Proposal for a Bachelor’s thesis

Conditioned Parametricity in Isabelle/HOL

Supervisors: Dr. Andreas Lochbihler, Dr. Dmitriy Traytel
Professor: Prof. David Basin
Issue date: September 16, 2016

Prerequisites

- Good skills in functional programming (e.g., from the FMFP course)

Introduction

Parametricity is a central notion in functional programming which singles out “truly” polymorphic functions. Such functions, whose type contains type variables, 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

\[
\text{map} :: (a \to b) \to [a] \to [b]
\]

is parametric in both type variables a and b, as map \( f \) \( xs \) merely applies the function \( f \) to all elements of \( xs \), independent of whether we use map on a list of Ints or on a list of Strings.

Parametricity has numerous theoretical and practical applications. On the theoretical side, various theorems follow “for free” for parametric constants [10]. On the practical side, in particular in the proof assistant Isabelle/HOL [9], parametricity is the key driving force behind data refinement [5,7], transfer of definitions and theorems accross subtypes and quotient types [3,4], and ensuring productivity of corecursive definitions [1,2].

Not all polymorphic functions are parametric. Equality

\[
(=) :: a \to a \to \text{Bool}
\]

is not parametric in a: for a = (), equality always returns True, while for any non-singleton type it is not constant.

Parametricity can be formally captured in theorems. In the case of map, it looks as follows:

\[
\forall A :: a \to a' \to \text{Bool}. \forall B :: b \to b' \to \text{Bool}. \\
((A \Rightarrow B) \Rightarrow \text{rel_list} A \Rightarrow \text{rel_list} B) \Rightarrow \text{map} \Rightarrow \text{map}
\]
Here, the relation \( \text{rel_list} \ A :: [a] \to [a'] \to \text{Bool} \) 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. Note how the above relation \((A \Rightarrow B) \Rightarrow \text{rel_list} A \Rightarrow \text{rel_list} B\) closely resembles the type of \( \text{map} \).

Since equality is not parametric, the statement

\[
\forall A :: a \to a' \to \text{Bool}. \ (A \Rightarrow A \Rightarrow \leftrightarrow \ (==) (==))
\]

does not hold, but a weakened version does:

\[
\forall A :: a \to a' \to \text{Bool}. \ \text{bi_unique} A \to (A \Rightarrow A \Rightarrow \leftrightarrow \ (==) (==))
\]

Here the precondition requires \( A \) to be a bi-unique relation, or in other words the single-valued and injective. So, we call equality \((==)\) conditionally parametric w.r.t. bi-unique relations.

Often, such preconditions propagate as we define new functions. For example, the function \( \text{delete} \) removes the first occurrence of an element in a list and is defined as follows.

\[
\begin{align*}
\text{delete} :: \text{Eq} \ a \Rightarrow a \to [a] \to [a] \\
\text{delete} \_ \mathbb{[]} &= \mathbb{[]} \\
\text{delete} \ x \ (y : ys) &= \text{if} \ x == y \ \text{then} \ ys \ \text{else} \ y : \text{delete} \ x \ ys
\end{align*}
\]

Since it uses equality in its definition, \( \text{delete} \) is also only conditionally parametric w.r.t. bi-unique relations:

\[
\forall A :: a \to a' \to \text{Bool}. \ \text{bi_unique} A \to (A \Rightarrow \text{rel_list} A \Rightarrow \text{rel_list} A) \ \text{delete} \ \text{delete}
\]

**Objectives**

Conditional parametricity statements can proved for many functions in the proof assistant Isabelle/HOL. It currently maintains a database for such theorems, but they must be stated and proved manually for each function. The goal of this thesis is to provide automation for stating and proving conditional parametricity statements of functions which have been defined in terms of other (conditionally) parametric constants.

**Tasks**

This project can be subdivided into the following tasks:

1. Familiarize yourself with Isabelle/HOL and Isabelle/ML (read parts of [8][9][11]).
2. Familiarize yourself with existing automation for proving parametricity theorems in Isabelle/HOL [4, §4], [6, Theory Param_Tool.thy].
3. Implement a command that takes a defining equation of the form \( c = t \) as input and
   - constructs the appropriate relation for the type of \( c \);
   - collects the necessary preconditions from the conditional parametricity theorems for the functions in \( t \) according to the database; and
• automatically proves the conditional parametricity statement for \( c \).

4. Provide a variant of the command in which the user can specify the preconditions. If the inferred preconditions do not meet the given ones, an error should be raised.

5. (optionally) Extend the infrastructure developed in step 3 to support side conditions on arguments. For example, the function \( \text{head} :: [a] \to a \) is parametric only when it is applied to a non-empty list. Thus, the parametricity statement becomes

\[
\forall A :: a \to a' \to \text{Bool}. \ \forall xs :: [a]. \ \forall ys :: [b]. \\
\text{not} (\text{null} \ xs) \to \text{rel_list} \ A \ xs \ ys \to A (\text{head} \ xs) (\text{head} \ ys)
\]

6. (optionally) Integrate the command with the existing definitional mechanisms like \text{definition}, \text{primrec}, and \text{primcorec}.

7. (optionally, research oriented) When operations from type classes are used, additional preconditions must restrict the relations to those that “respect the operations”. Develop support for type classes. This may include the following aspects:

• Generating respectfulness conditions for type classes.
• Proving respectfulness for type class instantiation.

**Deliverables**

The following deliverables are due at the end of the project:

**Final report** The final report should consist of an introduction; a theoretical background section; one or more sections describing the modelling, implementation and evaluation; and a conclusion. The report may be written in English or German. Two copies of the report must be delivered to the supervisor.

**Isabelle/ML code** Complete Isabelle/ML development that runs with a recent developer’s version.

**Presentation** At the end of the project, a presentation of 20 minutes must be given during an InfSec group seminar. It should give an overview and discuss the most important highlights of the work.

**References**


