1 Introduction

The cartesian closed monoid [6] provides an algebraic semantics for an un-
typed lambda calculus; this language includes a surjective pairing (*cons*),
partial functions, non-strict parameter passing and recursion. The cartesian
closed category [7] (with natural number objects) provides an algebraic se-
manics for the typed lambda calculus, which includes cartesian products,
discriminated unions, total functions and iteration. Topos theory [7] extends
the range of definable functions and types.

This paper applies categorical method to the definition of a range of conven-
tional programming languages, which include conditionals, non-determinism,
non-termination, strictness, recursion, higher order procedures and even
communications. A good example is Dijkstra’s language with predicate
transformer semantics [1]. To achieve this greater generality, we use a simple
kind of 2-category theory, i.e., a category whose homsets are preordered and
whose composition is monotonic.

Section 2 reviews the relevant categorical concepts. We need a preorder
with bottom element (⊥), the *glb* (greatest least bound, ∩), and limits of
ascending chains. The intended meaning of *q* ⊆ *p* in this order is that pro-
gram *p* is as good as *q* or better, for any purpose and in any context of use.
A simple 2-category has such a preorder defined on each of its homsets. The
definition of a natural transformation can be weakened to that of an up-
simulation, or its dual, a down-simulation (called 2-natural transformation
In order to give an algebraic semantics to a wider variety of programming languages, including strict, lazy, and non-deterministic languages, the definition of an adjunction will be weakened, so that its unit and counit are simulations. In this way, simple algebraic distinction are made between familiar functional models and more conventional languages. For example the command abort of Dijkstra's language is the bottom element of the pre-order (the worst of all programs). This is also a zero morphism in Dijkstra's language, i.e., a natural transformation from the identity functor to itself. In a language like CSP (with communication) [4], it is an up-simulation; and in Miranda (with partial functions and non-strict parameter) [9], it is a down-simulation. The definition of a product can also be weakened to accommodate non-determinism.

Section 3 applies the theoretical results of section 2 to a selection of constructors in a range of familiar languages. We show that they have familiar categorical interpretations. The method explored in this section will be extended in a later paper [5] to provide a uniform algebraic semantics to a wider variety of programming languages, including strict, lazy and non-deterministic languages.

2 Categorical Preliminaries

This section recalls the preliminary concepts in 2-category theory [2]. We assume that familiarity with most of the basic definition of category theory [7]

A 2-category C is a category with two different compositions ; (horizontal composition) and ; (vertical composition) such that

- each of its homsets C(b, c) is a category with the vertical composition.
- identities for the horizontal composition are also identities for the vertical composition.