Strongly Typed Term Representations in Coq

**Nick Benton · Chung-Kil Hur ·
Andrew J. Kennedy · Conor McBride**

Abstract There are two approaches to formalizing the syntax of typed object languages in a proof assistant or programming language. The *extrinsic* approach is to first define a type that encodes untyped object expressions and then make a separate definition of typing judgements over the untyped terms. The *intrinsic* approach is to make a single definition that captures well-typed object expressions, so ill-typed expressions cannot even be expressed. Intrinsic encodings are attractive and naturally enforce the requirement that metalanguage operations on object expressions, such as substitution, respect object types. The price is that the metalanguage types of intrinsic encodings and operations involve non-trivial dependency, adding significant complexity. This paper describes intrinsic-style formalizations of both simply-typed and polymorphic languages, and basic syntactic operations thereon, in the Coq proof assistant. The Coq types encoding object-level variables (de Bruijn indices) and terms are indexed by both type and typing environment. One key construction is the boot-strapping of definitions and lemmas about the action of substitutions in terms of similar ones for a simpler notion of renamings. In the simply-typed case, this yields definitions that are free of any use of type equality coercions. In the polymorphic

N. Benton (✉) · A. J. Kennedy
Microsoft Research, 7 J J Thomson Avenue, Cambridge, CB3 0FB, UK
e-mail: nick@microsoft.com

A. J. Kennedy
e-mail: akenn@microsoft.com

C.-K. Hur
Max Planck Institute for Software Systems, Campus E1 4, 66123 Saarbrücken, Germany
e-mail: gil@mpi-sws.org

C. McBride
Department of Computer and Information Sciences, University of Strathclyde,
26 Richmond Street, Glasgow, G1 1XH, Scotland
e-mail: conor@strictlypositive.org