Package 'tailr'

December 21, 2018

Version 0.1.3

Title Automatic Tail Recursion Optimisation

Description Implements meta-programming functions for automatically translating recursive functions into looping functions or trampolines.

License GPL-3

Encoding UTF-8

Language en-GB

LazyData true

ByteCompile true

Depends R (>= 3.2), foolbox

Imports rlang (>= 0.3.0), glue

Suggests covr, testthat, microbenchmark, compiler, pmatch (>= 0.1.4)

RoxygenNote 6.1.1

URL https://github.com/mailund/tailr

BugReports https://github.com/mailund/tailr/issues

NeedsCompilation no

Author Thomas Mailund [aut, cre]

Maintainer Thomas Mailund <mailund@birc.au.dk>

Repository CRAN

Date/Publication 2018-12-20 13:10:03 UTC

R topics documented:

build_transformed_function

Construct the expression for a transformed function body.

Description

This is where the loop-transformation is done. This function translates the body of a recursive function into a looping function.

Usage

```
build_transformed_function(fun, fun_name)
```

Arguments

fun	The original function
fun_name	The name of the function we are transforming

Value

The body of the transformed function.

can_loop_transform_

Tests if a function, provided by its name, can be transformed.

Description

This function analyses a recursive function to check if we can transform it into a loop or trampoline version with `transform`. Since this function needs to handle recursive functions, it needs to know the name of its input function, so this must be provided as a bare symbol.

Usage

```
can_loop_transform_(fun)

can_loop_transform(fun)
```

Arguments

fun	The function to check. Must be provided by its (bare symbol) name.

Functions

- `can_loop_transform_`: This version expects `fun` to be quosure.
- `can_loop_transform`: This version quotes `fun` itself.

Examples

```
factorial <- function(n)
    if (n <= 1) 1 else n * factorial(n - 1)
factorial_acc <- function(n, acc = 1)
    if (n <= 1) acc else factorial_acc(n - 1, n * acc)

can_loop_transform(factorial) # FALSE -- and prints a warning
can_loop_transform(factorial_acc) # TRUE

can_loop_transform_(rlang::quo(factorial)) # FALSE -- and prints a warning
can_loop_transform_(rlang::quo(factorial_acc)) # TRUE
```

`loop_transform` *Transform a function from recursive to looping.*

Description

Since this function needs to handle recursive functions, it needs to know the name of its input function, so this must be provided as a bare symbol.

Usage

```
loop_transform(fun, byte_compile = TRUE, set_srcref = TRUE)
```

Arguments

`fun`	The function to transform. Must be provided as a bare name.
`byte_compile`	Flag specifying whether to compile the function after transformation.
`set_srcref`	Flag specifying whether the "srcref" attribute should be set to the original value. If you do this, you can print the modified function and it will look like the original, but printing it will not show the actual, tranformed, source.

Index