Applications II
A Formal Semantics for a Language with Type Extension

Peter Bancroft
Queensland University of Technology
Brisbane 4001, Australia
e-mail: bancroft@fit.qut.edu.au

Ian Hayes
University of Queensland
Brisbane 4072, Australia
e-mail: ianh@cs.uq.oz.au

Abstract. The purpose of this paper is to give a formal semantics for a language which includes type extension. Used in association with pointer variables, this forms the basis of object-orientation in the languages Oberon and Oberon-2 which have evolved from Modula-2. The focus is on the meaning of assignment because this is the important difference between such languages and the strongly-typed Pascal family. An abstract syntax is defined using the Z notation and the static and dynamic semantics are given in a denotational style.

1 Introduction

In languages such as Pascal and Modula-2, program variables are declared to be of a certain type and may then be assigned only values that belong strictly to that type. In the latest of this family of languages, Oberon [RW92, Wir88b] and Oberon-2 [MW91], this restriction has been relaxed for record and pointer types to achieve inheritance. The same approach to the development of an object-oriented Ada9X is suggested in [Sei91].

Type extension [Wir88a] is the creation of new record types from existing record types by the addition of extra fields. A record extension is considered to be compatible with the original record type in that the values of the extension may be assigned to variables of that type. Of course the extra fields cannot be stored in such a variable and are "dropped off". This is known as the projection of the extended value onto the original type. Being able to extend record types in this way allows the creation of a record hierarchy, one of the common features of object-oriented languages.

Pointer types may also be extended. A pointer variable is declared to refer to a particular base record type; however it may be assigned a reference to a record extension of its base type. The difference between this pointer assignment and the corresponding record assignment is that the extra fields of the record extension remain accessible through the pointer. Such access is subject to a type guard (which is essentially an assertion that the dynamic type of the pointer is an extension of the type named in the guard). For example, in Oberon we write,

vr (tid) ^ . fld

meaning that variable vr has been extended so that it has dynamic type tid, a pointer to a record with a field name fld. Such an expression could be used in assignment as an r-value to read the content of fld or as an l-value to update the field's value.