PAT - Assignment 1

Mikkel Willén
bmq419

May 13, 2024

Task 1

$$inverter : \lambda p.\ \lambda x [\![invint]\!](p, x)$$

Task 2

$$[\![invint]\!] : \lambda p. \lambda y.(inverter(p)(y))$$

Task 3

We run the Janus program for the Fibonacci pair calculation forward with an initial store of $n = 4$, $x_1 = 0$ and $x_2 = 0$.

`Initial State`
  $(n, x_1, x_2) = (4, 0, 0)$
`Initialization`
  `Increment` x_1 - $(n, x_1, x_2) = (4, 1, 0)$
  `Increment` x_2 - $(n, x_1, x_2) = (4, 1, 1)$
`First iteration`
  `Decrement` n - $(n, x_1, x_2) = (3, 1, 1)$
  `Add` x_2 `to` x_1 - $(n, x_1, x_2) = (3, 2, 1)$
  `Swap` x_1 `and` x_2 - $(n, x_1, x_2) = (3, 1, 2)$
`Second iteration`
  `Decrement` n - $(n, x_1, x_2) = (2, 1, 2)$
  `Add` x_2 `to` x_1 - $(n, x_1, x_2) = (2, 3, 2)$
  `Swap` x_1 `and` x_2 - $(n, x_1, x_2) = (2, 2, 3)$
`Third iteration`
  `Decrement` n - $(n, x_1, x_2) = (1, 2, 3)$
  `Add` x_2 `to` x_1 - $(n, x_1, x_2) = (1, 5, 3)$
  `Swap` x_1 `and` x_2 - $(n, x_1, x_2) = (1, 3, 5)$
`Fourth iteration`
  `Decrement` n - $(n, x_1, x_2) = (0, 3, 5)$
  `Add` x_2 `to` x_1 - $(n, x_1, x_2) = (0, 8, 5)$
  `Swap` x_1 `and` x_2 - $(n, x_1, x_2) = (0, 5, 8)$
`Exit condition`
  `Exit condition is true, since` $n = 0$

Task 4

We run the Janus program for the Fibonacci pair calculation backward with an initial store of $n = 0$, $x_1 = 5$ and $x_2 = 8$.

`Initial state`

$(n, x_1, x_2) = (0, 5, 8)$

`First iteration`

`Swap` x_1 `and` x_2 `-` $(n, x_1, x_2) = (0, 8, 5)$

`Subtract` x_2 `to` x_1 `-` $(n, x_1, x_2) = (0, 3, 5)$

`Increment` n `-` $(n, x_1, x_2) = (1, 3, 5)$

`Second iteration`

`Swap` x_1 `and` x_2 `-` $(n, x_1, x_2) = (1, 5, 3)$

`Subtract` x_2 `to` x_1 `-` $(n, x_1, x_2) = (1, 2, 3)$

`Increment` n `-` $(n, x_1, x_2) = (2, 2, 3)$

`Third iteration`

`Swap` x_1 `and` x_2 `-` $(n, x_1, x_2) = (2, 3, 2)$

`Subtract` x_2 `to` x_1 `-` $(n, x_1, x_2) = (2, 1, 2)$

`Increment` n `-` $(n, x_1, x_2) = (3, 1, 2)$

`Fourth iteration`

`Swap` x_1 `and` x_2 `-` $(n, x_1, x_2) = (3, 2, 1)$

`Subtract` x_2 `to` x_1 `-` $(n, x_1, x_2) = (3, 1, 1)$

`Increment` n `-` $(n, x_1, x_2) = (4, 1, 1)$

`Undo initialization`

`Increment` x_2 `-` $(n, x_1, x_2) = (4, 1, 0)$

`Increment` x_1 `-` $(n, x_1, x_2) = (4, 0, 0)$

Task 5

Increment and decrement operators $(+=, -=)$.
In the forward direction, these operators modify the value of a variable by adding or subtracting another value.
In the backward direction, these operators are reversed so $+=$ becomes $-=$ and $-=$ becomdes $+=$.

Swap operator (<=>).
In the forward direction, this operator swaps the values of two variables.
In the backward direction, this operator also swaps the value pf two variables, as this operator is its own inverse.

Reversible loop (from-loop-until) with textual representation

$$\text{from } e_1 \text{ do } s_1 \text{ loop } s_2 \text{ until } e_2$$

In the forward direction, its starts by evaluating the e_1 condition. If e_1 evaluates to true, then the loop continues, otherwise, it skips the loop. s_1 is evaluated, when e_1 evaluates to true, and is then repeatedly evaluated, as long as e_2 evaluates to false. In the body of the loop s_2 is executed repeatedly as long as e_2 evaluates to false and as long as e_1 evaluates to false, after the first iteration. The loop exits, when e_2 evaluates to true.
In the backward direction, its starts by evaluating the e_2 condition. If e_2 evaluates to true, then the loop continues, otherwise, it skips the loop. s_1 is evaluated, when e_2 evaluates to true, and is then repeatedly

evaluated, as long as e_1 evaluates to false. In the body of the loop s_2 is executed repeatedly as long as e_1 evaluates to false and as long as e_2 evaluates to false, after the first iteration. The loop exits, when e_1 evaluates to true.

Task 6

Forward direction:
Increment: $\langle x_1 + = e_2, \sigma\rangle \rightarrow \sigma[x_1 \mapsto \sigma(x_1) + e_2]$
Decrement: $\langle x_1 - = e_2, \sigma\rangle \rightarrow \sigma[x_1 \mapsto \sigma(x_1) - e_2]$
Swap: $\langle x_1 <=> x_2, \sigma\rangle \rightarrow \sigma[x_1 \mapsto \sigma(x_2), x_2 \mapsto \sigma(x_1)$
Loop:
  Initial: $\langle$from e_1 loop s_2 until $e_2, \sigma\rangle \rightarrow \sigma$ if $e_1[\sigma] =$ **false**
  Iterate: $\langle$from e_1 loop s_2 until $e_2, \sigma\rangle \rightarrow \langle s_2;$ from e_1 loop s_2 until $e_2, \sigma'\rangle$ if $e_1[\sigma] =$ **true** and $e_2[\sigma] =$ **false**
  Terminate: $\langle$from e_1 loop s_2 until $e_2, \sigma\rangle \rightarrow \sigma$ if $e_2[\sigma] =$ **true**

Backward direction:
Re-Increment: $\langle x_1 - = e_2, \sigma\rangle \rightarrow \sigma[x_1 \mapsto \sigma(x_1) - e_2]$
Re-Decrement: $\langle x_1 + = e_2, \sigma\rangle \rightarrow \sigma[x_1 \mapsto \sigma(x_1) + e_2]$
Re-Swap: $\langle x_2 <=> x_1, \sigma\rangle \rightarrow \sigma[x_2 \mapsto \sigma(x_1), x_1 \mapsto \sigma(x_2)$
Loop:
  Re-Initial: $\langle$from e_1 loop s_2 until $e_2, \sigma\rangle \rightarrow \sigma$ if $e_2[\sigma] =$ **false**
  Re-Iterate: $\langle$from e_1 loop s_2 until $e_2, \sigma\rangle \rightarrow \langle s_2^{-1};$ from e_1 loop s_2 until $e_2, \sigma'\rangle$ if $e_2[\sigma] =$ **true** and $e_1[\sigma] =$ **false**
  Re-Terminate: $\langle$from e_1 loop s_2 until $e_2, \sigma\rangle \rightarrow \sigma$ if $e_1[\sigma] =$ **true**

where s_2^{-1} is the sequence of statements s_2 executed in reverse.