1 Curried vs. Uncurried Functions

Digression

Use of parenthesis:
1. \((A \rightarrow B) \rightarrow C\) Give me a function from A to B and I’ll give you C (Curried)
2. \(A \rightarrow (B \rightarrow C)\) Give me a item of type A and I will give you a function that maps from B to C
3. \(A*B \rightarrow C\) Give me a tuple\((A,B)\) and I’ll give you C (Uncurried)

Code samples for curried and uncurried functions...

\[
\text{NOTE: } x \text{ is of type A, and } y \text{ is of type B}
\]

\[
\text{let } \text{curry } f = \text{ fun } x \rightarrow \text{ fun } y \rightarrow f(x,y) \\
\text{let } \text{uncurry } f = \text{ fun}(x,y) \rightarrow (f x) y \text{ or } f x y
\]

Properties: \(\text{curry(uncurry}(A \rightarrow (B \rightarrow C)))=\text{curry(identity map)}\) \(\text{uncurry(curry}((A*B) \rightarrow C))=\text{uncurry(identity map)}\)

2 Continuation-Passing Style

First we develop a motivation for the CPS.

In the following examples: These are the definitions of \(\text{fst}\) and \(\text{snd}\) function.

\[
\text{let } \text{fst}(x,y) = x : (\alpha \ast \beta) \rightarrow \alpha \\
\text{let } \text{snd}(x,y) = y : (\alpha \ast \beta) \rightarrow \beta
\]

let \(f\ n\ s = \)
if \(n = 0\) then \(s\)
else let \(x = \text{fst } s\)
\(y = \text{snd } s\)
\(s’=(x-y, x+y)\)
in \(f\ (n-1)\ s’\)

Same as:

let \(f = \text{ fun } n \rightarrow \text{fun } s \rightarrow \)
if \(n = 0\) then \(s\)
else let \(x = \text{fst } s\)
\(y = \text{snd } s\)
\(s’=(x-y, x+y)\)
in \(f\ (n-1)\ s’\)
Let’s say that we know n early so that we make our first attempt at staging this function.

n = 2

let f n <s> =
  if n = 0 then s
  else let x = fst ~s
       y = snd ~s
       s’=(x-y,x+y)
       in ~(f (n-1) <s’>)

Specialize the function with n = 2:
<fun (a,b) ~(f 2 <(a,b)>)}>

This function has type: int→(int*int→int*int)

When this code is executed, we get the following:

<fun s~>
  let x = fst s
          y = snd s
          s’=(x-y,x+y)
          let x’ = fst s’
               y’ = snd s’
               s’’= (x-y,x+y)
               in s’’>

This code seems a bit costly. Fst and snd operations are not cheap. Creating tuples is also a costly operation, and we have 9 costly operations. We do not need all of this; optimization can be done at this time.

IDEALLY WE WOULD WANT THE CODE TO LOOK LIKE:
<fun (a,b) ~(a-b,(a+b),(a-b),(a+b))>

PROPOSED SOLUTION: Change the incoming type
int→(int*int→int*int)
let \( f \ n \ s = \)
    if \( n = 0 \) then \( s \)
    else let \( x = \text{fst} \ s \)
    \( y = \text{snd} \ s \)
    \( s' = (\langle -x - y \rangle, \langle -x + y \rangle) \)
    in \( f \ (n-1) \ s' \)

\(<\text{fun}(a,b)\rightarrow(\text{let}(x,y) = f \ 2 \ (\langle x \rangle, \langle y \rangle) \ \text{in} \ (>\langle -x, -y \rangle))>\)

**PROBLEM**
This function has code duplication of costly functions
which is also undesirable.

**TRUE SOLUTION:** Use CPS (for next time...)

**EXERCISE:** SEE IF GENERAL TRICKS FOR HANDLING CODE
EXPLOSION IN THIS EXAMPLE.