcal{E}$ be a specification of a function f of type $s_1, \ldots, s_n \rightarrow s$. $\mathcal{A}$ is a *distributing tree for* $\mathcal{E}$ iff $\mathcal{A}$ is a proof tree built only with the *Struct* rule and *Ind* rule such that:

1. its root is $\vdash \forall x_1 D_1(x_1), \ldots, \forall x_n D_n(x_n) \rightarrow D(f(x_1, \ldots, x_n))$ (termination statement).
2. if $\mathcal{L} = \{\Gamma_1 \vdash \theta_1, \ldots, \Gamma_q \vdash \theta_q\}$ is the set of $\mathcal{A}$'s leaves, then there exists a one to one application b: $\mathcal{L} \hookrightarrow \mathcal{E}$ such that $b(L) = (t, u)$ if and only if $L = (\Gamma \vdash \theta)$ where θ is an I-formula with $C(\theta) = t$.

One can see that the antecedents remain unchanged in the definition of the rules *Struct* and *Ind* in the *ProPre* system. Though this is not so usual, it turns out that the antecedent formulas are embedded in the consequents. So, as the context (i.e. the set of antecedents) is empty in the root of a distributing tree, there is no antecedent in each node of the tree. Therefore we will use the notation θ both for $\vdash \theta$ and for the formula itself. One notes that any formula in a distributing tree is an I-formula.

Before stating the right terminal state property that enjoy the distributing trees in the I-proofs developed in the *ProPre* system, we assume that there is a well founded ordering $\sqsubset$ on term corresponding to the interpretation of the relation symbol $\prec$ defined in the language. This ordering is made explicit in the next section. We also need the

Definition 3.7. We say that an I-formula or restrictive hypothesis P can be applied to a term t if $C(P)$ matches t according to a substitution σ such that for each variable x occurring free in P we have $\sigma(x) = x$.

Definition 3.8. Let $\mathcal{E}$ be a specification of a function f and $\mathcal{A}$ be a distributing tree for $\mathcal{E}$. We say that $\mathcal{A}$ satisfies *the right terminal state property* (r.t.s.p.) iff for all leaves $L = \theta$ of $\mathcal{A}$ with $e \in \mathcal{E}$ the equation such that $b(L) = e$ (b given in Definition 3.6) and for all recursive calls (t, v) of e, there exists a restrictive hypothesis $P = \forall z D z_{\prec s}, H_1, \ldots, H_k \rightarrow D(w)$ of θ and a such that P can be applied to v according to a substitution σ with:

1. $\sigma(z) \sqsubset s$ and
2. for all restrictive hypothesis H of P of the form $\forall y D' y_{\prec s'} \rightarrow K$ there is a restrictive hypothesis H_0 of θ of the form $\forall y D' y_{\prec s_0} \rightarrow K$ such that $\sigma(s') \sqsubseteq s_0$.

This characterization is due to the following proposition (see [12] for proof).

Proposition 3.9. There exists an I-proof for f iff there exists a distributing tree for f with the right terminal state.

Proposition 3.9 says that one can only focus on distributing trees that satisfy the right terminal state. So, as already mentioned, we do not explicit I-proofs here but we only consider distributing trees and the right terminal state properties.

4 Synthesising ordinal measures

The earlier system built proof trees which have the right terminal state property defined in [13]. It has been shown in [14] that one can extract an ordinal measure, which will be called R-measure, from each proof tree. The R-measure has the decreasing property if the proof tree satisfies the right terminal state property. This measure can be also defined against a proof tree with the new context. But the decreasing property of the R-measure is not valid anymore. A reason is that, as the system *ProPre* corresponds to an extension of the `Recursive Definition` of the Coq system, the existence of suitable measures does not correspond any longer to the R-measures. It turns out that if we want to retrieve the decreasing property, we need to extend the class of measures to other measures.

In this section we recall the definition of the R-measures but in the context of the extended system, and we present the theorem on the decreasing property of the measures that fails but which will be re-established. We then introduce the extended measures for which Theorem 1 holds again.

4.1 The R-measures

Before giving the ordinal measures we first introduce some definitions concerning the judgments in distributing trees.

Definition 4.1. Let $\mathcal{A}$ be a distributing tree. A branch $\mathcal{B}$ from the root θ_1 to a leaf θ_k will be denoted by $(\theta_1, x_1), \ldots, (\theta_{k-1}, x_{k-1}), \theta_k$ where x_i $(1 \leq i < k)$, is the variable for which either the rule *Struct* or *Ind* is applied on θ_i.

Definition 4.2. 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.

According to the definition of a distributing tree $\mathcal{A}$, we have the two following straightforward facts.

Fact 4.3. Let $\mathcal{E}$ be a specification of a function f of type $s_1, \ldots, s_n \rightarrow s$ and $\mathcal{A}$ be a distributing tree. 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 $\rho(C(\theta)) = f(t_1, \ldots, t_n)$.

Fact 4.4. 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 $\sigma_{j,i}(C(\theta_i)) = C(\theta_j)$.

Definition 4.5. The *recursive length* of a term t of sort s is defined by:

1. if t is a constant c, then $lg(c) = 0$,
2. if $t = C(t_1, \ldots, t_n)$ with $C : s_1, \ldots, s_n \to s \in \mathcal{F}_c$ then $lg(t) = 1 + \sum_{s_j = s} lg(t_j)$.

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 distributing tree $\mathcal{A}$ for $\mathcal{E}$. The *R-measure* $\Omega_R : \mathcal{T}(\mathcal{F}_c)_{s_1} * \ldots * \mathcal{T}(\mathcal{F}_c)_{s_n} \to \omega^\omega$, where ω is the least infinite ordinal, 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 $\rho(C(\theta)) = f(t)$ (Fact 4.3). 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 4.4. Then $\Omega_R(t)$ is defined as the following ordinal sum:

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

We now need some definitions before giving Theorem 1.

Definition 4.7. A finite sequence of positive integers q will be called a position, ϵ will denote the empty sequence and $\cdot$ the concatenation operation on sequences.

For each position q and sort s, we will assume there is a new variable of sort s indexed by q distinct from those of $\mathcal{X}$. The following definition allows us to state Theorem 1 below.

Definition 4.8. Let be a term t and q be a position, the term $[\![t]\!]_q$ is defined as follows: $[\![c]\!]_q = c$ if c is a constant, $[\![x]\!]_q = x$ 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 $C \in \mathcal{F}_c$, and $[\![f(t_1, \ldots, t_n)]\!]_q = x_q$ if $f \in F_d$.

Theorem 1. *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 right terminal state property. The associated measure Ω_R then satisfies the decreasing property. That is to say, for each recursive call $(f(t_1, \ldots, t_n), f(u_1, \ldots, u_n))$ of $\mathcal{E}$ and for every ground constructor substitution φ we have: $\Omega_R(\varphi(t_1), \ldots, \varphi(t_n)) > \Omega_R(\varphi([\![u_1]\!]_1), \ldots, \varphi([\![u_n]\!]_n))$*

Unfortunately, though Theorem 1 holds in the context of R-proofs (see [14]), examples show that it fails in the current context. Consider, for instance, the simple example of the specification of the addition function $add : nat, nat \to nat$, defined with an unusual way illustrating our purpose.

$$add(s(x), s(y)) = add(s(s(x)), y)$$
$$add(0, y) = y$$
$$add(s(x), 0) = s(x)$$

There exists a tree which enjoys the right terminal state property that leads to the following measure: $\Omega_R(u, v) = \omega * lg(u) + lg(v)$. Obviously the decreasing property does not hold.

In the remaining of the section, we introduce new measures that enable the theorem to be restored.

4.2 The new ramified measures

As already mentioned, an ordering relation $\sqsubset$ on term is introduced in the extended system. In contrast to the previous system, this relation can be checked outside of the formal proofs and so can be easily modified independently of the logical framework of the system. The ordering relation is related to a measure on terms in the following way.

Definition 4.9. Assume a measure m on the terms ranging over natural numbers. Let $u, v \in \mathcal{T}(\mathcal{F}_c, \mathcal{X})_s$ for a given sort s. We say that $u \sqsubset v$ iff:

$$1)\ m(u) < m(v), \qquad 2)\ \mathcal{V}ar(u) \subseteq \mathcal{V}ar(v), \qquad 3)\ u \text{ is linear}$$

A special measure, the so called size measure lgi, is used in the system and is defined as follows:

Definition 4.10. The *size measure* of a term t of sort s is given by:

1. if t is a constant or a variable, then $lgi(t) = 1$,
2. if $t = C(t_1, \ldots, t_n)$ with $C : s_1, \ldots, s_n \to s \in \mathcal{F}_c$ then $lgi(t) = 1 + lgi(t_1) + \ldots + lgi(t_n)$

Note that Definition 4.13 uses only the value on constructor ground terms for the measure m, but this one is also defined on constructor terms because it is needed for the termination proofs of the *ProPre* system.
In order to be able to prove the decreasing property of the new ordinal measures defined below, we will only need to assume a property on the measure m.

Property 4.11. Let $u, v \in \mathcal{T}(F, \mathcal{X})_s$ such that $u \sqsubset v$. Then for all constructor substitutions σ, we have $m(\sigma(u)) < m(\sigma(v))$.

Note that the lemma obviously holds for lgi. For that, it is enough to remark that $lgi(t) - 1 \geq 0$ and $lgi(\sigma(t)) = lgi(t) + \#(x,t) * \sum_{x \in \mathcal{V}ar(t)} (lgi(\sigma(x)) - 1)$ for any term t, where $\mathcal{V}ar(t)$ denotes the set of variables which occur in t and $\#(x,t)$ is the number of the occurrences of the variable x in t.

It is now necessary to distinguish the sequents coming respectively from an application of the *Struct*-rule and the *Ind*-rule. Therefore we introduce the following:

Definition 4.12. Let θ be a judgment in a distributing tree $\mathcal{A}$ and θ' an immediate children of θ. We say that θ is *decreasing* and θ' is an Ind-judgment if one comes from the other using the *Ind* rule. The *test* function ξ is defined on each node as follows: $\xi(\theta)$ is 1 if θ is a decreasing judgment and 0 if not.

Definition 4.13. Let $\mathcal{E}$ be a specification of a function $f : s_1, \ldots, s_n \to s$ such that there exists a distributing tree $\mathcal{A}$ for $\mathcal{E}$. The new *ramified measure* $\Omega_I : \mathcal{T}(\mathcal{F}_c)_{s_1} * \ldots * \mathcal{T}(\mathcal{F}_c)_{s_n} \to \omega^\omega$, 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 $\rho(C(\theta)) = f(t)$ (Fact 4.3). 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 4.4. Then

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

The intuition would suggest to substitute only the measure m instead of the recursive lg in Definition 4.6. But once again, examples show that Theorem 1 fails in that case. It is now far from obvious that the new ordinal measures enjoy the decreasing property. However Theorem 1 now holds with the new measures. whose version is given below with Theorem 2

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 right terminal state property. The associated measure Ω_I then satisfies the decreasing property. That is to say, for each recursive call $(f(t_1, \ldots, t_n), f(u_1, \ldots, u_n))$ of $\mathcal{E}$ and for every ground constructor substitution φ we have: $\Omega_I(\varphi(t_1), \ldots, \varphi(t_n)) > \Omega_I(\varphi([\![u_1]\!]_1), \ldots, \varphi([\![u_n]\!]_n))$*

Proof: The proof is long but it can be derived from the main Proposition 5.25 below. The reader is referred to [8] for a detailed proof of Theorem 2. □

Now that we have Theorem 2, we can extract from an automated termination proof of the *flatten* function defined at Section 3 the following ordinal measure which has the decreasing property:

$$\Omega_I(le) = \omega \qquad \Omega_I(br(le, a)) = \omega * (1 + lgi(a))$$
$$\Omega_I(br(br(a, b), c) = \omega * (2 + lgi(a) + lgi(b) + lgi(c)) + 1 + lgi(a) + lgi(b).$$

5 The analysis of the I-formulas

This section is devoted to the analysis of the I-formulas. Due to the shape of the distributing trees and the I-formula that appear in the branches, we need to introduce some definitions and to establish several lemmas which will is used for the proof of Theorem 2 and Proposition 5.25 .

Definition 5.1. For a term t and a subterm u of t that has only one occurrence in t, $u \triangleright t$ will denote the position of u in t.

Definition 5.2. $\mathcal{RH}(F)$ denotes the set of the restrictive hypotheses of an I-formula F and for $P = \forall z(Dz_{\prec s} \to F')$ with F' an I-formula, we define $\mathcal{RH}(P) = \mathcal{RH}(F')$. For P_i and P_j in $\mathcal{RH}(F)$ we say that P_i *is before* P_j if F can be written $P_1, \ldots, P_k \to D(t)$ with $1 \leq j < i \leq k$. Moreover, for a restrictive hypothesis P of F, then $\#(P, F) = 1 + card\{P' \in \mathcal{R}(F), P' \ before \ P\}$.

One can easily see that, if θ' is an immediate antecedent of θ in a distributing tree, then each restrictive hypothesis of θ corresponds to a restrictive hypothesis in θ'. A new restrictive hypothesis is also in θ' if the rule is *Ind.* Formally we have the following definition.

Definition 5.3. Let θ be a judgment in a distributing tree and θ' an immediate antecedent of θ. We define an injective application $\mathcal{R}es_{\theta',\theta} : \mathcal{R}(\theta) \hookrightarrow \mathcal{R}(\theta')$ with $\mathcal{R}es_{\theta',\theta}(P)$ the restrictive hypothesis P' in $\mathcal{R}(\theta')$ such that $\#(P', \theta') = \#(P, \theta)$.

$\mathcal{R}es_{\theta',\theta}(P)$ can be seen as the residual of P in θ' and therefore the application can be generalized to any antecedent θ' of θ using composition of applications.

Definition 5.4. For an Ind-judgment θ' in a distributing tree, the restrictive hypothesis P in θ such that $\#(P, \theta') = card(\mathcal{R}(\theta'))$ is called the *new hypothesis*, denoted by $\mathcal{N}(\theta')$. In particular, it is such that all restrictive hypotheses in θ' are before P.

Remark 5.5. We can remark that if θ is a decreasing judgment with x the induction variable and θ' an immediate antecedent then $x \triangleright C(\theta) = z \triangleright C(\mathcal{N}(\theta))$ where the new hypothesis $\mathcal{N}(\theta)$ is of the form $\forall z(Dz_{\prec s} \to H)$. This will be used for the proof of Proposition 5.25.

If θ' is an immediate antecedent of a decreasing judgment θ, we know that θ' is of the form: $\forall x_1 D_1(x_1), \ldots, \forall x_k D_k(x_k), \mathcal{N}(\theta') \to \theta_{-D(x)}[w/x]$, with $\mathcal{N}(\theta') = z(Dz_{\prec w} \to \theta_{-D(x)}[z/x])$. So, for a Ind-judgment θ', we can easily define the application $\mathcal{D}_{\theta'} : \mathcal{R}(\mathcal{N}(\theta')) \hookrightarrow \mathcal{R}(\theta')$ where $\mathcal{D}_{\theta'}(Q)$ is the restrictive hypothesis Q' of $\theta_{-D(x)}[w/x]$ with $\#(Q', \theta_{-D(x)}[w/x]) = \#(Q, \theta_{-D(x)}[z/x])$. We can say that $\mathcal{D}$ is a *duplication* of restrictive hypotheses.

Lemma 5.6. Let $P = \forall z(Dz_{\prec s}, H_1, \ldots, H_k \to D(t))$ be a restrictive hypothesis θ of a judgment in a distributing tree then
1) the variables of s are free in P and have no other occurrences in P,
2) the variables in P distinct of those in s are bounded in P.
3) s is a subterm of $C(\theta)$ and $s \triangleright C(\theta) = z \triangleright C(P)$.

Proof: See [12]. □

Definition 5.7. Let G and F be two restrictive hypotheses. We define a congruence relation as follows: F and G are said similar, denoted by $F \approx G$ if they are respectively of the form $\forall z(D(z)_{\prec s} \to H)$ and $\forall z(D(z)_{\prec t} \to H)$.

Lemma 5.8. Given an Ind-judgment θ in a distributing tree and P a restrictive hypothesis of $\mathcal{N}(\theta)$. Then $\mathcal{D}_\theta(P) \approx P$.

Proof: According to the form of $\mathcal{N}(\theta)$ (see the definition of $\mathcal{D}_{\theta'}$), we know that P and $\mathcal{D}_\theta(P)$ are of the form $\forall y(D'(y)_{\prec s'} \to H')[z/x]$ and $\forall y(D'(y)_{\prec s'} \to H')[w/x]$. Lemma 5.6 says that x does not occur in H (and may not possibly occur in s'). Therefore $P = \forall y(D'(y)_{\prec s'[z/x]} \to H')$ and $\mathcal{D}_\theta(P) = \forall y(D'(y)_{\prec s'[w/x]} \to H')$, thus $P \approx \mathcal{D}_\theta(P)$. □

Lemma 5.9. Let P be a restrictive hypothesis of θ in a distributing tree, and θ' an antecedent of θ. Then $\mathcal{R}es_{\theta',\theta}(P) \approx P$.

Proof: By induction on the branch between θ and θ'. □

Corollary 5.10. If θ is a judgment in a distributing tree, θ' an immediate antecedent of θ, and P a restrictive hypothesis of θ, then $\mathcal{R}(\mathcal{R}es_{\theta',\theta}(P)) = \mathcal{R}(P)$.

Proof: By Lemma 5.9, we have $P = \forall z(D(z)_{\prec s_1} \rightarrow F)$ and $\mathcal{R}es_{\theta',\theta}(P) = \forall z(D(z)_{\prec s_2} \rightarrow F)$. Thus $\mathcal{R}(P) = F = \mathcal{R}(\mathcal{R}es_{\theta',\theta}(P))$. □

Lemma 5.11. For all judgments θ in a distributing tree, then there does not exist two restrictive hypotheses similar in θ.

Proof: See [8] □

Definition 5.12. Let θ be a judgment in a distributing tree and $\theta_1, \ldots, \theta_n = \theta$ the consecutive judgments from the root θ_1 to θ. Let P be a restrictive hypothesis of θ. We note $\mathcal{J}(P)$ the first integer j such that there is $Q \in \mathcal{R}(\theta_j)$ with $P = \mathcal{R}es_{\theta,\theta_j}(Q)$, which is correct since $\mathcal{R}_{\theta,\theta}(P) = P$.

Since every application $\mathcal{R}es_{\theta',\theta}$ is injective, $\mathcal{R}es^{-1}_{\theta',\theta}(P)$ will denote the antecedent of P with the assumption that P is in the image of the application.

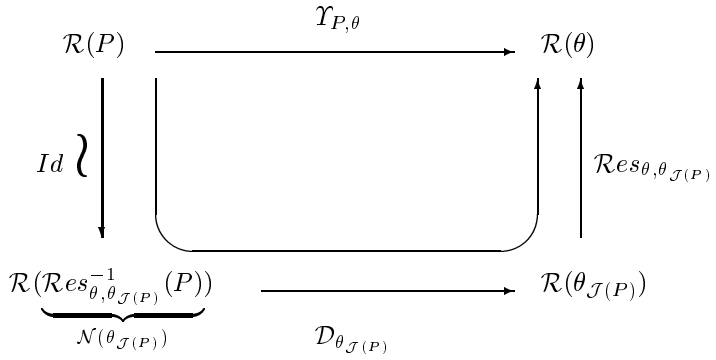
Lemma 5.13. In the context of the previous definition, the rule between $\theta_{\mathcal{J}(P)}$ and $\theta_{\mathcal{J}(P)-1}$ is the *Ind*-rule, and $\mathcal{R}es^{-1}_{\theta,\theta_{\mathcal{J}(P)}}(P) = \mathcal{N}(\theta_{\mathcal{J}(P)})$.

Proof: The opposite leads to a contradiction with the definition of $\mathcal{J}(P)$. □

Corollary 5.14. Let P be a restrictive hypothesis of a judgment θ in a distributing tree. Then, using also Corollary 5.10, we have

$$\mathcal{R}(P) = \mathcal{R}(\mathcal{R}es^{-1}_{\theta,\theta_{\mathcal{J}(P)}}(P)) = \mathcal{R}(\mathcal{N}(\theta_{\mathcal{J}(P)})).$$

Definition 5.15. Let θ be a judgment in a distributing tree and P be a restrictive hypothesis of θ. Then we can now etasblish the following diagram and thereby define the application $\Upsilon_{P,\theta} : \mathcal{R}(P) \hookrightarrow \mathcal{R}(\theta)$, with $\Upsilon_{P,\theta} = \mathcal{R}es_{\theta,\theta_{\mathcal{J}(P)}} \circ \mathcal{D}_{\theta_{\mathcal{J}(P)}}$.



In the case where θ is an Ind-judgment and $P = \mathcal{N}(\theta)$, then $\theta_{\mathcal{J}(P)} = \theta$ and $\Upsilon_{P,\theta} = \mathcal{D}_\theta$. So, Υ can be seen as a generalization of $\mathcal{D}$ for all restrictive hypotheses of any θ.

Fact 5.16. We remark that $\Upsilon_{P,\theta}$ is injective by composition of injective applications. Moreover, according to Lemmas 5.8 and 5.9, $\Upsilon_{P,\theta}(Q) \approx Q$ for all $Q \in \mathcal{R}(P)$.

Lemma 5.17. For a restrictive hypothesis P of a judgment θ in a distributing tree and Q a restrictive hypothesis of P, we have $\mathcal{J}(P) > \mathcal{J}(\Upsilon_{P,\theta}(Q))$.

Proof: See [8] □

Lemma 5.18. Let $\mathcal{A}$ be a distributing tree for a specification of a function, having the right terminal state property. Let θ be a leaf of $\mathcal{A}$ and (t, v) be a recursive call of $C(\theta)$. In this context, if P is the restrictive hypothesis of θ holding Definition 3.8 of the r.t.s.p of $\mathcal{A}$ and H and H_0 holding the point 2) of Definition 3.8 with the same notations, then $\Upsilon_{P,\theta}(H) = H_0$ and $\mathcal{J}(P) > \mathcal{J}(H_0)$.

Proof: According to the point 2) of Definition 3.8, we have $H \approx H_0$. Furthermore, by Fact 5.16, $\Upsilon_{P,\theta}(H) \approx H$. Hence Lemma 5.11 gives us that $\Upsilon_{P,\theta}(Q) = H_0$ and then $\mathcal{J}(P) > \mathcal{J}(H_0)$ with Lemma 5.17. □

Definition 5.19. For any θ in a distributing tree and an antecedent θ' of θ, then $[\theta, \theta']_D$ (respectively $[\theta, \theta'[_I$) will denote the set of the decreasing judgments (respectively Ind-judgments) between θ and θ' (respectively without θ').

Fact 5.20. Let $\mathcal{E}$ be a specification of a function f and $\mathcal{A}$ be a distributing tree for $\mathcal{E}$. If θ_1 is the root of $\mathcal{A}$, that is to say the termination statement of f, and if θ is an Ind-judgment in $\mathcal{A}$, then $card(\mathcal{R}(\mathcal{N}(\theta))) = card([\theta_1, \theta[_R)$.

Proof: Since $card(\mathcal{R}(\theta)) = card(\mathcal{R}(\mathcal{N}(\theta))) + 1$, it is actually enough to show that $card(\mathcal{R}(\theta)) = card([\theta_1, \theta]_R)$ which is then straightforward by induction on the number of judgments $\theta_1, \ldots, \theta$. □

Fact 5.21. Let P and P' be two distinct restrictive hypotheses of a judgment θ, then $\mathcal{J}(P) \neq \mathcal{J}(P')$.

Proof: The opposite leads to a contradiction thanks to Lemma 5.13. □

Lemma 5.22. Let $\mathcal{A}$ be a distributing tree having the r.t.s.p. with the root θ_1. Let P be the restrictive hypothesis of a leaf θ_k in the definition of the r.s.t.p., then for all $\theta \in [\theta_1, \theta_{\mathcal{J}(P)}[_I$, there is one and only one $H \in \mathcal{R}(P)$ such that $\theta = \theta_{\mathcal{J}(\Upsilon_{P,\theta_k}(H))}$.

Proof: By Lemma 5.13, for all $H \in \mathcal{R}(P)$, $\theta_{\mathcal{J}(\Upsilon_{P,\theta_k}(H))}$ is an Ind-judgment. Furthermore Lemma 5.18 says that $\mathcal{J}(P) > \mathcal{J}(\Upsilon_{P,\theta_k}(H))$ and so $\theta_{\mathcal{J}(\Upsilon_{P,\theta_k}(H))} \in [\theta_1, \theta_{\mathcal{J}(P)}[_I$. Let $U = \bigcup_{H \in \mathcal{R}(P)} \{\theta_{\mathcal{J}(\Upsilon_{P,\theta_k}(H))}\}$ included in $[\theta_1, \theta_{\mathcal{J}(P)}[_I$. As Υ_{P,θ_k} is injective, then, using Fact 5.21, we get $card(U) = card(\mathcal{R}(P))$. Now

$$\begin{aligned} Card([\theta_1, \theta_{\mathcal{J}(P)}[_I) &= card(\mathcal{R}(\mathcal{N}(\theta_{\mathcal{J}(P)}))) && \text{(Fact 5.20)} \\ &= card(\mathcal{R}(\mathcal{R}es^{-1}_{\theta,k\,\theta_{\mathcal{J}(P)}}(P))) && \text{(Lemma 5.13)} \\ &= card(\mathcal{R}(P)) && \text{(Corollary 5.14)} \end{aligned}$$

Hence $U = [\theta_1, \theta_{\mathcal{J}(P)}[_I$. □

Lemma 5.23. Let θ and θ' be two judgments in a distributing tree of a specification then $C(\theta)$ and $C(\theta')$ match the same term iff θ and θ' are in the same branch.

Proof: Fact 4.4 gives one sense, the other one is made assuming the opposite and using the fact that if a judgment does not match a term, then its antecedent do not neither. □

Lemma 5.24. Let θ be a judgment in a distributing tree of a specification and θ' an antecedent of θ. If P is a restrictive hypothesis of θ' such that $\theta_{\mathcal{J}(P)-1} \in [\theta, \theta'[_R$ then $C(\theta)$ matches $C(P)$.

Proof: by the previous lemma $C(\theta)$ matches $C(\theta_{\mathcal{J}(P)-1})$. Furthermore, let Q' denotes $\mathcal{R}es_{\theta', \theta_{\mathcal{J}(P)}}(P)$, then $Q \approx P$ with Lemma 5.9 and so $C(Q) = C(P)$. Now, since Q is the new hypothesis of $\theta_{\mathcal{J}(P)}$, it is easy to see that $C(\theta_{\mathcal{J}(P)-1})$ matches $C(Q)$. Hence $C(\theta)$ matches $C(P)$. □

We now state the main Proposition below that enables Theorem 2 to hold.

Proposition 5.25. Let $\mathcal{A}$ be a distributing tree of a specification $\mathcal{E}$ with the right terminal state property and (t, u) be an inductive call of $\mathcal{E}$. Let also $\mathcal{B} = (\theta_1, x_1), \ldots, (\theta_{k-1}, x_{k-1}), \theta_k$ be a branch of $\mathcal{A}$ with $C(\theta_k) = t$. Let P be a restrictive hypothesis of θ_k and σ^u be the substitution such that $\sigma^u(C(P)) = u$ with respect the r.t.s.p.. Then for each decreasing judgment θ_i in $\mathcal{B}$ which is a strict descendent of $\theta_{\mathcal{J}(P)-1}$ (i.e. $i < \mathcal{J}(P)-1$), $C(\theta_i)$ (respectively $C(\theta_{\mathcal{J}(P)-1})$) matches u according to a substitution σ_i^u (respectively $\sigma^u_{\mathcal{J}(P)-1}$) and

$$m(\varphi \circ \sigma_i^u(x_i)) \leq m(\varphi \circ \sigma_{k,i}(x_i)),$$
$$m(\varphi \circ \sigma^u_{\mathcal{J}(P)-1}(x_{\mathcal{J}(P)-1})) < m(\varphi \circ \sigma_{k,\mathcal{J}(P)-1}(x_{\mathcal{J}(P)-1}))$$

for all ground constructor substitution φ (where $\sigma_{k,j}$ are given in Fact 4.4).

Proof: Let θ_i be a decreasing judgment with $i < \mathcal{J}(P) - 1$. By Fact 4.4, we know that $C(\theta_i)$ matches $C(\theta_{\mathcal{J}(P)-1})$ which matches also $C(P)$ according to Lemma 5.24 (with $\theta = \theta_{\mathcal{J}(P)-1}, \theta' = \theta_k$), and so $C(\theta_i)$ matches u.
Now, we are going to show the first inequality. Since θ_{i+1} is an Ind-judgment, by Lemma 5.22, there is a restrictive Q of P such that $\mathcal{J}_{\Upsilon_{P,\theta_k}(Q)} = i + 1$. Let $\mathcal{N}(\theta_{i+1}) = \forall z(Dz_{\prec s} \to G)$ be the new hypothesis of θ_{i+1} and let Q_0 be $\Upsilon_{P,\theta_k}(Q)$. We know that $Q \approx Q_0$, likewise $Q_0 \approx \mathcal{R}es^{-1}_{\theta_k, \theta_{\mathcal{J}(Q_0)}}(Q_0) = \mathcal{N}(\theta_{\mathcal{J}(Q_0)})$. Hence $Q \approx \mathcal{N}(\theta_{i+1})$ and we can write $Q = \forall z(Dz_{\prec s'} \to G)$ and $Q_0 = \forall z(Dz_{\prec s_0} \to G)$. Now

$$\begin{aligned} x_i \triangleright C(\theta_i) &= z \triangleright C(\mathcal{N}(\theta_{i+1})) && \text{(Remark 5.5)} \\ &= z \triangleright C(Q) && (Q \approx \mathcal{N}(\theta_{i+1})) \\ &= s' \triangleright C(P) && \text{(Lemma 5.6)}. \end{aligned}$$

Moreover $C(\theta_i)$ matches $C(P)$ which matches u. Then, with the previous equalities, we have $\sigma_i^u(x_i) = \sigma^u(s')$.
Furthermore:

$$\begin{array}{lll} z \triangleright C(Q) = z \triangleright C(\Upsilon_{P,\theta_k}(Q)) & (Q \approx \Upsilon_{P,\theta_k}(Q)) \\ = s_0 \triangleright C(\theta_k) & (\text{Lemma } 5.6) \\ = s_0 \triangleright t & (C(\theta_k) = t). \end{array}$$

With the inequalities we have $x_i \triangleright C(\theta_i) = s_0 \triangleright t$. Hence, since $C(\theta_i)$ matches $C(\theta_k) = t$, we get $\sigma_{k,i}(x_i) = s_0$.
Finally, point 2) of the right terminal state property says that $\sigma^u(s') \sqsubseteq s_0$, and so, by Property 4.11, $m(\varphi \circ \sigma^u(s')) \leq m(\varphi(s_0))$. That is to say $m(\varphi \circ \sigma_i^u(x_i)) \leq m(\varphi \circ \sigma_{k,i}(x_i))$

It remains to show the second inequality. We recall that $\theta_{\mathcal{J}(P)}$ is an Ind-judgment whose new hypothesis is $\mathcal{N}(\theta_{\mathcal{J}(P)}) = \mathcal{R}es^{-1}_{\theta_{\mathcal{J}(P)},\theta_k}(P) \approx P$. Then

$$\begin{array}{lll} x_{\mathcal{J}(P)-1} \triangleright C(\theta_{\mathcal{J}(P)-1}) = z \triangleright C(\mathcal{N}(\theta_{\mathcal{J}(P)})) & (\text{Remark } 5.5) \\ = z \triangleright C(P) & (C(\mathcal{N}(\theta_{\mathcal{J}(P)}) = C(P)) \\ = s \triangleright C(\theta_k) & (\text{Lemma } 5.6) \\ = s \triangleright t & (C(\theta_k) = t). \end{array}$$

Thus $\sigma_{k,\mathcal{J}(P)-1}(x_{\mathcal{J}(P)-1}) = s$.
Furthermore, we have seen for the first inequality that $C(\theta_{\mathcal{J}(P)-1})$ matches P which matches u, then by a previous equality, $\sigma^u_{\mathcal{J}(P)-1}(x_{\mathcal{J}(P)-1}) = \sigma^u(z)$. Now using point 1) of the right terminal state property, we have $\sigma^u(z) \sqsubset s$ which gives, by Property 4.11, $m(\varphi \circ \sigma^u(z)) < m(\varphi(s))$. Therefore $m(\varphi \circ \sigma^u_{\mathcal{J}(P)-1}(x_{\mathcal{J}(P)-1})) < m(\varphi \circ \sigma_{k,\mathcal{J}(P)-1}(x_{\mathcal{J}(P)-1}))$. □

6 Conclusion

While the measures found from the termination proofs of the recursive definition command of Coq were shown in [14] to be suitable for other systems such as the NQTHM of [2, 3], they cannot be defined in the extended termination system without losing the decreasing property. We have solved the problem by showing the existence of other decreasing measures in the extended termination system in question (the new *ProPre* of [12]). Moreover, the new measures we found in this paper, enlarge the class of suitable measures in the sense that each recursive algorithm proven to terminate in the previous system *ProPre* [11] is also proven to terminate in the extended *ProPre* system [12].
The orders characterised by the measures differ from the lexicographic combinations of one fixed ordering [18, 15, 19]. We can also mention the work of [7] which supports the use of term orderings coming from the rewriting systems area especially those methods of [5, 17] which aim at automatically synthesising suitable polynomial orderings for termination of functional programs.
There is now no more obstacle to provide the measures to other systems that require such measures. The investigations of formal proofs in this paper highlight new measures and advocate as in [14] a termination method based on ordinal measures.

References

[1] 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.

[2] R.S. Boyer and J S. Moore. *A computational logic*, Academic Press, New York, 1979.

[3] R.S. Boyer and J S. Moore. *A computational logic handbook*, Academic Press, 1988.

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

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

[6] J. Giesl. *Automated termination proofs with measure functions.* Proceedings of the 19th Annual German Conference on Artificial Intelligence, Bielefeld, volume 981 of LNAI, 1995.

[7] J. Giesl. *Termination analysis for functional programs using term orderings.* Proceedings of the Second International Static Analysis Symposium, Glasgow, volume 983 of LNCS, 1995.

[8] F.D. Kamareddine and F. Monin. *On Formalised Proofs of Termination of Recursive Function.* http://www.cee.hw.ac.uk/ fairouz/papers/research-reports/ppdp99full.ps.

[9] J.L. Krivine and M. Parigot. Programming with proofs. *J. Inf. Process Cybern.*, EIK 26(3):149-167, 1990.

[10] D.S. Lankford. On proving term rewriting systems are Noetherian. *Technical Report Memo MTP-3*, Louisiana Technology University, 1979.

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

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

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

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

[15] F. Nielson and H.R. Nielson. Operational semantics of termination types. *Nordic Journal of Computing*, 3(2):144-187, 1996.

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

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

[18] C. Walther. Argument-bounded algorithms as a basis for automated termination proofs. *Proceedings of 9th International Conference on Automated Deduction*, Argonne, Illinois, volume 310 of LNCS, 1988.

[19] C. Walther. On proving the termination of algorithms by machine. *Artificial Intelligence*, 71(1):101-157, 1994.