A Unifying Approach to Recursive and Co-recursive Definitions[⋆]

Pietro Di Gianantonio and Marino Miculan

Dipartimento di Matematica e Informatica, Università di Udine,
Via delle Scienze 206, 33100 Udine, ITALY.
`digianantonio,miculan@dimi.uniud.it`

Abstract. In type theory based logical frameworks, recursive and co-recursive definitions are subject to syntactic restrictions that ensure their termination and productivity. These restrictions however greately decrease the expressive power of the language. In this work we propose a general approach for systematically defining fixed points for a broad class of well given recursive definition. This approach unifies the ones based on well-founded order to the ones based on complete metrics and contractive functions, thus allowing for *mixed recursive/corecursive* definitions. The resulting theory, implemented in the Coq proof assistant, is quite simple and hence it can be used broadly with a small, sustainable overhead on the user.

Introduction

In this paper we propose a general approach for systematically defining fixed points for a broad class of recursive definition.

The problem of ensuring well-defineteness of inductive schemata and productivity of coinductive schemata is a long-standing and well-known issue. Essentially, there are two kinds of approaches: either the proof system/compiler checks automatically the definition by means of some syntactically decidable mechanism, or the user has to provide the definition with a proof of its well-defineteness.

The two approaches are complementary each other. The first approach is more "user-friendly", but strictly less expressive than the second; we just mention the *guarded-by-constructors* condition [6, 8], implemented in Coq for ensuring productivity of corecursive definitions.

On the other hand, in a more semantic approach a model is given for recursive definitions, and a recursive definition is meaningful if it has a well-defined meaning in such a model. This approach has been used e.g. in Isabelle/HOL [16], where the underlying semantic model is that of monotone functions on sets. An alternative model, commonly used in concurrency theory [7], is that of *contractive functions* over *complete ultra-metric spaces*. Here, a well-given recursive definition induces a contractive function and therefore, by Banach theorem, admits a unique fixed-point.

⋆ Work supported by italian COFIN 2001013518 CoMeta and UE WG 29001 Types.

Besides this, the metric semantic approach suffers other two problems.

First, it is unnecessarily complicated: ultrametric spaces have far more structure than that strictly needed to defined the fixed point. For example there is no need to define the distance between to point as a value in $\mathbb{R}$, because most of the properties of $\mathbb{R}$ are not used in the proof of existence of fixed-points. As a consequence it is now quite standard to define the distance as a value in $\mathbb{N}$. Still one can ask what are the basic properties that the values of distance need to satisfy, to prove a fixed-point theorem, and at the same time what are the essential properties of metric space.

Another problem is that quite often definitions are given by a mixture of inductive and co-inductive arguments. A simple yet motivating example of such a function is the following. Consider a function $z : \mathit{Stream} \to \mathit{Stream}$ over streams of characters, implementing a sort of run-length compression: every sequence of up-to nine consecutive '1's is replaced by a decimal digit representing its length (for simplicity, we use just digit and not decimal numbers so that sequences longer that 9 are represent by sequences of 9 and a final digit). This function can be easily defined in any lazy functional language by a pattern matching definition like the following:

$$\begin{aligned}
z('0' :: s) &\triangleq '0' :: z(s) \\
z(c :: '0' :: s) &\triangleq c :: '0' :: z(s) \\
z(c :: '1' :: s) &\triangleq \begin{cases} c :: z(s) & \text{if } c \geq '9' \\ z(\mathrm{succ}(c) :: s) & \text{otherwise} \end{cases}
\end{aligned}$$

One easily is convinced that this natural definition is well-given (i.e., productive): whenever a character is consumed in input without producing a character in output, the function z is called on a different argument, *closer* to the threshold '9', and thus *eventually* a character is produced. However, since it is not guarded, this definition is not accepted by any co-recursive definition schema; on the other hand, it is not clear how the metric approach alone can be used to prove the convergency of this kind of definitions. In fact, the argument for proving productivity of the schema above is both coinductive and inductive ("closer... eventually").

In this work we present a generalization of the ultra-metric that can be used to prove convergency of a large class of recursive-corecursive definitions. Its main feature is to encompass in a unique framework the well-order approach used in inductive definition, and the metric approach, used in the co-inductive case. Moreover, bearing in mind the usage of this approach in formal proof systems, we strive for simplicity: unnecessary aspects of the metric approach have been simplified. In fact we generalize the metric approach by taking only some key properties needed to prove the existence of the fixed-point.

As an application to this idea we will show how our definition can be used in the proof assistant Coq [10] in order to validated co-recursive definitions that are not accepted by any syntactic schema.

Synopsis In Section 1 we develop the theory of general fixpoint definitions, introducing the notions of *(complete) ordered families of equivalences* stemming from ultrametric spaces. In Section 2 we apply this theory to two paradigmatic examples (such as the run-length compression mentioned above). In Section 3 we discuss briefly the implementation in the Coq proof environment. Comparison with related work and some final remarks are in Sections 4 and 5 respectively.

1 A theory of general fixpoint definitions

In this section, we introduce some fundamental definitions and properties about *(complete) ordered families of equivalences*, which can be seen as a synthetic (generalization of a) dual presentation of ultrametric spaces. Three running examples (function spaces over well ordered sets, streams and ultra-metric spaces) should be clarifying. Finally we will prove a result which allows the construction of fixed points for a large class of schemata.

1.1 Ordered families of equivalences

Definition 1 (Well-founded relation). *Given a set A, an order relation $<$ on A is* well-founded *if there is no infinite chain of elements $a_1, a_2, a_3, \dots$ such that $\dots a_3 < a_2 < a_1$.*

It is well known that, given a set A with a well founded relation $<$, it is sound to prove predicates, and to define functions, on A, by induction (recursion, respectively), on the relation A. In other words, in order to prove that a predicate P holds for all the elements of A, it is sufficient to prove that it holds for a generic a using as hypothesis that P holds for all the predecessors of A. This is formalized in the following *accessibility induction principle*

$$\frac{\forall a.(\forall a'.a' < a \Rightarrow P(a')) \Rightarrow P(a)}{\forall a.P(a)} \qquad \text{(Acc\_ind)}$$

which is available in many proof environments; for Coq [10], it is provided by the package `Wf`.

Notation. Given a set A with a relation $<$ and an element $a \in A$ we denote with $\downarrow a$ the *set of predecessors* of a (w.r.t. $<$), i.e., $\downarrow a \triangleq \{a' \in A \mid a' < a\}$.

Definition 2 (o.f.e.). *An* ordered family of equivalences (o.f.e.) *is a tuple $\mathcal{O} = \langle A, <, X, \equiv \rangle$ where A (the* carrier*) and X (the* domain*) are sets, $<$ is a well-founded order on A and $\equiv$ is an A-indexed family of equivalence relations $\langle \equiv_a \rangle_{a \in A}$ on X.*

We denote with $\equiv$ also the equivalence relation $\bigcap_{a \in A} \equiv_a$, that is $x \equiv y$ if for all $a \in A$: $x \equiv_a y$.

Remark 1. We do not require $\equiv$ to be exactly the equality relation over X. This allows us to also deal with situations where different kind of equalities can be considered on the set X, for example the extensional, the intensional and the Leibniz equalities. Important examples, for these situations, are logical frameworks based on type theories [10, 17]. In these situations our approach can be used, given a contractive function f, to define a fixed point x for f. But, in general this x is not the unique fixed point w.r.t. the intensional equality, but only w.r.t. $\equiv$.

We do not require the equivalences to get progressively finer, i.e., that for all $a < a'$, $\equiv'_a \subseteq \equiv_a$. In fact this is not needed in the construction of the fixed points, moreover there are applications that can be dealt with only using this more liberal definition, see the following Example 1.a. *(Remark 1)*

Example 1. (a) *(Function spaces)* Let $A = \langle A, < \rangle$ be a set with a well-founded order, and B a set. The ordered family of equivalences *induced* by $<$ on the function space $X = A \to B$ is $\mathcal{O}_{A\to B} \triangleq \langle A, <, A \to B, \equiv^{A\to B} \rangle$, defined by

$$f \equiv_a^{A\to B} g \overset{\triangle}{\iff} f(a) = g(a)$$

(b) *(Streams)* Let A be a set; we denote by Stream_A the usual datatype of infinite streams of elements of A (In the following, we will drop the index $_A$ when clear from the context). For $a \in A$ and $s \in \mathit{Stream}$, we denote by $a :: s$ the stream obtained by prepending a to s. For $s \in \mathit{Stream}$ and $n \in \mathbb{N}$, let $s^{(n)}$ denote the n-th element of s, and $s^{<n}$ the *n-th approximant* of s, that is the (finite) list of the first n elements of s (thus, $s^{<n} = (s^{(0)}, \ldots, s^{(n-1)})$, and $s^{<0} = ()$).
The *natural o.f.e.* over *Stream* is $\mathcal{O}_{\mathit{Stream}} \triangleq \langle \mathbb{N}, <, \mathit{Stream}, \equiv \rangle$ where for all $n \in \mathbb{N}$: $s \equiv_n t \overset{\triangle}{\iff} s^{<n} = t^{<n}$.

(c) *(Ultra-metrics)* Let c be a real number in the interval $(0, 1)$, let S be an ultra-metric space with a distance $d : (S \times S) \to \mathbb{R}$. We recall that an ultra-metric space S is a metric space whose distance d satisfies a stronger version of the triangular inequality, namely for all $r, s, t \in S$: $d(r, s) \le \max\{d(r, t), d(t, s)\}$. A pair S, c induces an ordered uniformity $\mathcal{O}_{S,c} = \langle \mathbb{N}, <, S, \equiv \rangle$, where $r \equiv_n s$ iff $d(r, s) \le c^n$ and $<$ is the standard order on naturals. It is not difficult to check $\mathcal{O}_{S,c}$ is indeed an ordered family of equivalences. (This can be seen as an inverse of [13, Ex. 4.2.27(13)]). Note that the order $<$ is such that larger elements in $\mathbb{N}$ induce finer equivalence relations on S. *(Example 1)*

1.2 Complete ordered families of equivalences

Definition 3. *Let $\mathcal{O} = \langle A, <, X, \equiv \rangle$ be an o.f.e.. Let I be a subset of A, and $(x_a)_{a\in I}$ a family of elements in X, indexed by I.*

– *We say that $(x_a)_{a\in I}$ is* coherent *if*

$$\forall a', a \in I\ .a' < a \Longrightarrow x_{a'} \equiv_{a'} x_a.$$

– *We say that* $(x_a)_{a\in I}$ *has as a* limit y *if*

$$\forall a' \in I \ . \ x_{a'} \equiv_{a'} y.$$

Example 2. (a) *(Function spaces)* In the o.f.e. $\mathcal{O}_{A\to B}$ of Example 1, a family of functions $(f_a)_{a\in I}$ is coherent iff for all $a' < a \in I$ we have $f_{a'}(a') = f_a(a')$.
(b) *(Streams)* A family of streams $(s_n)_{n\in\alpha}$ is coherent iff for all $n < m < \alpha$ we have $s_n^{<n} = s_m^{<n}$.
(c) *(Ultra-metrics)* In the o.f.e. generated by an ultra-metric space, a coherent family can be described as a subsequence of a Cauchy sequence with a fixed *convergency rate*, that is Cauchy sequences $(x_n)_{n\in\mathbb{N}}$ such that:

$$\forall n \in \mathbb{N}.\forall m > n : d(x_n, x_m) \leq c^n$$

(Example 2)

Definition 4 (c.o.f.e.). *A* complete ordered family of equivalences *(c.o.f.e.) is a tuple* $\mathcal{O} = \langle A, <, X, \equiv, \lim_{\cdot\in A}, \lim_{\cdot<\cdot}\rangle$ *such that*

– $\langle A, <, X, \equiv\rangle$ *is an o.f.e.;*
– $\lim_{\cdot\in A}$ *is a function such that for all coherent families* $(x_a)_{a\in A}$*,* $\lim_{a\in A} x_a$ *is a limit for* $(x_a)_{a\in A}$*;*
– $\lim_{\cdot<\cdot}$ *is a function such that for all* $a \in A$ *and for all coherent families* $(x_{a'})_{a'\in\downarrow a}$*:* $\lim_{a'<a} x_{a'}$ *is a limit for* $(x_{a'})_{a'\in\downarrow a}$*.*

Using a "dependent type" notation, the arity of the two limit constructors is

$$\lim_{\cdot\in A} : (A \to X) \to X \qquad \lim_{\cdot<\cdot} : \prod_{a\in A} (\downarrow a \to X) \to X$$

Usually, as in all examples presented in this paper, these limit constructors are defined by diagonalization, although in general this is not always the case.

Remark 2. Alternatively we could have given a stronger definition of completeness, that is to ask for a function that gives a limit to every coherent family of the form $(x_a)_{a\in I}$, where I is a *downward closed* subset of A (i.e. if $a \in I$ and $a' < a$ then $a' \in I$). However this alternative definition would be equivalent to ours. In fact $\downarrow a$ and A are particular cases of downward closed sets, and they suffices:

Proposition 1. *Let* $\mathcal{O} = \langle A, <, X, \equiv, \lim_{\cdot\in A}, \lim_{\cdot<\cdot}\rangle$ *be a c.o.f.e.. For all* $I \subseteq A$ *downward closed, every coherent family of the form* $(x_a)_{a\in I}$ *has a limit.*

Proof. Given a coherent family $(x_a)_{a\in I}$ on any downward closed set I, we define, by induction on $<$, a family of elements $(y_a)_{a\in A}$ as follows:

$$y_a \triangleq \begin{cases} x_a & \text{if } a \in I \\ \lim_{a'<a} y_{a'} & \text{if } a \notin I \end{cases}$$

It is straightforward to prove, by induction on $<$, that $(y_{a'})_{a'\downarrow a}$ is a coherent family. It follows that $(y_a)_{a\in A}$ is a coherent family. It is then immediate to check that $\lim_{a\in A} y_a$ is a limit also for $(x_a)_{a\in I}$. □

The advantage of Definition 4 is that we need to define limits only for two particularly simple kinds of downward closed subsets: the whole carrier A and the predecessors of elements of A. This is useful in practical reasoning in proof assistants like, e.g., Coq, where conditions like "$a' < a$" are rendered directly, while "$a \in I$" would require some (cumbersome) representation of sets.

(Remark 2)

Example 3. (a) *(Function spaces)* We extend $\mathcal{O}_{A \to B}$ to a c.o.f.e., by defining $\lim_{\cdot \in A}$ and $\lim_{\cdot < \cdot}$ as follows

$$\left(\lim_{a \in A} f_a\right)(a') \triangleq f_{a'}(a') \qquad \left(\lim_{c<a} f_c\right)(a') \triangleq \begin{cases} f_{a'}(a') & \text{if } a' < a \\ b & \text{otherwise} \end{cases}$$

where $b \in B$ is some fixed element in B. It is immediate to test that $\lim_{\cdot \in A}$ and $\lim_{\cdot < \cdot}$ satisfy all the necessary conditions.

(b) *(Streams)* We can extend $\mathcal{O}_{Stream}$ to a c.o.f.e., by defining $\lim_{\cdot \in \mathbb{N}}$ and $\lim_{\cdot < \cdot}$ as follows:

$$\lim_{m \in \mathbb{N}} s_m \triangleq (s_0^{(0)} :: s_1^{(1)} :: \cdots :: s_n^{(n)} :: \cdots), \qquad \text{i.e.} \quad \left(\lim_{m \in \mathbb{N}} s_m\right)^{(i)} \triangleq s_i^{(i)}$$

$$\lim_{m<n+1} s_m \triangleq s_n \qquad \lim_{m<0} s_m \triangleq s_0$$

where s_0 is an arbitray chosen stream. Again, it is immediate to check that $\lim_{\cdot \in \mathbb{N}}$ and $\lim_{\cdot < \cdot}$ give exactly a limit for coherent families, therefore they extend $\mathcal{O}_{Stream}$ to a c.o.f.e..

(c) *(Ultra-metrics)* It is easy to check that for every generic ultrametric space $\langle S, d\rangle$, and for every $c \in (0,1)$, the o.f.e. $\mathcal{O}_{S,c}$ is complete if and only if the metric space $\langle S, d\rangle$ is complete. In fact suppose that $\mathcal{O}_{S,c}$ is complete and let $(s_i)_{i \in \mathbb{N}}$ be a Cauchy sequence in $\langle S, d\rangle$, then there exists a subsequence of it, $(s_{f(i)})_{i \in \mathbb{N}}$, such that $\forall i, j \,.\, d(s_{f(i)}, s_{f(i+j)}) = c^{-i}$, i.e., there exists a subsequence with with a fixed rate of convergency. It follows that $(s_{f(i)})_{i \in \mathbb{N}}$ is a coherent family and that $\lim_{i \in \mathbb{N}} s_{f(i)}$ is a convergency point for $(s_i)_{i \in \mathbb{N}}$. This proves that if $\mathcal{O}_{S,c}$ is complete then also S is complete. The converse is immediate. *(Example 3)*

1.3 Contractive functions

Definition 5 (Contractivity). *Given an o.f.e.* $\langle A, <, X, \equiv\rangle$ *a function* $F : X \to X$ *is* contractive *if for every pair of elements* x, y *in* X *and for every element* a *in* A*, if* $\forall a' < a \,.\, x \equiv_{a'} y$*, then* $F(x) \equiv_a F(y)$*.*

Example 4. (a) *(Recursive definitions)* A *recursive definition*, on a well-founded relation $<_A$ on A, of a function $A \to B$ can be defined by giving a function F which maps an element $a \in A$ and a function $f : \downarrow a \to B$ to a value on B.[1]

[1] Using dependent types, this can be written as $F : \prod_{x:A} \left(\prod_{y:A} (y < x) \to B\right) \to B$.

In this case the function $h : A \to B$ *recursively defined by* F is the unique function satisfying the equation:

$$h(a) = F(a, h|_{\downarrow a})$$

In fact, F induces a functional $F^* : (A \to B) \to (A \to B)$, defined by

$$F^*(f)(a) \triangleq F(a, f|_{\downarrow a})$$

It is straightforward to prove that for any recursive definition F the function F^* is contractive w.r.t. the o.f.e. $\mathcal{O}_{A\to B}$ induced by $<_A$.

(b) *(Streams)* A quite standard metric over streams, which can be generalized to any syntactic structure, is $d : \mathit{Stream} \times \mathit{Stream} \to \mathbb{R}$ given as follows:

$$\text{for } s, t \in \mathit{Stream} : \quad d(s,t) \triangleq 2^{-\sup\{n | s^{<n} = t^{<n}\}}$$

It is immediate to check that $d(s,t) \leq 2^{-n}$ if and only if $s \equiv_n t$. A function $F : \mathit{Stream} \to \mathit{Stream}$ is said to be *contractive w.r.t.* d iff for all $s, t \in \mathit{Stream}$, we have

$$d(F(s), F(t)) \leq \frac{d(s,t)}{2} \ .$$

We can prove easily that F is contractive w.r.t. d, if and only if F is contractive w.r.t. the o.f.e. $\mathcal{O}_{\mathit{Stream}}$. In fact if F is contractive, w.r.t. the metric d, then for each $s, t \in \mathit{Stream}$ and $n \in \mathbb{N}$ such that for all $m < n : s \equiv_m t$, then $d(s,t) \leq 2^{-(n-1)}$, and, by contractivity of F, $d(F(s), F(t)) \leq 2^{-n}$, which implies that $F(s) \equiv_n F(t)$. The other implication can be proved similarly.
It is worthwhile to notice that syntactic conditions ensuring productivity of corecursive schemata by means of some *guardedness* of recursive calls (such as the *guarded induction principle* [6], and the *guarded-by-constructors* condition of $CC^{(Co)Ind}$ [8] implemented in Coq [10]), ultimately give rise to functions contractive w.r.t. d.

(c) *(Ultrametrics)* Given an ultrametric space $\langle S, d\rangle$ with $d : (S \times S) \to [0,1]$, if a function F is contractive, with constant c, in the metric d, then F is contractive in the o.f.e. $\mathcal{O}_{S,c}$.
The implication in the other direction does not hold. A contractive map F in $\mathcal{O}_{S,c}$ maps x and y with distance $d(x,y) \in [c^{n+1}, c^n)$ to $F(x)$ and $F(y)$, respectively, with distance $d(F(x), F(y)) \in [c^n, c^{n-1})$; however, the difference between the two distances may be arbitrarily small: they could be both arbitrary close to c^n. So we cannot find a constant contraction factor. Thus, in defining the o.f.e. $\mathcal{O}_{S,c}$ we lost some information on the metric d.
(Example 4)

1.4 Construction of generalized fixed-points

We come now to the main result of this paper:

Theorem 1 (Generalized fixed-point). *Let $\mathcal{O} = \langle A, <, X, \equiv, \lim_{\cdot \in A}, \lim_{\cdot <\cdot} \rangle$ be a c.o.f.e., and F a contractive function on X. Then, there exists a* generalized fixed point[2] $x \in X$*, that is an element x such that $x \equiv F(x)$.*

Moreover for every other generalized fixed-point y, we have that $x \equiv y$.

Proof. We define by induction on $<$ a family $(x_a)_{a \in A}$ as follows:

$$x_a \triangleq F\left(\lim_{b<a} x_b\right)$$

By induction on $<$ it is possible to prove that for each $a \in A$ the family $(x_b)_{b \in \downarrow a}$ is coherent, that is:

$$\forall c < b \in {\downarrow}a \ . \ x_c \equiv_c x_b \ .$$

In fact, by contractivity of F, this proposition follows from

$$\forall d < c < b \in {\downarrow}a \ . \ \lim_{e<c} x_e \equiv_d \lim_{e<b} x_e \ .$$

Indeed, by inductive hypothesis these limits are well defined; moreover, $x_d \equiv_d \lim_{e<c} x_e$ and $x_d \equiv_d \lim_{e<b} x_e$ by construction.

By a similar schema it is possible to prove that the complete family $(x_b)_{b \in A}$ is coherent.

We finally define the generalized fixed point $x \in A$ as

$$x \triangleq \lim_{a \in A} x_a$$

Let us prove that $x \equiv F(x)$. For every $a \in A$ we have that by construction $x_a \equiv_a x$, while by construction and by contractivity of F, $x_a \equiv_a F(x)$, and from these two equivalence we have the thesis.

Given any other generalized fixed-point y, it is then straightforward to prove by induction on $<$ that $\forall a \ . \ x \equiv_a y$. □

Example 5 (Function spaces). By applying Theorem 1 the previous recursive definition, on a well-founded relation $<_A$ on A, of a function $A \to B$ can be defined by giving a function F that having in input an element $a \in A$ and a function $f_a : ({\downarrow}a) \to B$ returns a value on B

The function F induces a function: $F^* : (A \to B) \to (A \to B)$, defined by:

$$F^*(f)(a) = F(a, f|_{\downarrow a})$$

It is straightforward to prove that for any recursive definition F the function F^* is contractive w.r.t. the o.f.e. on $(A \to B)$ induced by $<_A$. Then, there is a fixed point $h \equiv F^*(h)$, from which it follows that $h(a) = F(a, h|_{\downarrow a})$. *(Example 5)*

[2] These fixed points are called "generalized" because the equivalence $\equiv$ need not to be the equality relation "=", i.e., the diagonal.

2 Extended Examples

In this section we apply the theory developed in Section 1 to two definitions which are not dealt easily within Coq. The first example is the definition of the run-length compression function mentioned in the Introduction; the second is the definition of the stream of prime numbers. From these examples, finally, we will draw some general guidelines for defining c.o.f.e.'s.

2.1 Run-length compression

Let us denote by $\mathit{Char} = \{0 \ldots 255\}$ the finite set of character codes; in what follows, we denote by *Stream* the set of streams of characters. Consider the function $z : \mathit{Stream} \to \mathit{Stream}$, performing a *run-length compression* of up-to nine consecutive '1', declared by the following ML-style pattern matching schema (for sake of simplicity, let us suppose that the input stream of the function contains only '0's and '1's):

$$\begin{aligned} z('0' :: s) &\triangleq '0' :: z(s) \\ z(c :: '0' :: s) &\triangleq c :: '0' :: z(s) \\ z(c :: '1' :: s) &\triangleq \begin{cases} c :: z(s) & \text{if } c \geq '9' \\ z(\mathrm{succ}(c) :: s) & \text{otherwise} \end{cases} \end{aligned}$$

This definition is not accepted in Coq or similar systems where productivity is ensured by some syntactical condition. In fact, the definition is not guarded. In spite of this, we will prove that z is a generalized fixed point of the obvious function over the o.f.e. $\mathcal{O} = \langle \mathbb{N} \times \mathit{Char}, <, \mathit{Stream} \to \mathit{Stream}, \equiv \rangle$ defined as follows:

- $\langle \mathbb{N} \times \mathit{Char}, < \rangle$ is the lexicographic order, reversed on the second component (i.e., $(n_1, c_1) < (n_2, c_2)$ if $n_1 < n_2$, and $(n, c_1) < (n, c_2)$ if $c_2 < c_1$);
- $\equiv$ is defined as follows:

$$\begin{aligned} f \equiv_{n,c} g \iff & \forall s \in \mathit{Stream} \,.\, f(s) \equiv_n g(s) \\ & \qquad \wedge \forall s \in \mathit{Stream} \,.\, \forall a > c \,.\, f(a :: s) \equiv_{n+1} g(a :: s) \end{aligned}$$

It is easy to see that $\mathcal{O}$ can be extended to a c.o.f.e.. In fact it is sufficient to check that any coherent family in the form $(f_{n',c'})_{n',c' \in \mathbb{N} \times \mathit{Char}}$ has a limit f. Using the limit construction on coherent families of streams, presented in Example 3(b) we can define $f \triangleq \lambda s. \lim_{n \in \mathbb{N}} f_{n+1,'0'}(s)$ (notice that the definition works also if we substitute $'0'$ with any other character.) It is immediate to prove that for any pair n, c we have: $f_{n,c} \equiv_{n,c} f_{n+1,'0'} \equiv_{n,c} f$. Therefore f is a limit.

Then consider the function $F : (\mathit{Stream} \to \mathit{Stream}) \to (\mathit{Stream} \to \mathit{Stream})$ defined by the schema above, i.e.,

$$\begin{aligned} F(f)('0' :: s) &\triangleq '0' :: f(s) \\ F(f)(c :: '0' :: s) &\triangleq c ::' 0' :: f(s) \\ F(f)(c :: '1' :: s) &\triangleq \begin{cases} c :: f(s) & \text{if } c \geq '9' \\ f(\mathrm{succ}(c) :: s) & \text{otherwise} \end{cases} \end{aligned}$$

In order to check that F is contractive it is sufficient to prove that if $f \equiv_{n,\mathrm{chr}(0)} g$ then $F(f) \equiv_{n+1,\mathrm{chr}(255)} F(g)$, and that $f \equiv_{n,succ(c)} g$ then $F(f) \equiv_{n,c} F(g)$, and this fact can be proved by cases.

For example to prove that: $F(f) \equiv_{n+1,\mathrm{chr}(255)} F(g)$ it is sufficient to prove that for all streams s, $F(f)(s) \equiv_{n+1} F(g)(s)$, and this can be proved by cases on the shape of s. The most difficult case is when $s = c :: '1' :: s'$ and $c < '9'$, by definition of F, our goal became $f(\mathrm{succ}(c) :: s) \equiv_{n+1} g(\mathrm{succ}(c) :: s)$, and since $\mathrm{chr}(0) < \mathrm{succ}(c)$ this is implied by the hypothesis $f \equiv_{n,\mathrm{chr}(0)} g$.

2.2 Prime numbers

Consider the domain $X \triangleq \mathbb{N}\times\mathbb{N} \to \mathit{Stream}$, and the function $p : \mathbb{N}\times\mathbb{N} \to \mathit{Stream}$ defined by the following recursive equation:

$$p(m,n) = \begin{cases} n :: p(m*n, n+1) & \text{if } MCD(m,n) = 1 \\ p(m, n+1) & \text{otherwise} \end{cases} \tag{1}$$

Thus, $p(m,n)$ is the stream of numbers $\geq n$, prime with respect each other and to m. In particular, $\mathit{Primes} \triangleq p(1,2)$ is the stream of all prime numbers. A definition like (1) is not guarded, thus it cannot be accepted by any syntactic check based on guardedness of schemata. (In fact, the usual definition of the sieve of Eratosthenes in Coq replaces in the stream of naturals, non-prime numbers with 0's [11]). Actually, the proof of the productivity of (1) relies on the proof that there are infinitely prime numbers.

In order to prove that the above definition is well defined, we introduce the contractive functional F defined by the schema above and prove that it is contractive w.r.t. the o.f.e. $\mathcal{O} = \langle \mathbb{N}\times\mathbb{N}, <, \mathbb{N}\times\mathbb{N} \to \mathit{Stream}, \equiv\rangle$ defined as follows:

– $\langle \mathbb{N}\times\mathbb{N}, <\rangle$ is the lexicographic order
– $\equiv$ is defined as follows:

$$\begin{aligned} f \equiv_{i,j} g \iff & \forall m,n \in \mathbb{N}.\ f(m,n) \equiv_i g(m,n) \\ & \wedge \forall m,n \in \mathbb{N}.(-(n+1) \mod m) \leq j) \Rightarrow\ f(m,n) \equiv_{i+1} g(m,n) \end{aligned}$$

So we need to prove that if $f \equiv_{i,j} g$ then $F(f) \equiv_{i,j+1} F(g)$, and this implication holds since a recursive call either generates a new digit of the stream, or increases by one the second argument of the function. Moreover we need to prove that if for every $j \in \mathbb{N}$ $f \equiv_{i,j} g$ then $F(f) \equiv_{i+1,0} F(g)$. The main point in proving the above equivalence is to prove that for all m, n such that $-(n+1) \mod m = 0$ we have that $F(f)(m,n) \equiv_{i+2} F(g)(m,n)$; but in this case we have then $n = (c \times m) + 1$, which implies that n and m must be prime with respect to each other. For these arguments the recursive call of F has to generate an argument and from this it is straightforward to prove that the desired equivalence holds.

2.3 c.o.f.e.'s from mixed recursive/corecursive definitions

The examples presented above are two paradigmatic cases of *mixed recursive/corecursive* definitions of coinductive datatypes. In fact, from a careful analysis of these definitions, one can infer the right order and equivalences for the corresponding c.o.f.e..

These mixed recursive/corecursive definitions are given by cases; in some cases we have guarded corecursive calls, in the others unguarded calls but whose arguments have "decreased," with respect to some order. If the order of these arguments is well-founded, these unguarded calls cannot be nested endlessly; therefore, eventually a case of guarded call has to be applied, and thus an element has to be produced.

Bearing these considerations in mind, we can say that in general

- the carrier is given by the product of two well founded orders $\langle A_1, < \rangle$, $\langle A_2, < \rangle$. The first one is associated to the finite approximants of the elements in X while $\langle A_2, < \rangle$ is a well-founded order on the datatypes of the arguments which change in the non-guarded cases of the schema. The idea is that in the tuple (a_1, a_2), a_1 is a measure of how much the schema has produced so far (that is, how many productive calls have been applied), while the second component, a_2, is a measure of how many non-productive calls can be still applied;
- the order is a lexicographic order; the components of the arguments are ordered decreasing accordingly to the non-productive calls of the schema;
- given $x, y \in X$, $x \equiv_{a_1,a_2} y$ is defined by requiring x and y to be equivalent at least up to the first a_1 elements (they have been obtained by at least a_1 productive calls of the schema), and if they are given an argument less than a_2, then the results must be equivalent up to a'_1, for some $a'_1 > a_1$.

3 Implementation in Coq

An important point of this work is that all the theory developed in Section 1 is constructive. In particular, the proof of Theorem 1 gives us an effective way for building the fixed point for computable contractive functions over c.o.f.e.'s whose limit functions are computable.

In fact, the theory presented in this paper has been thoroughly formalized in Coq. This package[3] can be effectively used for the definition of fixed points for contractive functions which are rejected by the syntactic restriction of the logical framework. The user has to provide all the data of a c.o.f.e., that is a order, the indexed equivalences and the two limit constructors:

```
Variable A    : Set.
Variable less : (relation A).
Hypothesis less_wf : (well_founded A less).
Hypothesis less_trans : (transitive A less).
```

[3] the full code is available at `http://www.dimi.uniud.it/~miculan/CoqCode/COFE`

```
Variable  X    : Set.
Variable  eqn  : A -> (relation X).
Hypothesis eqn_equiv : (a:A)(equiv X (eqn a)).
Definition Eqn := [x,y:X](a:A)(eqn a x y).

Variable lim_t : (A -> X) -> X.
Variable lim_d : (a : A)((a':A)(less a' a) -> X) -> X.

Hypothesis completeness_t :
      (fx:A->X)(coherence_t fx) -> (a:A)(eqn a (fx a) (lim_t fx)).
Hypothesis completeness_d :
      (a:A)(fxa:(a':A)(less a' a) -> X)(coherence_d a f xa) ->
         (a':A)(l : (less a' a))(eqn a' (fxa a' l) (lim_d a fxa)).
```

Then, for any contractive function, the package provides the fixed point, unique up to the equivalences.

```
Variable F : X -> X.
Hypothesis F_contr : (contractive F).

Definition F_aux := [a:A][pfx : (a':A)(less a' a)-> X](F (lim_d a pfx)).
Definition fx :=
   [a:A](Acc_rec ? ? ([_:A] X)([a1:A][_:...] (F_aux a1)) a (less_wf a)).
...
Definition Fix_F := (lim_t fx).
Theorem GFPT_e : (Eqn Fix_F (F Fix_F)).
Theorem GFPT_u : (fix:X)(Eqn fix (F fix)) -> (Eqn fix Fix_F).
```

Remark 3. Notice that in general, `Eqn := [x,y:X](a:A)(eqn a x y)` need not to be equivalent to `(eq X)`, that is the intensional equality of CIC. For instance, on coinductive datatypes the interesting `Eqn` usually is some kind of bisimulation. On streams, this becomes the extensional equality: two streams are equivalent (bisimilar) iff they produce the same sequence of elements. This is strictly weaker than internal equality (i.e., convertibility), because the latter has to be decidable, while the former is not.

Extensional equality is what we get also when we apply the theory to the definition of plain recursive function over inductive datatypes, e.g. of type `nat->nat`. If `f:nat->nat` is fixed point of a contractive functional F, we can prove that `(n:nat)(f n)=(F f n)`, but in general we cannot prove that `f=(F f)`. The same fact holds also for the other approaches to fixed point constructor in type theoretic frameworks, see e.g. [5] and [2, Equation (2)], where the reduction of $(Rec_{wf<}\ F)$ to its unfolding is given pointwise. However, if we compute fixpoints over inductive datatypes (e.g., *nat*), we get back the usual internal equality.

(Remark 3)

As an aside, it is interesting to notice that in the formal proof of Theorem 1, we had to get rid of the recursion schemata over $<$, which is `Acc_rec`. The reduction we need is the following:

```
Lemma BalaaBertot : (a : A)(fx a) = (F_aux a [a':A;_:?](fx a')).
```

which is provable using the systematic method described in [2].

4 Related work

There is a lot of research aiming to overcome the limitations of guardedness-based checks; the focus now seems to be on specific types systems. In [9] Giménez has adapted Mendler's *type variables* approach [14] to the Calculus of Constructions; although quite expressive, to our knowledge there is no proof of subject reduction and strong normalization for this solution. Some recent development along this line has been done in [1, 4].

On the other side, we mention Paulson's work [16], where inductive (resp. coinductive) objects in Isabelle/HOL can be defined as least (resp., greatest) fixed points of monotone set-trasforming operators, using an implementation of Knaster-Tarski theorem. Syntactic restrictions in introduction rules are avoided by reasoning about the set-trasforming semantics of the rules. Despite its generality, this theory has not a unifying solution for inductive and coinductive definitions; thus, it is not clear if (and how) definitions like those in Section 2 can be rendered.

The closest research to ours is Matthews' [12], where a theory of *converging equivalence relations* and contractive functions is developed in Isabelle/HOL. In fact, c.o.f.e.'s can be seen as a generalization of CERs, since three axioms of CERs ([12, Fig. 1,(4)–(6)]) are not required for c.o.f.e.'s. Another significant difference is that the development in [12] relies on Hilbert's choice operator, which is not available in most proof editors (apart, of course, Isabelle/HOL).

Finally, general recursion definitions in type theories, which have been of inspiration for this work, have been studied by Nordström [15], Balaa and Bertot [2], and Bove and Capretta [5].

5 Conclusions

In this work we have presented a general methodology for proving well-definiteness of a large class of recursive-corecursive definitions. Its main feature is to encompass in a unique framework the well-order approach used in inductive definition, and the metric approach, used in the co-inductive case. Moreover, we strived for simplicity: although this approach is rooted in the theory of contractive functions over ultrametric spaces, we generalize the metric approach by taking only some key properties needed to prove the existence of the fixed-point.

In order to check the expressivity of this approach, we intend apply this approach to significative case studies. Some interesting applications could be in the formalization of processes and protocols.

From a pragmatical point of view, an interesting possible development is to investigate how much the check of contractivity can be automatized.

Acknowledgments. We are grateful to the anonymous referees for their helpful hints and suggestions.

References

[1] A. Abel. Termination checking with types. Technical Report 0201, Institut für Informatik, Ludwig-Maximilians-Universität München, 2002.

[2] A. Balaa and Y. Bertot. Fix-point equations for well-founded recursion in type theory. In *Proc. TPHOL 2000*, volume 1869 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2000.

[3] H. Barendregt and T. Nipkow, editors. *Proceedings of TYPES'93*, volume 806 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[4] G. Barthe, M. J. Frade, E. Giménez, L. Pinto, and T. Uustalu. Type-based termination of recursive definitions. *Mathematical Structures in Computer Science*, 2003. To appear.

[5] A. Bove and V. Capretta. Modelling general recursion in type theory. Technical report, Dept. of Computer Science, Chalmers University of Technology, 2002.

[6] T. Coquand. Infinite objects in type theory. In Barendregt and Nipkow [3], pages 62–78.

[7] J. W. de Bakker and J. I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54:70–120, 1982.

[8] E. Giménez. Codifying guarded recursion definitions with recursive schemes. In J. Smith, editor, *Proc. of TYPES'94*, volume 996 of *Lecture Notes in Computer Science*, pages 39–59, Båstad, Sweden, June 1995. Springer-Verlag.

[9] E. Giménez. Structural recursive definitions in type theory. In K. Larsen, S. Skyum, and G. Winskel, editors, *Proc. 25th ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 397–408. Springer-Verlag, 1998.

[10] INRIA. *The Coq Proof Assistant*, 2002. `http://coq.inria.fr/doc/main.html` .

[11] F. Leclerc and C. Paulin-Mohring. Programming with streams in Coq. A case study: The sieve of Eratosthenes. In Barendregt and Nipkow [3], pages 191–212.

[12] J. Matthews. Recursive function definition over coinductive types. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, editors, *Proc. TPHOL'99*, volume 1690 of *Lecture Notes in Computer Science*, pages 73–90. Springer-Verlag, 1999.

[13] K. Meinke and J. Tucker. Universal algebra. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic for Computer Science*, volume I: Mathematical Structures, pages 189–411. Oxford University Press, 1992.

[14] N. P. Mendler. Recursive types and type constraints in second-order lambda calculus. In *Proc. LICS'87*, pages 22–25. IEEE, 1987.

[15] B. Nordström. Terminating general recursion. *BIT*, 28:605–619, 1988.

[16] L. C. Paulson. A fixedpoint approach to (co)inductive and (co)datatype definitions. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language, and Interaction: Essays in Honour of Robin Milner*, pages 187–211. MIT Press, 2000.

[17] R. Pollack. *The Theory of LEGO.* PhD thesis, University of Edinburgh, 1994. Available at `http://www.brics.dk/~pollack/export/thesis.dvi.gz`.