CIS 425 - Week 3 - Lecture 1

Grayson Guan, BoWen Hou

15 Oct 2019

1 Higher order function

Higher order functions is able to take a function as an argument or return a function In this lecture we are using Racket as the example language. Racket is a general-purpose, multi-paradigm programming language based on the Scheme dialect of Lisp. It is designed to be a platform for programming language design and implementation. It is also a programming language looks like both java and cpp. It uses list as it's data structure exclusively.

Basic Form

$$E = m|(+EE)$$

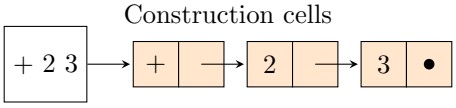
Left associative expression example

$$2 + 3 + 1 \rightarrow (\ +\ (\ +\ 2\ 3\ )\ 1\ )$$

Right associative expression example

$$2 + 3 + 1 \rightarrow (\ +\ 2\ (+\ 3\ 1)\ )$$

Construction cells



Beginning of the cell (where data lives) is called "contents of the address part of register number" (CAR). Ending of the cell (where pointer lives) is called "contents of the decrement part of register number" (CDR)

2 Example Racket Code

#Racket

>define x (+ 3 2)
>x
5

>(define z '(+ 2 3))
>z
>'(+ 2 3)
>(eval z)
5

>(list 2 3 4)
'(2 3 4)
>(define w(list 2 3 4)
>car w
2
>(eval w)
"Error..."
(Numbers are not operations)

3 Language Concepts

In order to understand the behaviour of our sample code, there are some concepts we must be aware of.

(1) _CALL_BY_VALUE: The call by value method of passing arguments to a function copies the actual value (evaluate expression for actual value)of an argument into the formal parameter of the function.

(2) dynamically typed: A language is dynamically-typed if the type of a variable is checked during run-time. Common examples of dynamically-typed languages includes JavaScript, Objective-C, PHP, Python, Ruby, Lisp, and Tcl.

(3)TYPE_INTERFERENCE: Type inference refers to the automatic detection of the data type of an expression in a programming language. It is a feature present in some strongly statically typed languages. It is often characteristic of functional programming languages in general.

4 Example ML Code

#ML

>>If true then "a" else 99;
(this will result in error due to different type between "a" (string) and 99 (int)).

>>fun fun loop x = loop x;
    val loop - fun : 'a →'b

>>(2, true);
    val it = (2, true) : int *bool

>>[ ];
    val it = [ ] : 'a list

(Note: cons (1, [ ]) ≡ 1 :: []

>>1 :: $true$ :: "a" :: [];
    (Error)