A Logic for Parametric Polymorphism

Gordon Plotkin*       Martín Abadi†

Abstract

In this paper we introduce a logic for parametric polymorphism. Just as LCF is a logic for the simply-typed λ-calculus with recursion and arithmetic, our logic is a logic for System F. The logic permits the formal presentation and use of relational parametricity. Parametricity yields—for example—encodings of initial algebras, final co-algebras and abstract datatypes, with corresponding proof principles of induction, co-induction and simulation.

1 Introduction

In this paper we introduce a logic for parametric polymorphism, in the binary relational sense of Reynolds [Rey83]. Just as LCF is a first-order logic for the simply-typed λ-calculus, with recursion and arithmetic, so our logic is a second-order logic for System F. It is intended as a step towards a general logic of polymorphically typed programs. The terms are those of the second-order λ-calculus, and the formulae are built from equations and relations by propositional operators and quantifiers over elements of types, or over types, or over relations between types. The logic permits the formal presentation and use of relational parametricity, which is expressed by an axiom schema. Parametricity yields—for example—encodings of initial algebras, final co-algebras and abstract datatypes, with corresponding proof principles of induction, co-induction and simulation.

Our first goal is to provide a formal system for arguments that exploit relational parametricity. In all models of System F, standard constructions such as products and initial algebras are available in a weak sense (see e.g. [Böh85, IIas90, RP90, Wra89]). If the models are relationally parametric, then these constructions become universal constructions in the usual sense of category theory. Bainbridge, Freyd, Scedrov, and Scott have given such results for the parametric Per model [BFSS90], and Hasegawa and Wadler for classes of models [IIas90, Wad89]. Hasegawa [IIas91] has shown that the second order minimal model—arising from the maximal consistent theory of Moggi and Statman—is parametric. By defining a formal logic, we hope to display the assumption of relational parametricity in a clear way, and to be able to obtain simple, general arguments for the results.

Our second goal is to pursue the idea of LCF. In LCF, a logic is given over a simply-typed λ-calculus with recursion and arithmetic. This λ-calculus is suitable for denotational semantics, and thus the corresponding logic acts as a rather powerful logic of programs. However, the simply-typed λ-calculus is inadequate for dealing with programming languages with abstract or polymorphic types, and an extension of the kind considered here is needed. The present work is but an intermediate step: it does not include recursion, at either

*Department of Computer Science, University of Edinburgh, King's Buildings, Edinburgh EH9 3JZ, UK. Part of this work was completed while at Digital Equipment Corporation, Systems Research Center.

†Digital Equipment Corporation, Systems Research Center. 130 Lytton Avenue, Palo Alto, California 94301, USA.
the level of values or the level of types. Moreover, one may also wish to consider richer

type systems (e.g., that of $F_\omega$ or even Constructions), richer logics (e.g., that of Topos

Theory), or more general notions of computation (e.g., as in Moggi's suggestions for the use

of monads). In yet another direction, we may consider a formal theory of subtypes, based,

say, on $F_\leq$ [CG91, CG92].

There has been some work along related lines. Abadi, Cardelli, and Curien gave a system

with very elementary judgments and few relations, and with a syntax as close as possible
to the basic typed $\lambda$-calculus [ACC93]. Mairson used second-order logic over the untyped

$\lambda$-calculus [Mai91]. He interpreted the types of System F as relations; as it happens his

interpretation of polymorphic types is not parametric, but it could easily be made so. The

second-order theory of subtypes of Cardelli, Martini, Mitchell and Scedrov embodies some

aspects of parametricity via an equational rule [CMMS91].

In Section 2 of this paper we present the logic. Apart from axioms giving the equational

theory of System F, the only non-logical principle is a schema for parametricity; further, only

intuitionistic reasoning is employed. Other axioms may well be possible: one would expect

that intuitionistic reasoning is (unsurprisingly) consistent with various choice principles. The

availability of these principles would be a miniature form of Pitts' result that polymorphism

can be considered as set-theoretic as long as one works constructively [Pit87]. A model can be

provided as in [BFSS90]; it remains to consider a more abstract notion and other examples.

We also compare other notions of polymorphism: we present a tentative formalisation of

Strachey's idea; we consider the dinatural approach of Bainbridge et al; and, finally, we

make the evident generalisation of Reynolds' binary relations to relations of other degrees.

We conjecture that if two terms of System F are equal in all models parametric in Strachey's

sense then they are also equal in all models parametric in Reynolds's sense. We prove that a

schema expressing dinaturality is a consequence of the schema expressing binary relational

parametricity. The relationship between the various notions of relational parametricity is

unknown.

In Section 3 we consider a variety of constructions showing the availability of finite

products and sums, second-order existential types, initial algebras and final co-algebras.
The constructions are by now standard; the main point here is that we can derive their

properties within our logic. We also consider the logical properties of the constructions. For

example, we give a general induction principle for initial algebras and a general bisimulation

(or co-induction) principle for final co-algebras [AM89, Smy91, Pit92]. For second-order

existential types, parametricity becomes a "simulation principle", namely that if there is a

relation between two types respected by the two corresponding "implementations" then

the corresponding elements of the existential type (the abstract types) are equal. Thus we

also obtain a proof rule for abstract types, along the lines envisaged by Mitchell [Mit91].

Presumably one would obtain similar principles relative to relational parametricity of other

degrees, and, again, it would be interesting to know the relationships between these. In

both Sections 2 and 3, we generally omit proofs.

2 Basic Logic

In this section we present the basic logic. The types and terms are those of System F and

they are given by the grammar:

\[
\begin{align*}
\text{Types:} & \quad A ::= X | A \to B | \forall X. A \\
\text{Terms:} & \quad t ::= x | \lambda x : A. t | u(t) | \Lambda X. t \mid t(A)
\end{align*}
\]

Here $X$ ranges over type variables, $x$ over ordinary variables; we use notations such as

$A[X]$ to indicate possible occurrences of variables in expressions, and then may write for