Functional programming - Tail Call Optimization

Pavol Karlík (xkarli05), Markéta Jančová (xjanco06)

November 2019

Abstract

Tail call is a subroutine call that is to be performed as a final action of a procedure. The subroutine is said to be *tail-recursive* if such call might lead to the same subroutine being called again. When a subroutine is called, certain information in the form of a stack frame, such as return address and function variables is saved on a call stack. However, in the case of tail-recursive functions there is no need to add stack frame to a call stack. Instead, the stack frame of the current procedure can be replaced by the frame of the tail call. Specifically, the subroutine call in this case can be replaced with a jump instruction. Tail-recursive functions are optimized by eliminating unnecessary call stack operations, hence the process is called *Tail Call Elimination*.

Tail Call Elimination is considered a standard in every functional language. Thanks to this feature, the tail-recursive methods can run as fast as their imperative counterparts. We will discuss possible implementations of Tail Call Elimination in various languages and their limitations.