15.2 A text filter

Description

Here we show an example in the domain of text processing. On the Macintosh, one may set an option that implies that two blanks in a text is replaced by a dot (`'.'`) and a blank. The coroutine `doubleSpaceToDot` below scans through a text string and makes these replacements.

```
charFilter: obj
   nextChar: var char
   input: var String
   i: var integer
   next -> c: var char:
      i := i + 1
      c := input.get[i]
   send(ch: var char):
      nextChar := ch
   doubleSpaceToDot: obj
      ch: var char
      doubleSpaceToDot.suspend
      cycle
         ch := next
         if (ch = ' ') :then
            ch := next
            if (ch = ' ') :then
               send('.')
               doubleSpaceToDot.suspend
               send(' ')
               doubleSpaceToDot.suspend
            :else
               send(' ')
               doubleSpaceToDot.Suspend
               send(ch)
               doubleSpaceToDot.Suspend
         :else
            send(ch)
            doubleSpaceToDot.Suspend
   input := "Hello world  What a nice day!\n"
   loop: do
      cycle
         doubleSpaceToDot.call
         put(nextChar)
         if (nextChar = ascii.newline) :then
            leave(loop)
```

The central element of this program is the coroutine object `doubleSpaceToDot`. When the object is generated, it starts by executing a `suspend`.

The object `charFilter` has a local String `input`, which holds the text string to be filtered. For the purpose of the example, we assign it the `String "Hello world What a nice day!\n"`.

The `charFilter` then starts executing a cycle and the first statement in the `cycle` is a call of `doubleSpaceToDot`. This implies that `doubleSpaceToDot` is resumed and executes its `cycle`-statement.

It assigns the next char in input to `ch` by executing `ch := next`.

If `ch = ' '`, it reads the next char and checks if it is also a blank. If blank it sends a dot (`'.'`) and suspends executions, and when resumed it sends a blank. If the next char is not a blank, it sends a blank one at a time and suspends in between.

© Copyright by Ole Lehrmann Madsen and Birger Møller-Pedersen 2023.

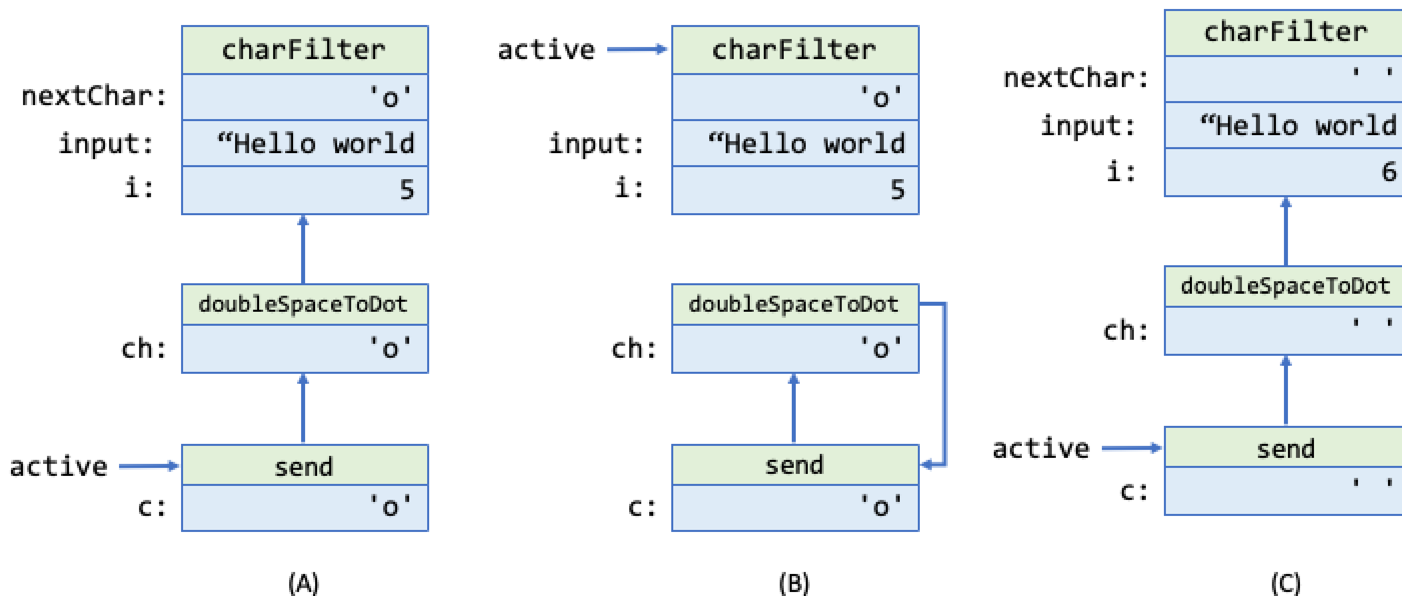
The `else`-part of the outermost `if`-statement, is executed If ch is not a blank and here it just sends ch and suspends execution.

A simple version using a method

A coroutine object may invoke methods and a method may suspend the coroutine object. In the next example we use this to make a simpler version of `doubleSpaceToDot`. Here the `send`-method is made a local method of `doubleSpaceToDot` and `send` executes the `suspend`, suspending the enclosing `doubleSpaceToDot` coroutine:

```
doubleSpaceToDot: obj
  ch: var char
  send(ch: var char):
     c := ch
     doubleSpaceToDot.suspend
  doubleSpaceToDot.suspend
  cycle
     ch := next
     if (ch = ' ') :then
        ch := next
        if (ch = ' ') :then
           send('.')
           send(' ')
        :else
           send(' ')
           send(ch)
     :else
        send(ch)
```

When `send` executes a `suspend`, we have a situation where `doubleSpaceToDot` has invoked `send` after having read the `'o'` in `"Hello"` and thus `i = 5`. In the figure below, the left-side (A) shows the situation before suspend and the middle part (B) shows the situation after suspend.



The figure shows the activation stack. (A) shows that `charFilter` has called `doubleSpaceToDot`, which has called `send`. `Send` is currently the active object executing statements. Note that the elements of the activation stack may be objects as well as method objects. Here the bottom element is an object and the other elements are method objects.

When `send` executes `suspend`, the situation is shown at (B). `CharFilter.suspend` has the effect that control is

© Copyright by Ole Lehrmann Madsen and Birger Møller-Pedersen 2023.

returned to after `charFilter` has called `doubleSpaceToDot`. The arrow from `doubleSpaceToDot` illustrates that it is suspended and that `send` is at the top of its activation stack.

When `charFilter` makes another call to `doubleSpaceToDot`, the situation is as in the (C)-part, but with `ch = ' '` and `i = 6`. As can be seen, `doubleSpaceToDot` now again points to the calling `charFilter`-object.

© Copyright by Ole Lehrmann Madsen and Birger Møller-Pedersen 2023.