15-150 Assignment 3

Karan Sikka
ksikka@andrew.cmu.edu
G
2/7/12

1: Task 2.3

Claim: For all `l : (int * string) list`, `zip(unzip l)` $\cong$ `l`

Proof: The proof is by structural induction on `l`.

Case for []

To show `zip(unzip([]))` $\cong$ `l`:

```
  zip(unzip([]))                                   given
≅ zip(case [] of [] => ([],[]) | ...)              step
≅ zip([],[])                                       step
≅ case [] of [] => [] |...                         step
≅ []                                               step
```

Case for x::xs. Inductive hypothesis: `zip(unzip(xs))` $\cong$ `xs`

```
≅ zip(unzip(x::xs))
≅                                                  step

  zip(case x::xs of
          [] => ([],[])
        | x::xs => let val (pint, pstr) = x
                       val (ints, strs) = unzip(xs)
                   in (pint::ints,pstr::strs)
                   end)

≅                                                  step, evaluate case

  zip(let val (pint, pstr) = x
          val (ints, strs) = unzip(xs)
      in (pint::ints,pstr::strs)
      end)

≅ zip(pint::ints,pstr::strs)                       step, evaluate let
≅                                                  step

  case pint::ints of [] => []
                   | pint::ints => let val y::ys = pstr::strs
                                   in (pint,y)::zip(ints,ys)
                                   end
```

$\cong$ step, evaluate case

```
let val y::ys = pstr::strs
in (pint,y)::zip(ints,ys)
end
```

$\cong$	`(pint,pstr)::zip(ints,strs)`	step, evaluate let
$\cong$	`x::zip(unzip(xs))`	by definition in let
$\cong$	`x::xs`	IH

2: Task 2.4

Claim:
For all `l1 : int list` and `l2 : string list`,
`unzip(zip (l1,l2))` $\cong$ `(l1,l2)` Counterexample:

Let l1 be the value `[]:int list`
Let l2 be the value `["7"]:string list`

$\cong$ `unzip(zip (l1,l2))`
$\cong$ `unzip(zip ([],["7"]))`
$\cong$ `unzip(case [] of [] => [] | x::xs => ...)` $\cong$ `unzip([])`
$\cong$ `case [] of [] => ([],[]) | x::xs...`
$\cong$ `([],[])`
This is not equivalent to `([],["7"])`

Therefore, the claim is not true.

3: Task 4.2

Let W_n be the work of `prefixSum`

$$W_n = k_0 + W_{n-1} + k(n-1)$$

$$W_n = k_0 + k_0 + W_{n-2} + k(n-2) + k(n-1)$$

$$W_n = k_0 + k_0 + k_0 + W_{n-3} + k(n-3) + k(n-2) + k(n-1)$$

.

.

.

$$W_n = n(k_0) + \sum_{i=1}^{n} k(n-i)$$

$$W_n \leq n(k_0) + \sum_{i=1}^{n} k(n)$$

$$W_n \leq n(k_0) + n^2 k$$

Therefore, W_n has a $O(n^2)$ time complexity.

4: Task 4.4

Let W_n be the work of prefixSumFast for a list of length n, $n > 0$.

$$W_n = k_0 + W_{prefixSumHelp(n)}$$

Therefore, the work of prefixSumFast is bounded by the work of prefixSumHelp.
Let W_n be the work of prefixSumHelp for a list of length n, $n > 0$.

$$W_n = k_0 + W_{n-1}$$

This is because the function calls itself exactly once on a sublist of length $n-1$. The other operations which occur on the function are constant time and can be considered as k_0. The closed form for this recurrance is obvious, and has been shown many times before in class:

$$W_n = nk_0$$

Therefore the function is bounded by $O(n)$.

5: Task 5.1

$i \geq 0, \qquad k \geq 0, \qquad k \geq \texttt{length}(\texttt{l}) - i$

6: Task 6.4

If `subset_sum_dc (s,t)` $\cong$ true, then there exists a subset of s which sums to t.
Assume the hypothesis. The only way for `subset_sum_dc (s,t)` to be equivalent to true, is if it returns true. Let's see when the function returns true:

```
fun subset_sum_dc (l : int list, s : int) : bool =
    let val (sumP, yoU) = subset_sum_cert(l,s)
    in
      case sumP of
           false => false
         | true => (case ((sum_list(yoU) = s) andalso (contained(yoU,l))) of
                        false => raise Fail "invalid certificate"
                      | true => true)
    end
```

We can see that the function if the function returns true if and only if the following evaluates to true:

```
(case ((sum_list(yoU) = s) andalso (contained(yoU,l))) of
     false => raise Fail "invalid certificate"
   | true => true)
```

Karan Sikka
ksikka@andrew.cmu.edu

This case statement will only return true if the logical condition:
`(sum_list(yoU) = t) andalso (contained(yoU,s))` is equivalent to true, where yoU is of type `int list` .
This occurs iff `sum_list(yoU)=t` is true and `contained(yoU,l)` is true.
Here we assume that sum_list and contained behaved as described in the task.
Since (contained(yoU,s) $\cong$ true), then according to the spec, yoU is a subset of s.
Since (sum_list(yoU) $\cong =$ t), then according to the spec, yoU sums to t.
Therefore, there exists a subset of s which sums to t. ■