Conditional Parametricity in Isabelle/HOL

Extended Abstract

Jan Gilcher Andreas Lochbihler Dmitriy Traytel

Institute for Information Security, Department of Computer Science, ETH Zurich, Zurich, Switzerland

Parametricity [6] is a central notion in functional programming which singles out "truly" polymorphic functions. Such functions behave exactly the same, no matter what concrete type they are actually used with at run-time. For example, the standard map function on lists $\mathsf{map} :: (\alpha \to \beta) \to \alpha\ \mathit{list} \to \beta\ \mathit{list}$ is parametric in both type variables α and β as $\mathsf{map}\ f\ xs$ merely applies the function f to all elements of xs, independent of whether xs is a list of *int*s or a list of *string*s.

Parametricity has numerous theoretical and practical applications. On the theoretical side various theorems follow "for free" for parametric functions [7]. On the practical side, in particular in the proof assistant Isabelle/HOL [5], parametricity is the key driving force behind data refinement [4], transfer of definitions and theorems across subtypes and quotient types [3], productivity checks for corecursive definitions [1], and nonuniform (co)datatypes [2].

Parametricity can be formally captured by theorems. For map, it looks as follows:

$$\forall A :: \alpha \to \alpha' \to \mathbb{B}.\ \forall B :: \beta \to \beta' \to \mathbb{B}.\ ((A \Rightarrow B) \Rightarrow \mathsf{rel}_{list}\ A \Rightarrow \mathsf{rel}_{list} B)\ \mathsf{map}\ \mathsf{map}$$

Here, the relation $\mathsf{rel}_{list}\ A :: \alpha\ \mathit{list} \to \alpha'\ \mathit{list} \to \mathbb{B}$ holds between two lists iff they have the same length and their elements are pairwise related by A. Similarly, the *function space relator* $\Rightarrow$ lifts relations on the domain and codomain to relations on functions by $(A \Rightarrow B)\ f\ g$ iff $A\ x\ y$ implies $B\ (f\ x)\ (g\ y)$ for all x and y. Note how the above relation $(A \Rightarrow B) \Rightarrow \mathsf{rel}_{list}\ A \Rightarrow \mathsf{rel}_{list}\ B$ closely resembles the type of map.

Not all polymorphic functions in higher-order logic (HOL) are parametric though. For example, equality $(==) :: \alpha \to \alpha \to \mathbb{B}$ is not parametric in α: for the singleton type *unit*, equality always returns True, while for any non-singleton type it is not a constant function. As equality is not parametric, the statement $\forall A :: \alpha \to \alpha' \to \mathbb{B}.\ (A \Rightarrow A \Rightarrow \leftrightarrow)\ (==)\ (==)$ does *not* hold, but a weaker version does:

$$\forall A :: \alpha \to \alpha' \to \mathbb{B}.\ \mathsf{bi\_unique}\ A \longrightarrow (A \Rightarrow A \Rightarrow \leftrightarrow)\ (==)\ (==)$$

where the condition $\mathsf{bi\_unique}\ A$ requires A to be a single-valued and injective relation. So, we call equality $(==)$ *conditionally parametric* for bi-unique relations.

Overloading as supported by Isabelle's variant of HOL [8] is another source of non-parametric functions: clearly, algebraic operations like addition $(+) :: \alpha \to \alpha \to \alpha$ and $0 :: \alpha$ behave differently on the integers compared to a finite monoid.

The conditions on the relations propagate when new functions are defined using conditionally parametric ones. For example, the function $\mathsf{delete} :: \alpha \to \alpha\ \mathit{list} \to \alpha\ \mathit{list}$ removes the first occurrence of an element in a list and is defined as follows.

$$\mathsf{delete}\ \_\ [] = [] \qquad \mathsf{delete}\ x\ (y : ys) = \mathsf{if}\ x == y\ \mathsf{then}\ ys\ \mathsf{else}\ y : \mathsf{delete}\ x\ ys$$

As delete's definition uses $(==)$, it is also conditionally parametric only for bi-unique relations: $\forall A :: \alpha \to \alpha' \to \mathbb{B}.\ \mathsf{bi\_unique}\ A \longrightarrow (A \Mapsto \mathsf{rel}_{list}\ A \Mapsto \mathsf{rel}_{list}\ A)\ \mathsf{delete}\ \mathsf{delete}$. Similarly, list summation $\mathsf{sum} :: \alpha\ \mathit{list} \to \alpha$, which uses $(+)$ and 0, is conditionally parametric for relations that respect $(+)$ and 0, i.e., that are monoid homomorphisms.

Contribution. We have implemented an inference engine that analyses a function definition, computes the appropriate relation and a minimal set of conditions, and automatically proves the corresponding conditional parametricity theorem. Inference is necessary because HOL's type system does not capture when overloaded functions are used, unlike e.g. Haskell's type system. Hence, users no longer have to manually state and prove the parametricity theorems, but have the engine find and prove them. Our implementation is integrated with Isabelle's existing parametricity infrastructure and the generated theorems can directly be used by the subsequent tools. Our evaluation shows that our algorithm finds new theorems for hundreds of functions in Isabelle's library and in some cases, the inferred parametricity theorem is more general than what human Isabelle users had proven before.

At present, our engine expects the function definition to be given as a single non-recursive equation. Thus, it works only for non-recursive definitions and primitively (co)recursive functions, as the latter express the recursion using a parametric combinator. The other two important definition principles in Isabelle/HOL, (co)inductive predicates and well-founded recursion, need some preprocessing, on which we are currently working. (Co)inductive predicates bring elements from logic programming into the functional world and a mode analysis must first determine possible usage types.

Acknowledgements. Andreas Lochbihler was supported by Swiss National Science Foundation grant 153217.

References

[1] Blanchette, J. C., A. Bouzy, A. Lochbihler, A. Popescu and D. Traytel, *Friends with benefits: Implementing corecursion in foundational proof assistants*, in: *ESOP 2017*, LNCS **10201** (2017), pp. 111–140.

[2] Blanchette, J. C., F. Meier, A. Popescu and D. Traytel, *Foundational nonuniform (co)datatypes for higher-order logic*, in: *LICS 2017* (2017).

[3] Kunčar, O., "Types, Abstraction and Parametric Polymorphism in Higher-Order Logic," Ph.d. thesis, Technische Universität München (2016).

[4] Lammich, P., *Refinement based verification of imperative data structures*, in: *CPP 2016* (2016), pp. 27–36.

[5] Nipkow, T., L. C. Paulson and M. Wenzel, "Isabelle/HOL: A Proof Assistant for Higher-Order Logic," Springer, 2002.

[6] Reynolds, J. C., *Types, abstraction and parametric polymorphism*, in: *IFIP 1983*, Information Processing **83** (1983), pp. 513–523.

[7] Wadler, P., *Theorems for free!*, in: *FPCA 1989, London* (1989), pp. 347–359.

[8] Wenzel, M., *Type classes and overloading in higher-order logic*, in: *TPHOLs 1997*, LNCS **1275** (1997), pp. 307–322.