Berkeley FP User's Manual, Rev. 4.1

by

Scott Baden

ABSTRACT

This manual describes the Berkeley implementation of Backus' Functional Programming Language, FP. Since this implementation differs from Backus' original description of the language, those familiar with the literature will need to read about the system commands and the local modifications.

August 3, 1987

1. Background

FP stands for a *Functional Programming* language. Functional programs deal with *functions* instead of *values*. There is no explicit representation of state, there are no assignment statments, and hence, no variables. Owing to the lack of state, FP functions are free from side-effects; so we say the FP is *applicative*.

All functions take one argument and they are evaluated using the single FP operation, *application* (the colon ':' is the apply operator). For example, we read + : <3 4> as ''apply the function '+' to its argument <3 4>''.

Functional programs express a functional-level combination of their components instead of describing state changes using value-oriented expressions. For example, we write the function returning the *sin* of the *cos* of its input, i.e., *sin*(*cos*(*x*)), as: sin @ cos. This is a *functional expression*, consisting of the single *combining form* called *compose* ('@' is the compose