FUN — Untyped — Environment

Grigore Roşu and Traian Florin Şerbănuţă (grosu,tsarbanu@illinois.edu)
University of Illinois at Urbana-Champaign

Abstract

This is a basic extension definition of the untyped FUN language. FUN is a pedagogical and research language that captures the essence of the functional programming paradigm, extended with several features often encountered in functional programming languages. Its syntax is similar to that of the Standard ML family of languages, but it does not include any type system or any of its extensions. Functions can be defined anywhere and can have other terms in the language. FUN is well-suited for basic research on programming language design, because it is very small and it has a well-defined syntax and semantics. It is also well-suited and is highlight some of FL samples, FUN includes the following features:

- Integer, Boolean, and Unit primitive types;
- Identifiers, which can be used as expressions, including other fun, Unit, or Int and can contain operator characters and their elements are comma-separated, e.g., f1,f2,f3;
- Tuples, which are of the form (e1,...,en) for arbitrary terms. Construction terms start with a capital letter (while any other identifier in the language begins with a lowercase letter), and they can be followed by an arbitrary number of comma-separated expressions enclosed in parentheses (parentheses are not needed when the constructor takes no arguments). Tuples are also used as lists, e.g., Cons(1,Cons(2,Cons(3,Nil))) is a term that represents the list [1,2,3], and Cons(1,Cons(2,Cons(3,Cons(4,Nil)))) represents the list [1,2,3,4]. Tuples are also used as arguments to binary operations, e.g., 1+(2,3) is a term that represents the term 1+2+3. Tuples are also used to define lists, e.g., [1,2,3] is a term that represents the term Cons(1,Cons(2,Cons(3,Nil))). Tuples are also used to define sets, e.g., {1,2,3} is a term that represents the term Cons(1,Cons(2,Cons(3,Nil))). Tuples are also used to define maps, e.g., {1 |-> 2, 3 |-> 4} is a term that represents the term Cons(Cons(1,2),Cons(3,4)). Tuples are also used to define functions, e.g., fun f(x,y) = x+y is a term that represents the term f(x,y) = x+y.
- Function definitions, which are of the form fun f(p1,...,pn) = e, where f is a function name, p1,...,pn are parameters, and e is the function body. Function definitions can be nested, and they can be followed by other terms in the language. Function definitions can also be anonymous, in which case they are of the form fun(p1,...,pn) = e. Function definitions can also be recursive, in which case they are of the form fun rec f(p1,...,pn) = e. Function definitions can also be mutually recursive, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be curried, in which case they are of the form fun f(p1)(p2)...(pn) = e. Function definitions can also be pattern matching, in which case they are of the form fun f(p1,...,pn) = e1 | ... | en. Function definitions can also be pattern matching and curried, in which case they are of the form fun f(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and recursive, in which case they are of the form fun rec f(p1,...,pn) = e1 | ... | en. Function definitions can also be pattern matching and mutually recursive, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be pattern matching and anonymous, in which case they are of the form fun(p1,...,pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous, in which case they are of the form fun(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and recursive, in which case they are of the form fun rec f(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and mutually recursive, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be pattern matching and curried and anonymous, in which case they are of the form fun(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and recursive and anonymous, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and mutually recursive and anonymous, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be pattern matching and curried and anonymous and recursive, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and mutually recursive and anonymous, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous, in which case they are of the form fun rec(p1)(p2)...(pn) = e1 | ... | en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive, in which case they are of the form fun rec f1,...,fn = e1,...,en. Function definitions can also be pattern matching and curried and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive and mutually recursive and anonymous and recursive