FullBlockClosure

Last updated at 6:54 am UTC on 1 November 2018

Subclass of BlockClosure.

Instances of FullBlockClosure represent blocks, a sequence of statements inside square brackets that can be evaluated at any time via one of the value messages (value, value:, value:value:, ... valueWithArguments:), which answer their last statement. Blocks therefore allow deferred evaluation and so are used to buikld control structures where a sequence of statements are evaluated or not depending on other values in the program.

FullBlockClosure is a refinement of BlockClosure that allows the block to use its own method to hold its code instead of embedding that code within its home method.

Implementation:

A FullBlockClosure is a closure that can be independent of any outerContext if desired. It has its own method (currently reusing the startpc inst var) and its own receiver. outerContext can be either a Context or nil.

This closure design, implemented by Eliot Miranda and Clement Bera along with the sista work aims to simplify the block closure model while enhacing its capabilities. It allows lazy decompilation of closures and fast machine code dispatch in Cog's JIT, while allowing inlining of methods and blocks to be independent from their enclosing blocks.

At closure creation time, the bytecode specifies:

- the compiledBlock to execute when executing this block's code (in the literal frame)
- if the receiver is the current receiver or a receiver passed on stack before the copied values.
- if the closure needs an outerContext. outerContexts are used for non local returns and debugging. Blocks with non local returns have to set their outerContext. For other blocks (97% of blocks), it's a trade-off between performance and debuggability.

```
Instance Variables (inherited)
        numArgs                         <SmallInteger>
        outerContext:                   <Context|nil>
        compiledBlock(startpc) <CompiledBlock>

Instance Variables
        receiver:                               <Object>

numArgs
        - the number of arguments the block expects. This is superfluous; the number of arguments can be obtained from the receiver's compiled

outerContext
        - the Context of the method or block activation in which the receiver is created.

compiledBlock(startpc)
        - reused to refer to the CompiledBlock that implements the receiver's code.

receiver
        - the receiver of the message that created the block's home method activation.
```