宣言型プログラム論 ミニプロジェクト 1

200911434 青木大祐

平成 24 年 10 月 2 日

以下にミニプロジェクト全体のソースコードを示す。

```
let base_bits = 31
let base_mask = (1 lsl base_bits) - 1

let suint_inc m =
  let x = m + 1 in
  let r = x land base_mask in
  (r, x <> r);;

let suint_add_carry m n c =
  let x = n + m in
  let x = if c then x + 1 else x in
  let r = x land base_mask in
  (r, x <> r);;

let high_mask = base_mask lsl base_bits

let suint_mult m n =
let x = n * m in
let l = x land base_mask in
let h = (x land high_mask) lsr base_bits in
(h, l)

(* 1 *)
let buint_add (bi1 : int list) (bi2 : int list) : int list =
  let myhd l =
    match l with
      | [] -> 0
      | hd::tl -> hd in
  let mytl l =
    match l with
      | [] -> []
      | hd::tl -> tl in
  let rec buint_add_carry bc1 bc2 cr =
    if bc1 = [] && bc2 = [] then
      if cr then [1] else []
    else let (sum, carry) = suint_add_carry (myhd bc1) (myhd bc2) cr in
    sum::(buint_add_carry (mytl bc1) (mytl bc2) carry)
  in
  buint_add_carry bi1 bi2 false;;

(* 2 *)
let buint_fib2 (n:int) :int list * int list =
  let rec fib2_internal f1 f2 ni =
    if n = ni then (f1, f2)
    else
      fib2_internal f2 (buint_add f1 f2) (ni+1)
  in
  fib2_internal [1] [1] 1;;

(* 3 *)
let buint_suint_mult l n =
  let rec mult_internal li ni ui ci =
    match li with
      | hd::tl -> let (upper, current) = suint_mult hd ni in
                  let (sum, carry) = suint_add_carry current ui ci in
                  [sum] @ mult_internal tl ni upper carry
      | _ -> let (s, c) = suint_add_carry 0 ui ci in
             if s = 0 then [] else [s]
  in
  mult_internal l n 0 false;;

(* 4 *)
let buint_fact n =
  let rec fact_internal now ans =
    if now = 0 then
      ans
    else
      fact_internal (now-1) (buint_suint_mult ans now)
  in
  fact_internal n [1];;

(* 5 *)
let buint_mult target multiplier =
  let rec mult_internal target multiplier ans =
    match multiplier with
      | hd::tl -> let ans = if ans = [] then [] else [0] @ ans in
                  mult_internal target tl (buint_add ans (buint_suint_mult target hd))
      | _ -> ans
  in
  mult_internal target (List.rev multiplier) [0];;


(* run 1 - test input
```

```
% perl6 dec_to_n.pl '("1" x 10).Int ** 2' "2**31"
12345679009876543 21 -> 2**31 based
[91750577; 574890478] # arg1

% perl6 dec_to_n.pl '("2" x 10).Int ** 2' "2**31"
4938271603950617284 -> 2**31 based
[367002308; 152078264; 1] # arg2

% perl6 dec_to_n.pl '(("1" x 10).Int ** 2) + (("2" x 10).Int ** 2)' "2**31"
61728395049382716 05 -> 2**31 based
[458752885; 726968742; 1] # expected answer
 *)
buint_add [91750577; 574890478] [367002308; 152078264; 1];;

(* run 2 - test input
% perl6 dec_to_n.pl `perl6 fib.pl 100`
3542248481792619150 75 -> 2**31 based
[1167376323; 1740041551; 76] # expected answer
 % perl6 dec_to_n.pl `perl6 fib.pl 101`
5731478440138170841 01 -> 2**31 based
[277887173; 604790509; 124] # expected answer
 *)
buint_fib2 100;;

(* run 3 - test input
 % perl6 dec_to_n.pl "`perl6 fib.pl 100` * (1 x 10)"
39358316460426603361897089832 5 -> 2**31 based
[444755861; 1144662764; 1592882151; 39] # expected answer
 *)
let (num, _) = buint_fib2 100 in
buint_suint_mult num 1111111111;;

(* run 4 - test input
% perl6 dec_to_n.pl '(sub fact($_) { $_ ?? $_ * fact($_ -1) !! 1}).(25)'
15511210043330985984000000 -> 2**31 based
[2076180480; 1128227104; 3363457] # expected answer
 *)
buint_fact 25;;

(* run 5 - test input
% perl6 dec_to_n.pl '(("1" x 10).Int ** 2) * (("2" x 10).Int ** 2)' "2**31"
60966316085962505719250114311591588 4164 -> 2**31 based
[1069672836; 1984622252; 393424931; 615602474] # expecter answer
 *)
buint_mult  [91750577; 574890478] [367002308; 152078264; 1];;
```

1　多倍長整数の加算

```
(* 1 *)
let buint_add (bi1 : int list) (bi2 : int list) : int list =
  let myhd l =
    match l with
      | [] -> 0
      | hd::tl -> hd in
  let mytl l =
    match l with
      | [] -> []
      | hd::tl -> tl in
  let rec buint_add_carry bc1 bc2 cr =
    if bc1 = [] && bc2 = [] then
      if cr then [1] else []
    else let (sum, carry) = suint_add_carry (myhd bc1) (myhd bc2) cr in
    sum::(buint_add_carry (mytl bc1) (mytl bc2) carry)
  in
  buint_add_carry bi1 bi2 false;;
```

内部関数として、List.hd、List.tl に空リストの場合の処理を追加した myhd、mytl を用いており、各桁の和と carry を計算しながら、再帰的に次の桁へ計算を進めている。

以下に実行部分とその結果を示す。以降、テスト用の数値は別に作成した検算用プログラムで生成したものを用いている。

```
(* run 1 - test input
% perl6 dec_to_n.pl '("1" x 10).Int ** 2' "2**31"
12345679009876543 21 -> 2**31 based
[91750577; 574890478] # arg1

% perl6 dec_to_n.pl '("2" x 10).Int ** 2' "2**31"
4938271603950617284 -> 2**31 based
[367002308; 152078264; 1] # arg2

% perl6 dec_to_n.pl '(("1" x 10).Int ** 2) + (("2" x 10).Int ** 2)' "2**31"
6172839504938271605 -> 2**31 based
[458752885; 726968742; 1] # expected answer
 *)
buint_add [91750577; 574890478] [367002308; 152078264; 1];;
```

```
 - : int list = [458752885; 726968742; 1]
```

正しく計算できていることが分かる。

2　n 番目と n+1 番目のフィボナッチ数の組を多倍長整数として計算する関数

```
(* 2 *)
let buint_fib2 (n:int) :int list * int list =
  let rec fib2_internal f1 f2 ni =
    if n = ni then (f1, f2)
    else
      fib2_internal f2 (buint_add f1 f2) (ni+1)
  in
  fib2_internal [1] [1] 1;;
```

ni を 1 から与えられた n まで増やしながら、課題 1 で作成した buint_add 関数を用いて、フィボナッチ数を計算している。

以下に実行部分とその結果を示す。

```
(* run 2 - test input
% perl6 dec_to_n.pl `perl6 fib.pl 100`
354224848179261915075 -> 2**31 based
[1167376323; 1740041551; 76] # expected answer
 % perl6 dec_to_n.pl `perl6 fib.pl 101`
573147844013817084101 -> 2**31 based
[277887173; 604790509; 124] # expected answer
 *)
buint_fib2 100;;
```

```
- : int list * int list =
([1167376323; 1740041551; 76], [277887173; 604790509; 124])
```

正しく計算できていることが分かる。

3 多倍長整数と整数 (0 以上 $2^{31} - 1$ 以下) の乗算

```
(* 3 *)
let buint_suint_mult l n =
  let rec mult_internal li ni ui ci =
    match li with
      | hd::tl -> let (upper, current) = suint_mult hd ni in
                  let (sum, carry) = suint_add_carry current ui ci in
                  [sum] @ mult_internal tl ni upper carry
      | _ -> let (s, c) = suint_add_carry 0 ui ci in
             if s = 0 then [] else [s]
  in
  mult_internal l n 0 false;;
```

各桁に整数を掛けて下位からの繰り上がりを加算した後で、上位への繰り上がりを算出し、再帰的に桁を辿っていく。

以下に実行部分とその結果を示す。

```
(* run 3 - test input
 % perl6 dec_to_n.pl "`perl6 fib.pl 100` * (1 x 10)"
39358316460426603361897089832 5 -> 2**31 based
[444755861; 1144662764; 1592882151; 39] # expected answer
 *)
let (num, _) = buint_fib2 100 in
buint_suint_mult num 1111111111;;
```

```
- : int list = [444755861; 1144662764; 1592882151; 39]
```

正しく計算できていることが分かる。

4 一桁分の整数 n に対して n! を多倍長整数として計算する関数

```
(* 4 *)
let buint_fact n =
  let rec fact_internal now ans =
    if now = 0 then
      ans
    else
      fact_internal (now-1) (buint_suint_mult ans now)
  in
  fact_internal n [1];;
```

課題 3 で作成した buint_suint_multi 関数を用いて、now を 0 まで減らしながら再帰的に乗算していく。

以下に実行部分とその結果を示す。

```
(* run 4 - test input
% perl6 dec_to_n.pl '(sub fact($_) { $_ ?? $_ * fact($_ -1) !! 1}).(25)'
155112100433309859840000000 -> 2**31 based
[2076180480; 1128227104; 3363457] # expected answer
 *)
buint_fact 25;;
```

```
- : int list = [2076180480; 1128227104; 3363457]
```

正しく計算できていることが分かる。

5 多倍長整数の乗算

```
(* 5 *)
let buint_mult target multiplier =
  let rec mult_internal target multiplier ans =
    match multiplier with
      | hd::tl -> let ans = if ans = [] then [] else [0] @ ans in
                  mult_internal target tl (buint_add ans (buint_suint_mult target hd))
      | _ -> ans
  in
  mult_internal target (List.rev multiplier) [0];;
```

同じく課題 3 で作成した buint_suint_multi 関数を用いて計算を行う。計算を再帰に適した形にするため、今までは下位の桁から計算していたところを上位の桁から計算している。

以下に実行部分とその結果を示す。

```
(* run 5 - test input
% perl6 dec_to_n.pl '(("1" x 10).Int ** 2) * (("2" x 10).Int ** 2)' "2**31"
60966316085962505719250114311159884164 -> 2**31 based
[1069672836; 1984622252; 393424931; 615602474] # expecter answer
 *)
buint_mult  [91750577; 574890478] [367002308; 152078264; 1];;
```

```
- : int list = [1069672836; 1984622252; 393424931; 615602474]
```

正しく計算できていることが分かる。