Structured general corecursion and coinductive graphs [extended abstract]

Tarmo Uustalu
Institute of Cybernetics at Tallinn University of Technology, Estonia
tarmo@cs.ioc.ee

Bove and Capretta's popular method for justifying function definitions by general recursive equations is based on the observation that any structured general recursion equation defines an inductive subset of the intended domain (the "domain of definedness") for which the equation has a unique solution. To accept the definition, it is hence enough to prove that this subset contains the whole intended domain.

This approach works very well for "terminating" definitions. But it fails to account for "productive" definitions, such as typical definitions of stream-valued functions. We argue that such definitions can be treated in a similar spirit, proceeding from a different unique solvability criterion. Any structured recursive equation defines a coinductive relation between the intended domain and intended codomain (the "coinductive graph"). This relation in turn determines a subset of the intended domain and a quotient of the intended codomain with the property that the equation is uniquely solved for the subset and quotient. The equation is therefore guaranteed to have a unique solution for the intended domain and intended codomain whenever the subset is the full set and the quotient is by equality.

Unique solutions to recursive equations General recursive definitions are commonplace in programming practice.

In particular, it is highly desirable to be able to define functions by some forms of controlled general recursion in type-theoretically motivated languages of total functional programming (in particular, proof assistants) that come with a set-theoretic rather than a domain-theoretic semantics. For an overview of this area, see Bove et al. [5].

In this paper, we are concerned with describing a function $f : A \to B$ definitely by an equation of the form:

$$\begin{array}{ccc} FA & \xleftarrow{\alpha} & A \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle f} \\ FB & \xrightarrow[\beta]{} & B \end{array} \tag{1}$$

where A, B are sets (the intended domain and codomain), F is a functor (the branching type of recursive call [corecursive return] trees), α is an F-coalgebra structure on A (marshals arguments for recursive calls) and β is an F-algebra structure on B (collects recursive call results). We are interested in conditions under which the equation is guaranteed to have a unique solution (rather than a least solution in a domain-theoretic setting or some solution that is canonical in some sense). There are several important generalizations of this setting, but we will not treat them here.

There are some well-known good cases.

D. Miller and Z. Ésik (Eds.): Fixed Points
in Computer Science 2012 (FICS 2012)
EPTCS 77, 2012, pp. 55–61, doi:10.4204/EPTCS.77.8

Some good cases (1): Initial algebra The following equation has a unique solution for any B, β.

$$\begin{array}{ccc} 1+\mathsf{El}\times\mathsf{List} & \xleftarrow{[\mathsf{nil},cons]^{-1}} & \mathsf{List} \\ {\scriptstyle 1+\mathsf{El}\times f}\downarrow & & \downarrow{\scriptstyle f} \\ 1+\mathsf{El}\times B & \xrightarrow[\beta]{} & B \end{array}$$

E.g., for $B = \mathsf{List}$ (lists over El), $\beta = \mathsf{ins}$ (insertion of an element into a list assumed to be sorted), we get $f = \mathsf{isort}$ (insertion sort).

A unique f exists because $(\mathsf{List}, [\mathsf{nil}, \mathsf{cons}])$ is the *initial algebra* for the functor $FX = 1 + \mathsf{El} \times X$. It is the *fold* (the unique algebra map) determined by the algebra (B, β).

Some good cases (2): Recursive coalgebras A unique solution exists for any B, β also for the equation

$$\begin{array}{ccc} 1+\mathsf{El}\times\mathsf{List}\times\mathsf{List} & \xleftarrow{\mathsf{qsplit}} & \mathsf{List} \\ {\scriptstyle 1+\mathsf{El}\times f\times f}\downarrow & & \downarrow{\scriptstyle f} \\ 1+\mathsf{El}\times B\times B & \xrightarrow[\beta]{} & B \end{array}$$

where $\mathsf{qsplit}\,\mathsf{nil} = \mathsf{inl}\,*$ and $\mathsf{qsplit}\,(\mathsf{cons}\,(x,xs)) = \mathsf{inr}\,(x, xs|_{\leq x}, xs|_{>x})$. E.g., for $B = \mathsf{List}$, $\beta = \mathsf{concat}$ (concatenation of the first list, the element and the second list), we get $f = \mathsf{qsort}$ (quicksort).

$(\mathsf{List}, \mathsf{qsplit})$ is <u>not</u> the inverse of the initial algebra of $FX = 1 + \mathsf{El} \times X \times X$ (which is the algebra of binary node-labelled trees), but we still have a unique f for any (B, β).

For this property, $(\mathsf{List}, \mathsf{qsplit})$ is called a *recursive coalgebra* of F. Recursive F-coalgebras form a full subcategory of the category of all F-coalgebras. The inverse of the initial F-algebra is the final recursive F-coalgebra.

While recursiveness is a very useful property of a coalgebra, it is generally difficult to determine whether a given coalgebra is recursive. For more information on recursive coalgebras, see Taylor [8], Capretta et al. [6], Adámek et al. [1].

Some good cases (3): Final coalgebra This equation has a unique solution for any A, α.

$$\begin{array}{ccc} \mathsf{El}\times A & \xleftarrow{\alpha} & A \\ {\scriptstyle 1+\mathsf{El}\times f}\downarrow & & \downarrow{\scriptstyle f} \\ \mathsf{El}\times\mathsf{Str} & \xrightarrow[\langle \mathsf{hd},\mathsf{tl}\rangle^{-1}]{} & \mathsf{Str} \end{array}$$

E.g., for $A = \mathsf{Str}$ (streams), $\alpha = \langle \mathsf{hd}, \mathsf{tl} \circ \mathsf{tl} \rangle$ (the analysis of a stream into its head and the tail of its tail), we get $f = \mathsf{dropeven}$ (the function dropping every even-position element of a given stream).

A unique f exists for any (A, α) because $(\mathsf{Str}, \langle \mathsf{hd}, \mathsf{tl} \rangle)$ is the *final coalgebra* of $FX = \mathsf{El} \times X$. It is the *unfold* (the unique F-coalgebra map) given by the coalgebra (A, α).

Some good cases (4): Corecursive algebras This equation has a unique solution for any A, α:

$$\begin{array}{ccc} \mathsf{El}\times A\times A & \xleftarrow{\ \alpha\ } & A \\ {\scriptstyle \mathsf{El}\times f\times f}\downarrow & & \downarrow{\scriptstyle f} \\ \mathsf{El}\times \mathsf{Str}\times \mathsf{Str} & \xrightarrow[\ \mathsf{smerge}\ ]{} & \mathsf{Str} \end{array}$$

Here $\mathsf{hd}\,(\mathsf{smerge}(x,xs_0,xs_1)) = x$ and $\mathsf{tl}\,(\mathsf{smerge}(x,xs_0,xs_1)) = \mathsf{smerge}(\mathsf{hd}\,xs_0,xs_1,\mathsf{tl}\,xs_0)$.

$(\mathsf{Str},\mathsf{smerge})$ is not the inverse of the final coalgebra of $FX = \mathsf{El}\times X\times X$, but a unique f still exists for any (A,α). We say that $(\mathsf{Str},\mathsf{smerge})$ is a *corecursive algebra* of F, cf. Capretta et al. [7]. [The inverse of the final F-coalgebra is the initial corecursive F-algebra and thus a special case.] Similarly to recursiveness of a coalgebra, corecursiveness of an algebra is a useful property, but generally difficult to establish.

The equation 1 can of course have a unique solution also in other cases. In particular, it may well happen that neither is (A,α) corecursive nor is (B,β) recursive, but the equation still has exactly one solution.

General case (1): Inductive domain predicate Bove and Capretta [3, 4] put forward the following approach to recursive definitions in type theory (the idea has occurred in different guises in multiple places; it must go back to McCarthy): for a given recursive definition, work out its "domain of definition" and see if it contains the intended domain.

For given (A,α), define a predicate dom on A *inductively* by

$$\frac{a : A \quad (\tilde{F}\,\mathsf{dom})\,(\alpha\,a)}{\mathsf{dom}\,a}$$

(i.e., as the smallest/strongest predicate validating this rule), denoting by $\tilde{F}\,P$ the lifting of a predicate P from A to FA.

Write $A|_{\mathsf{dom}}$ for the subset of A determined by the predicate dom, the "domain of definedness". It is easily verified that, for any (B,β), there is $f : A|_{\mathsf{dom}} \to B$ uniquely solving

$$\begin{array}{ccc} F(A|_{\mathsf{dom}}) & \xleftarrow{\ \alpha|_{\mathsf{dom}}\ } & A|_{\mathsf{dom}} \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle f} \\ FB & \xrightarrow[\ \beta\ ]{} & B \end{array}$$

If $\forall a : A.\,\mathsf{dom}\,a$, which is the same as $A|_{\mathsf{dom}} \cong A$, then f is a unique solution of the original equation 1, i.e., the coalgebra (A,α) is recursive.

For $A = \mathsf{List}$, $\alpha = \mathsf{qsplit}$, dom is defined inductively by

$$\frac{}{\mathsf{dom}\,\mathsf{nil}} \qquad \frac{x : \mathsf{El} \quad xs : \mathsf{List} \quad \mathsf{dom}\,(xs|_{\leq x}) \quad \mathsf{dom}\,(xs|_{>x})}{\mathsf{dom}\,(\mathsf{cons}\,(x,xs))}$$

We can prove that $\forall xs : \mathsf{List}.\,\mathsf{dom}\,xs$. Hence $(\mathsf{List},\mathsf{qsplit})$ is recursive.

If $A|_{\mathsf{dom}} \cong A$, the coalgebra (A,α) is said to be *wellfounded*. Wellfoundedness gives an induction principle on A: For any predicate P on A, we have

$$\frac{\begin{array}{cc} a:A & \begin{array}{c}a':A \quad (\tilde{F}\,P)\,(\alpha\,a')\\ \vdots \\ P\,a'\end{array}\end{array}}{P\,a}$$

We have seen that wellfoundedness suffices for recursiveness. In fact, it is also necessary. While this equivalence is easy for polynomial functors on the category of sets, it becomes remarkably involved in more general settings, see Taylor [8].

For $FX = 1 + \mathsf{El} \times X \times X$, $A = \mathsf{List}$, $\alpha = \mathsf{qsplit}$, we get this induction principle:

$$\frac{\begin{array}{cc} xs:\mathsf{List} \quad P\,\mathsf{nil} & \begin{array}{c}x:\mathsf{El} \quad xs':\mathsf{List} \quad P\,(xs'|_{\leq x}) \quad P\,(xs'|_{>x})\\ \vdots \\ P\,(\mathsf{cons}\,(x,xs'))\end{array}\end{array}}{P\,xs}$$

General case (2): Inductive graph relation The original Bove-Capretta method separates determining the domain of definition of a function from determining its values. Bove [2] showed that this separation can be avoided.

For given (A,α), (B,β), define a relation $\downarrow$ between A, B *inductively* by

$$\frac{a:A \quad bs:FB \quad \alpha\,a\,(\tilde{F}\downarrow)\,bs}{a\downarrow\beta\,bs}$$

Further, define a predicate Dom on A by

$$\mathsf{Dom}\,a = \exists b:B.\,a\downarrow b$$

It is straightforward to verify that $\forall a:A,b,b_*:B.a\downarrow b \wedge a\downarrow b_* \to b = b_*$. Moreover, it is also the case that $\forall a:A.\,\mathsf{Dom}\,a \leftrightarrow \mathsf{dom}\,a$. So, Dom does not really depend on the given (B,β)!

From the last equivalence it is immediate that there is $f:A|_{\mathsf{Dom}} \to B$ uniquely solving

$$\begin{array}{ccc} F(A|_{\mathsf{Dom}}) & \xleftarrow{\alpha|_{\mathsf{Dom}}} & A|_{\mathsf{Dom}} \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle f} \\ FB & \xrightarrow[\beta]{} & B \end{array}$$

And, if $\forall a:A.\,\mathsf{Dom}\,a$, which is the same as $A|_{\mathsf{Dom}} \cong A$, then f is a unique solution of the original equation.

As a matter of fact, recursiveness and wellfoundedness are equivalent exactly because $\forall a:A.\,\mathsf{Dom}\,a \leftrightarrow \mathsf{dom}\,a$.

For $FX = 1 + \mathsf{El} \times X \times X$, $A = \mathsf{List}$, $\alpha = \mathsf{qsplit}$, $B = \mathsf{List}$, $\beta = \mathsf{concat}$, the relation $\downarrow$ is defined inductively by

$$\frac{}{\mathsf{nil}\downarrow\mathsf{nil}} \qquad \frac{x:\mathsf{El} \quad xs:\mathsf{List} \quad xs|_{\leq x}\downarrow ys_0 \quad xs|_{>x}\downarrow ys_1}{\mathsf{cons}\,(x,xs)\downarrow\mathsf{app}\,(ys_0,\mathsf{cons}(x,ys_1))}$$

Inductive domain and graph do not work for non-terminating productive definitions Unfortunately, for our dropeven example,

$$\begin{array}{ccc} \mathsf{El}\times\mathsf{Str} & \xleftarrow{\langle \mathsf{hd},\mathsf{tl}\circ\mathsf{tl}\rangle} & \mathsf{Str} \\ {\scriptstyle 1+\mathsf{El}\times\mathsf{dropeven}}\downarrow & & \downarrow{\scriptstyle \mathsf{dropeven}} \\ \mathsf{El}\times\mathsf{Str} & \xrightarrow[\langle \mathsf{hd},\mathsf{tl}\rangle^{-1}]{} & \mathsf{Str} \end{array}$$

we get $\forall xs : \mathsf{Str}.\, \mathsf{dom}\, xs \equiv \bot$! Now, surely there is a unique function from $0 \to \mathsf{Str}$. But this is uninteresting! We would like to learn that there is a unique function $\mathsf{Str} \to \mathsf{Str}$.

Intuitively, the reason why this equation has a unique solution lies not in how a given argument is consumed but in how the corresponding function value is produced. This is not a terminating but a productive definition.

General case (3): Coinductive bisimilarity relation The concept of the domain of definedness can be dualized [7]. Besides partial solutions that are defined only on a subset of the intended domain, it makes sense to consider "fuzzy" solutions that are defined everywhere but return values in a quotient of the intended codomain. But since the category of sets is not self-dual, the theory dualizes only to a certain extent and various mismatches arise.

For given (B,β), define a relation $\approx$ on B *coinductively* by

$$\frac{b, b_* : B \quad b \approx b_*}{\exists bs, bs_* : FB.\, b = \beta\, bs \wedge b_* = \beta\, bs_* \wedge bs\, (\tilde{F} \approx^*)\, bs_*}$$

(i.e., we take $\approx$ to be the largest/coarsest relation validating this rule).

There need not necessarily be a function f solving the equation

$$\begin{array}{ccc} FA & \xleftarrow{\alpha} & A \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle f} \\ F(B/_{\approx^*}) & \xrightarrow[\beta/_{\approx^*}]{} & B/_{\approx^*} \end{array}$$

but, if such a function exists, it can easily checked to be unique. (See Capretta et al. [7, Thm. 1].)

If $\forall b, b_* : B.\, b \approx b_* \to b = b_*$, which is the same as $B/_{\approx^*} \cong B$ (where $B/_{\approx^*}$ is the quotient of B by the reflexive-transitive closure of $\approx$), we say that (B,β) is *antifounded*. If (B,β) is antifounded, solutions to equation 1 are the same as solutions to the equation above, and thus unique.

For $FX = \mathsf{El}\times X\times X$, $B = \mathsf{Str}$, $\beta = \mathsf{smerge}$, the relation $\approx$ is defined coinductively by

$$\frac{xs, xs_* : \mathsf{Str} \quad xs \approx xs_*}{\begin{array}{l}\exists x : \mathsf{El}, xs_0, xs_1, xs_{0*}, xs_{1*} : \mathsf{Str}. \\ \quad xs = \mathsf{smerge}(x, xs_0, xs_1) \wedge xs_* = \mathsf{smerge}(x, xs_{0*}, xs_{1*}) \wedge xs_0 \approx xs_{0*} \wedge xs_1 \approx xs_{1*}\end{array}}$$

It turns out that $\forall xs, xs' : \mathsf{Str}.\, xs \approx xs' \to xs = xs'$. Based on this knowledge, we may conclude that solutions are unique. (They do in fact exist as well for this example, but this has to be verified separately.)

Solutions need not exist for antifounded algebras. E.g., for $FX = X$, $B = \mathsf{Nat}$, $\beta = \mathsf{succ}$, we have that (B,β) is antifounded, but for A any set and $\alpha = \mathsf{id}_A$, the equation has the form $f\, a = \mathsf{succ}\, (f\ a)$ and has no solutions.

We have thus seen that antifoundedness of (B,β) does not guarantee that it is corecursive. The converse also fails: not every corecursive algebra (B,β) is antifounded [7, Prop. 5].

However, for an antifounded algebra (B,β), we do get an interesting coinduction principle on B: For any relation R on B, we have

$$\frac{\begin{array}{c} \\ b,b_* : B \quad bRb_* \quad \exists bs', bs'_* : FB.\, b' = \beta\, bs' \wedge b'_* = \beta\, bs'_* \wedge bs'\,(\tilde{F}\,R^*)\,bs'_* \end{array} \qquad \begin{array}{c} b',b'_* : B \quad b'Rb'_* \\ \vdots \end{array}}{b = b_*}$$

For $FX = \mathsf{El} \times X \times X$, $B = \mathsf{Str}$, $\beta = \mathsf{smerge}$, we get this coinduction principle:

$$\frac{\begin{array}{c} \\ xs, xs_* : \mathsf{Str} \quad xsRxs_* \quad \begin{array}{l} \exists x' : \mathsf{El}, xs'_0, xs'_1, xs'_{0*}, xs'_{1*} : \mathsf{Str}. \\ \quad xs' = \mathsf{smerge}(x', xs'_0, xs'_1) \wedge xs'_* = \mathsf{smerge}(x', xs'_{0*}, xs'_{1x*}) \wedge xs'_0 R xs'_{0*} \wedge xs'_1 R xs'_{1*} \end{array} \end{array} \qquad \begin{array}{c} xs', xs'_* : \mathsf{Str} \quad xs' R xs'_* \\ \vdots \end{array}}{xs = xs_*}$$

General case (4): Coinductive graph relation Could one also dualize the notion of the inductive graph? The answer is positive. Differently from the case of the coinductive concept of bisimilarity, this yields a criterion of unique solvability.

For given (A,α), (B,β), define a relation $\downarrow^\infty$ between A, B *coinductively* by

$$\frac{a : A \quad b : B \quad a \downarrow^\infty b}{\exists bs : FB.\, b = \beta\, bs \wedge \alpha\, a\, (\tilde{F} \downarrow^\infty)\, bs}$$

Define a predicate Dom^∞ on A by

$$\mathsf{Dom}^\infty a = \exists b : B.\, a \downarrow^\infty b$$

Now we can construct $f : A|_{\mathsf{Dom}^\infty} \to B/_{\approx^*}$ that we can prove to uniquely solve

$$\begin{array}{ccc} F(A|_{\mathsf{Dom}^\infty}) & \xleftarrow{\alpha|_{\mathsf{Dom}^\infty}} & A|_{\mathsf{Dom}^\infty} \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle f} \\ F(B/_{\approx^*}) & \xrightarrow[\beta/_{\approx^*}]{} & B/_{\approx^*} \end{array}$$

If both $\forall a : A.\, \mathsf{Dom}^\infty a$ and $\forall b, b_* : B.\, b \approx b_* \to b = b_*$, which are the same as $A|_{\mathsf{Dom}^\infty} \cong A$ resp. $B/_{\approx^*} \cong B$, then f uniquely solves also the equation 1. Notice, however, that in this situation we have obtained a unique solution only for the given (A,α): we have not established that (B,β) is corecursive.

To formulate a further condition, we define a relation $\equiv$ on B by

$$b \equiv b_* = \exists a : A.\, a \downarrow^\infty b \wedge a \downarrow^\infty b_*$$

A unique solution to equation 1 also exists if $\forall a : A.\, \mathsf{Dom}^\infty a$ and $\forall b, b_* : B.\, b \equiv b_* \to b = b_*$.

This condition is weaker: while $\forall b, b_* : B.\, b \equiv b_* \to b \approx b_*$, the converse is generally not true.

For $FX = \mathsf{El} \times X \times X$, $B = \mathsf{Str}$, $\beta = \mathsf{smerge}$ and any fixed A, α, the relation $\downarrow^\infty$ is defined coinductively by

$$\frac{a : A \quad xs : \mathsf{Str} \quad a \downarrow^\infty xs}{\exists xs_0, xs_1 : \mathsf{Str}.\, xs = \mathsf{smerge}\,(\mathsf{fst}\,(\alpha\, a), xs_0, xs_1) \wedge \mathsf{fst}\,(\mathsf{snd}\,(\alpha\, a)) \downarrow^\infty xs_0 \wedge \mathsf{snd}\,(\mathsf{snd}\,(\alpha\, a)) \downarrow^\infty xs_1}$$

It turns out that $\forall a : A.\, \mathsf{Dom}^\infty\, a$ no matter what A, α are. So in this case we do have a unique solution f for any A, α, i.e., $(\mathsf{Str}, \mathsf{smerge})$ is corecursive.

Conclusion We have considered two flavors of partiality of a function: a function may be defined only on a subset of the intended domain and the values it returns may be underdetermined.

The Bove-Capretta method in its graph-based version scales meaningfully to equations where unique solvability is not due to termination, but productivity or a combination the two. But instead of one condition to check by ad-hoc means, there are two in the general case.

The theory of corecursion/coinduction is not as clean as that of recursion/induction—in particular, to admit coinduction is not the same as to admit corecursion. We would like like to study the coinductive graph approach further and to find out to what extent it proves useful in actual programming practice. The main pragmatic issue is the same as with Bove and Capretta's method: how to prove the conditions.

Acknowledgments This research was supported by Estonian Science Foundation grant no. 6940 and the ERDF funded Estonian Centre of Excellence in Computer Science, EXCS.

References

[1] A. Adámek, D. Lücke & S. Milius (2007): *Recursive coalgebras of finitary functors*. *Theor. Inform. and Appl.* 41(4), 447–462. doi:10.1051/ita:2007028

[2] A. Bove (2009): *Another look at function definitions*. In S. Abramsky, M. Mislove & C. Palamidessi, editors: *Proc. of 25th Conf. on Mathematical Foundations of Programming Semantics, MFPS-XXV (Oxford, Apr. 2009)*, *Electron. Notes in Theor. Comput. Sci.* 249, Elsevier, 61–74. doi:10.1016/j.entcs.2009.07.084

[3] A. Bove & V. Capretta (2005): *Modelling general recursion in type theory*. *Math. Struct. in Comput. Sci.* 15(4), 671–708. doi:10.1017/s0960129505004822

[4] A. Bove & V. Capretta (2008): *A type of partial recursive functions*. In O. Aït Mohamed, C. Muñoz & S. Tahar, editors: *Proc. of 21st Int. Conf. on Theorem Proving in Higher Order Logics TPHOLs 2008 (Montreal, Aug. 2008*, *Lect. Notes in Comput. Sci.* 5170, Springer, 102–117. doi:10.1007/978-3-540-71067-7_12

[5] A. Bove, A. Krauss & M. Sozeau (2011): *Partiality and recursion in interactive theorem provers: an overview*. Manuscript, submitted to *Math. Struct. in Comput. Sci.*.

[6] V. Capretta, T. Uustalu & V. Vene (2006): *Recursive coalgebras from comonads*. *Inform. and Comput.* 204(4), 437–468. doi:10.1016/j.ic.2005.08.005

[7] V. Capretta, T. Uustalu & V. Vene (2009): *Corecursive algebras: a study of general structured corecursion*. In M. V. M. Oliveira & J. Woodcock, editors: *Revised Selected Papers from 12th Brazilian Symp. on Formal Methods, SBMF 2009 (Gramado, Aug. 2009)*, *Lect. Notes in Comput. Sci.* 5902, Springer, 84–100. doi:10.1007/978-3-642-10452-7_7

[8] P. Taylor (1999): *Practical Foundations of Mathematics*, chapter VI. Cambridge University Press.