15-150 Assignment 02

Jack Kasbeer
jkasbeer@andrew.cmu.edu
Section M
September 15, 2015

2: Basics

1. `fun pow : (int, int) -> int`,
   `fun cube : real -> real`,
   `fun bop1 : real -> int`,
   `fun bop2 : real -> real`,
   `fun bop3` is not well-typed; `trunc z` will return an `int`, but `cube` expects a `real` as an argument.

2.
```
pow(1, 21 + 21)
=> (fn k,x => if k = 0 then 1 else x * pow (k-1, x) (1, 21 + 21))
=> (fn k,x => if k = 0 then 1 else x * pow (k-1, x) (1, 42))
=> if 1 = 0 then 1 else 42 * pow (1-1, 42)
=> if false then 1 else 42 * pow(1-1, 42)
=> 42 * pow(1-1, 42)
=> 42 * pow(0, 42)
=> 42 * (fn k,x => if k = 0 then 1 else x * pow (k-1, x) (0, 42))
=> 42 * (fn k,x => if true then 1 else x * pow (k-1, x) (0, 42))
=> 42 * 1
=> 42
```

3: Recursive Functions

1. `fact(3)` $=_{int}$ `6`
   $fact(3) =_{int} 3 * fact(3-1)$
   $=_{int} 3 * fact(2)$, by referential transparency
   $=_{int} 3 * 2 * fact(1)$, by (b)
   $=_{int} 3 * 2 * 1 * fact(0)$, by (a)
   $=_{int} 3 * 2 * 1 * 1$, by (1)
   $=_{int} 6$, trivially
   Hence, $fact(3) =_{int} 6$

2. `fact(~1)` $\neq_{int}$ `fact(~2)`
   Both of these expressions fail to terminate since the function `fact` doesn't account for negative arguments. They will execute in an infinite recursive loop and never terminate since the argument `n` will never be equal to `0`.

3. Task 3.3 is in `hw02.sml`

4: Gregorian Functional Programming

1. What is its type?

   (i) `gauss1 : (int, int, int) -> int`

   (ii) `gauss1(2,2,2015) : int`

   (iii) `gauss1(1+1, 3-1, 5*403) : int`

   (iv) `gauss1(1.0, 2.0, 2015.0)` is not well-typed because `gauss1` expects three arguments of type `int` and this expression passes in arguments of type `real`.

2. What are their values?

   (i) `val gauss1 = fn (D:int, M:int, Y:int) : int =`

```
let
val m = if M>2 then M-2 else M+10
val y = if M>2 then Y else Y-1
val c = y div 100
val a = y mod 100
val b = (13 * m - 1) div 5 + (a div 4) + (c div 4)
in
(a + b + D - 2 * c) mod 7
end;
```

   (ii) `val gauss1(2,2,2015) =`

```
(fn (D:int, M:int, Y:int) : int =
let
val m = if M>2 then M-2 else M+10
val y = if M>2 then Y else Y-1
val c = y div 100
val a = y mod 100
val b = (13 * m - 1) div 5 + (a div 4) + (c div 4)
in
(a + b + D - 2 * c) mod 7
end)(2,2,2015)
```

   (iii) `val gauss1(2,2,2015) =`

```
(fn (D:int, M:int, Y:int) : int =
let
val m = if M>2 then M-2 else M+10
val y = if M>2 then Y else Y-1
val c = y div 100
val a = y mod 100
val b = (13 * m - 1) div 5 + (a div 4) + (c div 4)
in
(a + b + D - 2 * c) mod 7
end)(1+1, 3-1, 5*403)
```

(iv) `val gauss1(1.0, 2.0, 2015.0)` does not exist

3. What are their syntactic values?
   (i) Syntactic value of `gauss1 =`

```
fn (D:int, M:int, Y:int) : int =
let
val m = if M>2 then M-2 else M+10
val y = if M>2 then Y else Y-1
val c = y div 100
val a = y mod 100
val b = (13 * m - 1) div 5 + (a div 4) + (c div 4)
in
(a + b + D - 2 * c) mod 7
end;
```

   (ii) Syntactic value of `gauss1(2,2,2015) = 1`

   (iii) Syntactic value of `gauss1(1+1, 3-1, 5*403) = 1`

   (iv) Syntactic value of `gauss1(1.0, 2.0, 2015.0)` does not exist

4. Task 4.4 is in `hw02.sml`

5. Which of these statements are true/false?
   (i) `gauss1` and `gauss2` are extensionally equivalent.. FALSE.

   These functions handle the arguments differently and take different algorithmic approaches to determining the day of the week. It is only for valid dates do they evaluate to the same value. So if we enter an invalid date into `gauss1` and the same invalid date into `gauss2`, they will evaluate to different values; hence, they are not extensionally equivalent.

   (ii) `gauss1 (d, m, y) = gauss2 (d, m, y)` for all integer values `d, m, y`.. FALSE. Same reasoning applies from above, but the main point is these functions compute the day of the week differently so negative integers will yield different answers for both functions.

   (iii) `gauss1 (d, m, y) = gauss2 (d, m, y)` for all integer values `d, m, y` such that `is_valid_date (d, m, y) = true`.. TRUE. If the date is valid, both of these functions are correct implementations so they will yield the same answer.

6. Task 4.6 is in `hw02.sml`

5: Sums of Powers

1. Prove: $\forall n \geq 1.\ \sum_{k=1}^{n} k^4 = (6n^5 + 15n^4 + 10n^3 - n)/30$

Proof. Let $P(n)$ be "$\sum_{k=1}^{n} k^4 = (6n^5 + 15n^4 + 10n^3 - n)/30$".
We will prove "$\forall n \geq 1.P(n)$" by induction on n.
Base case: $P(1)$ holds since,

$$P(1) \Rightarrow \sum_{k=1}^{1} k^4 = 1^4 = 1$$

and

$$\begin{aligned}(6(1^5) + 15(1^4) + 10(1^3) - 1)/30 \\ = (6 + 15 + 10 - 1)/30 \\ = (31 - 1)/30 \\ = 30/30 \\ = 1\end{aligned}$$

Now, suppose $P(m)$ holds for some arbitrary m. WWTS $P(m+1)$ holds in order to prove $P(n).\forall n \geq 1$.

$$P(m+1) \Rightarrow [6(m+1)^5 + 15(m+1)^4 + 10(m+1)^3 - (m+1)]/30$$

$$\begin{aligned}= [(6m^5 + 30m^4 + 60m^3 + 60m^2 + 30m + 6) + (15m^4 + 60m^3 + 90m^2 + 60m + 15) \\ +(10m^3 + 30m^2 + 30m + 10) - (m+1)]/30\end{aligned}$$

$$= \frac{6m^5 + 15m^4 + 10m^3 - m}{30} + \frac{30m^4 + 120m^3 + 180m^2 + 120m + 30}{30}$$

$$\Rightarrow \sum_{k=1}^{m} k^4 + (m^4 + 4m^3 + 6m^2 + 4m + 1) \text{ (induction step)}$$

$$= \sum_{k=1}^{m} k^4 + (m+1)^4$$

$$= \sum_{k=1}^{m+1} k^4$$

$$\therefore \sum_{k=1}^{m+1} k^4 = \frac{6(m+1)^5 + 15(m+1)^4 + 10(m+1)^3 - (m+1)}{30}$$

Hence, by PMI, it follows that $\forall n \geq 1.\ P(n)$ and we're done. □

2. k multiplication operations are performed in `pow(k,n)`
3. 15 multiplication operations are performed in `check(n)`
4. Task 5.4 is in `hw02.sml`