How to reason informally coinductively

Anton Setzer[1]

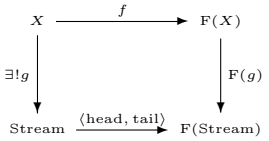
Dept. of Computer Science, Swansea University, Singleton Park, Swansea SA2 8PP
a.g.setzer@swan.ac.uk

In our article [1] we introduced the representation of final coalgebras, which correspond to non-well-founded data structures, as defined by their elimination rules rather than by their introduction rules.

When determined by their introduction rules, elements of final coalgebras are given by possibly non-well-founded many applications of the constructors. For instance a stream is given by an infinite application of the cons operation, cons n_1 (cons n_2 (cons n_3 $\cdots$)). Then increasing stream starting with n is given as inc n = cons n (inc $(n+1)$) which reduces to inc n = cons n (cons $(n+1)$ (cons $(n+2)$ $(\cdots)$)). The problem is that this results in non-normalisation and proper infinite terms.

When defined by their introduction rules, a coalgebra is given by the result of applying destructors (eliminators to it). For instance a stream is given by applying the operations head : Stream $\to \mathbb{N}$ and tail : Stream $\to$ Stream to it. As an example we have head (inc n) $= n$ and tail (inc n) = inc $(n+1)$. The problem of non-normalisation disappears under certain restrictions, for instance inc n is in normal form, unfolding its infinite nature requires repeated applications of tail to it.

Coalgebras are given as weakly final or as final coalgebras for a functor F. For instance the set of streams is given as a final coalgebra for the functor F : Set $\to$ Set, where Set is the category of sets, with object part $\mathrm{F}(X) = \mathbb{N} \times X$. This means that there exists a function Stream $\to$ F(Stream) (which is just $\langle \mathrm{head}, \mathrm{tail} \rangle$, and for any other coalgebra $f : X \to \mathrm{F}(X)$ there exists a unique $g : X \to$ Stream such that the following diagram commutes:



For weakly final coalgebras the condition on the uniqueness of g is omitted.

The principle of final coalgebras is equivalent to the principle of guarded recursion together with the fact that bisimilarity implies equality. Bisimilarity is in itself an example of an indexed coalgebra, in case of Stream we have Bisim : Stream $\times$ Stream $\to$ Set. Therefore iteration over Bisim allows to show equality over Stream and other final coalgebras. This principle amounts to a coinduction principle over these coalgebras.

In this talk we will discuss how to reason using this coalgebra principle informally, rather than referring to formal schemes or the existence of a bisimulation relation. This is similar to the way we reason about inductive data types informally rather than referring to them being defined as largest fixed points or to formal induction schemes. We will apply this to proving bisimilarity of elements of process algebras.

References

[1] Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer. Copatterns: Programming infinite structures by observations. In Roberto Giacobazzi and Radhia Cousot, editors, *Proceedings of the 40th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '13, pages 27–38, New York, NY, USA, 2013. ACM.