Notre Dame Journal of Formal Logic
Volume 30, Number 3, Summer 1989

Propositional Functions and Families of Types

JAN M. SMITH*

Abstract When specifying the task of a computer program, it is often natural to use recursion on a data type. In Martin-Löf's type theory, a universe must be used when defining a propositional function by recursion. Using a logical framework for type theory, formulated by Martin-Löf, an extension of type theory is proposed by which propositional functions can be directly defined without using a universe.

1 Introduction In order to capture some programmers' errors several computer languages, like Pascal and ML, are equipped with a type system. Using the Curry–Howard interpretation of propositions as types (see [4] and [8]), or, as we shall say here, propositions as sets, a type system can be made strong enough to be used to specify the task a program is supposed to do. This is one of the bases for Martin-Löf's suggestion in [12] to use his formulation of type theory for programming; his ideas are exploited in [14] and there are several computer implementations of type theory (see [3] and [16]). Similar ideas are also behind Coquand and Huet's calculus of constructions [2].

The idea of propositions as sets is closely related to the intuitionistic explanations of the logical constants given by Heyting [7]. In Martin-Löf's type theory the interpretation of propositions as sets is fundamental, since the notions of proposition and set are identical. So a logical constant is definitionally equivalent to the corresponding set constant. Conversely, every set forming operation can be viewed as a logical constant, although some sets are more natural to think of as data types.

When using Martin-Löf's type theory for programming one often has to use strong principles, such as a universe or well-orderings, when writing specifications or defining data types. For instance, a universe must be used when

*The convenient possibility of defining propositions by induction on sets has been made clear for a long time by people using type theory as a programming logic. For the extension proposed here I am particularly indebted to Per Martin-Löf, Bengt Nordström, and Kent Petersson for many discussions.

Received April 13, 1989