Indexed copatterns: reasoning about infinite structures by observations

David Thibodeau and Brigitte Pientka
McGill University

Reasoning about finite data such as lists or even lambda terms is well understood and programming languages and proof systems offer extensive support to inductively define such finite data. Beyond encoding such structures in a simply typed setting, indexing types allows us to enforce even more precise invariants about inductive objects; for example, we may index lists with their length to reason about their size or index lambda-terms with their types to guarantee that we are only working with well-typed objects and our interpreter is type preserving. The dual notion of induction is coinduction, which enables representing and reasoning about infinite data such as processes or streams. Unlike the latter, current representations of coinduction have been proven to be unsatisfactory, either because their lack of elegance makes them difficult to use and reason about or because they break desirable properties (for example subject reduction is broken in Coq [4][7]).

We have proposed in [2] the idea to think of infinite data as defined by *observations* and characterize the result of each observation using *copatterns*. We then demonstrated in [1] how to guarantee that programs in this language are productive. In this line of work, infinite structures are defined by observations, and finite data, defined by constructors, can be mixed. In this talk, we investigate how to extend the idea of defining infinite data by allowing observations to be indexed by a domain. This is dual to indexing constructors with a domain. Just as in the inductive case, this indexing gives us the ability to reason about infinite structures and fix specifics invariants, together with the possibility to define coinductive predicates or relations.

Our extension towards indexed types is restricted to Beluga's type system. Beluga [3][8][9] is a two-levels language. The first one is an implementation of the logical framework LF [5]. The other level is a functional programming language environment supporting dependent types and pattern matching on LF data. We have developed an extension to Beluga that permits defining indexed observations and writing corecursive/recursive programs about them. Since our observations are defined as records, indexing obeservations correspond to indexed records; as far as we know, indexed records have been absent in dependently typed programming. While dependent records allow latter fields to depend on earlier ones, they typically do not allow us to distinguish between fields based on an index. We demonstrate the elegance and power of indexed observations using three examples: 1) Indexing a stream of bits by a natural number indicating how many bits have been read so far. These indices allow us to certify programs reading sequences of bits of fixed length from the stream and modifying them. 2) The divergence of evaluation of lambda terms with evaluation inside lambda abstractions. In particular, we will prove that $(\lambda x.x\ x)\ (\lambda x.x\ x)$ diverges. 3) A type-preserving environment-based interpreter where we represent closures coinductively following ideas by Milner and Tofte [6].

Our prototype is the first programming environment that supports coinductive reasoning about LF specifications. This will allow us to explore new applications such as bisimulation proofs and proofs about contextual equivalence. We are currently extending and adapting our prior work on copatterns [2] to indexed copatterns in Beluga. More broadly, we see our work which restricts dependencies to a decidable domain as an interesting step towards supporting copatterns in a fully dependently typed languages.

References

[1] Andreas Abel and Brigitte Pientka. Well-founded recursion with copatterns:a unified approach to termination and productivity. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming (ICFP '13)*, 2013.

[2] Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer. Copatterns: programming infinite structures by observations. In *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '13)*, pages 27–38. ACM Press, 2013.

[3] Andrew Cave and Brigitte Pientka. Programming with binders and indexed data-types. In *39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'12)*, pages 413–424. ACM Press, 2012.

[4] Eduardo Giménez. *Un Calcul de Constructions Infinies et son application à la vérification de systèmes communicants*. PhD thesis, Ecole Normale Supérieure de Lyon, December 1996. Thèse d'université.

[5] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, January 1993.

[6] Robin Milner and Mads Tofte. Co-induction in relational semantics. *Theoretical Computer Science*, 87(1):209 – 220, 1991.

[7] Nicolas Oury. Coinductive types and type preservation. Message on the coq-club mailing list, June 2008.

[8] Brigitte Pientka. A type-theoretic foundation for programming with higher-order abstract syntax and first-class substitutions. In *35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08)*, pages 371–382. ACM Press, 2008.

[9] Brigitte Pientka and Joshua Dunfield. Beluga: a framework for programming and reasoning with deductive systems (System Description). In Jürgen Giesl and Reiner Haehnle, editors, *5th International Joint Conference on Automated Reasoning (IJCAR'10)*, Lecture Notes in Artificial Intelligence (LNAI 6173), pages 15–21. Springer-Verlag, 2010.