Vers le Calcul Formel

Cours d'Informatique de

l'École Centrale d'Électronique

Ingénieur Deuxième Année

Sébastien Veigneau

Janvier 1998

INTRODUCTION

Un exemple en arithmétique

- **Problème** : calculer la limite de la suite :

$$\begin{cases} x_0 = 2 & , \\ x_1 = -4 & , \\ x_n = 111 - \frac{1130}{x_{n-1}} + \frac{3000}{x_{n-1}x_{n-2}} & . \end{cases}$$

- Pourquoi ne pas utiliser Caml-Light ?

```
#let rec f = function
  0 -> 2.
| 1 -> -4.
| n -> let fn_1 = f (n-1) and
          fn_2 = f (n-2) in
        111. -. 1130. /. fn_1 +.
                3000. /. (fn_1 *. fn_2);;
f : int -> float = <fun>
```

Un exemple en arithmétique

- Calcul de la limite :

```
#f 10;;
- : float = 6.27443866273
#f 20;;
- : float = 99.8985692662
#f 25;;
- : float = 99.9999999193
```

- **Problème** : la limite est 6 et non pas 100.
- **Raison** : précision limitée des nombres réels.

remplacer le calcul réel par du calcul réel

Vers la solution...

- Développer un système de manipulation de grands entiers ;
- Différer et limiter les calculs réels.

- Pourquoi ne pas utiliser Caml-Light ?

```
#type BigNum = N of int list;;
Type BigNum defined.

#let IsBigNumNull bn = bn=(N []);;
IsBigNumNull : BigNum -> bool = <fun>

#let string_of_BigNum (N v) =
  string_of_BigNum_rec v where rec
  string_of_BigNum_rec = function
      [] -> "0"
  |  [d] -> string_of_int d
  | d::l -> (string_of_BigNum_rec l)^
            (string_of_int d);;
```

```
string_of_BigNum : BigNum -> string = <fun>

#let print_BigNum bn = print_string
  ((string_of_BigNum bn)^"\n");;
print_BigNum : BigNum -> unit = <fun>

#install_printer "print_BigNum";;
- : unit = ()

#let IncBigNum (N v) = N (IncBigNum_list 0 v)
  where rec IncBigNum_list pos = function
      [] -> [1]
  | 9::l -> 0::(IncBigNum_list 1 l)
  | d::l -> (d+1)::l;;
IncBigNum : BigNum -> BigNum = <fun>

#let NormalBigNum (N v) =
  N (rev (NormalBigNum_rec (rev v)))
  where rec NormalBigNum_rec = function
      [] -> []
  | 0::l -> NormalBigNum_rec l
  | d::l -> d::l;;
NormalBigNum : BigNum -> BigNum = <fun>

#let DecBigNum (N v) =
```

```
  NormalBigNum (N (DecBigNum_list v))
  where rec DecBigNum_list = function
      [] -> []
  | 0::l -> 9::(DecBigNum_list l)
  | d::l -> (d-1)::l;;
DecBigNum : BigNum -> BigNum = <fun>

#let rec AddBigNum bn1 = function
  N [] -> bn1
|  bn2 -> IncBigNum
            (AddBigNum bn1 (DecBigNum bn2));;
AddBigNum : BigNum -> BigNum -> BigNum = <fun>

#let rec MinusBigNum bn1 = function
  N [] -> bn1
|  bn2 -> DecBigNum
            (MinusBigNum bn1 (DecBigNum bn2));;
MinusBigNum : BigNum -> BigNum -> BigNum = <fun>

#let rec MultBigNum bn1 = function
   N [] -> N []
| N [1] -> bn1
|   bn2 -> AddBigNum (MultBigNum
                     bn1 (DecBigNum bn2)) bn1;;
MultBigNum : BigNum -> BigNum -> BigNum = <fun>
```

```
#let DivBigNum bn1 bn2 =
  DecBigNum (DivBigNum_rec bn1 bn2)
  where rec DivBigNum_rec = fun
     _ (N []) -> raise (Failure "Division by 0")
|    (N []) _ -> N []
| bn1 (N [1]) -> bn1
|     bn1 bn2 -> IncBigNum (DivBigNum_rec
                     (MinusBigNum bn1 bn2) bn2);;
DivBigNum : BigNum -> BigNum -> BigNum = <fun>

#let a = N [3;6;1;6];;
a : BigNum = 6163
#let b = N [1;4;0;0;3;2;5;1;7;8;4;3;2;7;3;9;1;4;
            0;0;3;2;3;4;5;3;4;5;6];;
b : BigNum = 65435432300419372348715230041

#AddBigNum b a;;
- : BigNum = 65435432300419372348715236204
```

La solution en MAPLE

- Voici la solution en MAPLE, logiciel de calcul formel :

```
> f := proc(n) options remember;
    if n = 0 then 2 elif n = 1 then -4
    else 111-1130*1/f(n-1)+3000*1/f(n-1)/f(n-2)
    fi
end:

> f(30);
          990176025870222717970867
          ------------------------
          164874117215934539909207

> evalf(");;
              6.005648689
> f(50);

  36368216593815882102779587493871769920 67
  ----------------------------------------
   606122140256603032959120809808954954407
```

```
> evalf(");;
                 6.000146535
> f(500);

53559646949233706641975483846843957352424298972
62354934243827186151836692692562216281858012019
86175074103354749616040409486622001024904951489
58833275218333753196804894176078139076773127350
75937717668829794304528542176773875980435787773
96825841525120651345499511721516577295477515950
35703710983567000165908871628208142105539072670
06858569155273986864752941344644698370014083377
09734187069067
-----------------------------------------------
89266078248722844403292473078073262254035406727
10008289626036647020858884870972126679667965589
12064891684236388917988674849339047789707998678
36596870874572143402478401247123091845063323727
59743214509595587085252094924283861830804332339
81101258076364760777271463036910086886379544538
89306126230808181578035705710147567684836435961
10917639412659505354861228752075540683385526126
5805917571407
```

```
> evalf(", 40);

  6.000000000000000000000000000000000000000
```

Première caractéristique

des systèmes de calcul formel

Arithmétique entière exacte

Affaire à suivre...