Why partial coalgebras? (Abstract)

Frank Felfe and Horst Reichel

Institut Theoretical Computer Science
Dresden University of Technology
D–01062 Dresden, Germany

1 Motivation

Sometimes one can read, that there is no need for partial coalgebras since an endofunctor like

$$\mathcal{F}_0 : Set \to Set \quad \text{with} \quad \mathcal{F}_0(X) = X + \{*\},$$

also called partiality monad, can be used to represent partiality. But, for each computable $\mathcal{F}_0$–coalgebra $\mathbb{A}$, i.e., a coalgebra with a recursively enumerable set of states and a partial recursive state transition function $\alpha_{\mathbb{A}} : A \to A + \{*\}$, the subset

$$(1) \quad \{x \in A \mid \alpha_{\mathbb{A}}(x) = in_2(*)\} \subseteq A$$

is a decidable subset. This observation shows that total coalgebras systematically exclude those processes for which the subset (1), as defined above, is recursively enumerable and not decidable, in other words, *the most important case of partiality in recursion theory, caused by recursive definitions or by while–loops, is systematically excluded if one works with total coalgebras only.* Total $\mathcal{F}_0$–coalgebras represent processes where after each computation step it can be observed, whether the process stops or continues.

2 Basic Notions

Let $\mathcal{F} : Set \to Set$ by any endofunctor. A *partial* $\mathcal{F}$*–coalgebra*

$$\mathbb{A} = (A, \alpha : A \to ?\ \mathcal{F}(A))$$

is given by a set A and a partial mapping $\alpha : A \to ?\ \mathcal{F}(A)$. As usual $dom(\alpha) \subseteq A$ denotes the domain of the partial mapping. Since partiality of coalgebras affects the domain of the mapping $\alpha : A \to ?\mathcal{F}(A)$, the formal framework of partial coalgebras can not be developed as a dualization of the theory of partial algebras.

We will show that in contrast to partial algebras homomorphisms between partial coalgebras do not only preserve applicability of the partial state transition function, but have also to reflect applicability. Therefore a *strong homomorphism*

$$f : (A, \alpha) \to (B, \beta)$$

between $\mathcal{F}$–coalgebras is a mapping $f : A \to B$ such that $a \in dom(\alpha)$ if and only if $f(a) \in dom(\beta)$, and $f(\alpha(a)) = \beta(f(a))$ if $a \in dom(\alpha)$.

Since we are interested in properties of computable processes we will study partial $\mathcal{F} : Rec \to Rec$ coalgebras, where Rec denotes the subcategory of Set whose objects are recursively enumerable sets and the morphisms of Rec are the partial recursive mappings. This implies that Rec is not a full subcategory of Set.

We will present some very first results on the existence of fininal $\mathcal{F}$–coalgebras for the base category Rec.

If one takes into account the internal structure of type functors $\mathcal{F} : Rec \to Rec$ and distinguishes between pure changes of states and observations (corresponding to methods and attributes in object oriented programming), then surprisingly it turns out that partial observations are of greater importance than partial changes of states. We will demonstrate this by the example of deterministic automata with fixed input and output alphabets In, Ou, with a total state changing function $next : In \times States \to States$ and with a partial observation $out : States \to ?\, Out$.

3 Applications of partial coalgebras

Examples of specifications with partial coalgebras will be presented in an extension of CASL, called CO_CASL, which roughly allows besides initiality constraints **free** $\{\ldots\}$ also finality constraints **cofree** $\{\ldots\}$ such that both kinds of constraints can be nested.

If we want to specify the liveness property for streams which states that eventually an element x with $p(x) = true$ appears, then we can use a partial observation $witness : Str \to ?\, Nat$:

spec STREAMS_L = NAT_WITH_BOOL
then
sort $Elem$;
op $p : Elem \to Bool$;
then
cofree { **sort** Str
  ops $head : Str \to Elem$;
    $tail : Str \to Str$;
    $witness : Str \to ?\, Nat$;
  then
  free { **op** $iter : Nat \times Str \to Str$;
    axioms $\forall s : Str \bullet iter(zero, s) = s$;
      $\forall s : Str, n : Nat \bullet iter(suc(n), s) = tail(iter(n, s))$; **}**
  vars $s : Str$
  axiom $witness(s) \stackrel{e}{=} witness(s) \Rightarrow p(head(iter(witness(s), s))) \stackrel{e}{=} true$; **}**
end

The domain of the partial observation $witness : Str \to ?\, Nat$ defines exactly the intended class of streams.

If we want to define the fairness property which states that *always sometimes* p holds we have to add the axiom, that the domain of the partial observation is the carrier of a sub–coalgebra:

spec $\textsc{Streams\_f} = \textsc{Nat\_with\_Bool}$
then
sort $Elem$;
op $p : Elem \to Bool$;
then
cofree { **sort** Str
  ops $head : Str \to Elem$;
    $tail : Str \to Str$;
    $witness : Str \to ?\ Nat$;
  then
  free { **op** $iter : Nat \times Str \to Str$;
    axioms $\forall s : Str \bullet iter(zero, s) = s$;
      $\forall s : Str, n : Nat \bullet iter(suc(n), s) = tail(iter(n, s))$; }
  vars $s : Str$
  axioms $witness(s) \stackrel{e}{=} witness(s) \Rightarrow p(head(iter(witness(s), s))) \stackrel{e}{=} true$;
    $witness(s) \stackrel{e}{=} witness(s) \Rightarrow witness(tail(s)) \stackrel{e}{=} witness(tail(s))$ }
end