Guy Laffitte, suite de propos de banc de touche

Sans les bonnes odeurs de la Biscuiterie Nantaise !

Quant à celles de LU, elles ont quitté le quartier de la Cité des Congrès !


BN
BIENVENUE
WELCOME
pladis

```
Programme SYRACUSE(x : NAT1)
WHILE x > 1 DO
  IF pair(x) THEN
    x := x / 2
  ELSE
    x := 3 * x + 1
  ENDIF
ENDWHILE
```

Un entier positif est syracusain ssi la boucle se termine.
Problème non résolu : tout entier positif est-il syracusain ?
Conseil : partir de 27.

Boucle contenant un corps non-déterministe

Programme HYBRIDE(x : INTEGER)

```
WHILE x /= 0 DO
  IF x > 0 THEN
    x := x - 1
  ELSE
    x :: { y | y : INTEGER & y > x }
  ENDIF
INVARIANT
  x : INTEGER
VARIANT
  ?
ENDWHILE
```

Note 1 : INTEGER désigne les entiers relatifs (Z en maths).
Note 2 : Le symbole "::" désigne l'affectation non-déterministe d'un élément de l'ensemble à sa droite vers la variable à sa gauche.

Avec un choix judicieux du variant, la boucle se termine toujours.
Par contre, si la valeur de départ est négative, on ne peut pas borner le nombre d'itérations en fonction de la valeur initiale.

```
MODEL

   fifo_1 (Max, XX)

CONSTRAINTS

   Max : NAT &
   Max >= 2 &
   XX /= {}

VARIABLES

   in1, out1, tab1

INVARIANT

   in1 : NATURAL1 &
   out1 : NATURAL1 &
   out1 <= in1 &
   in1 <= out1 + Max &
   tab1 : (1 .. (in1 - 1)) --> XX
```

INITIALISATION

```
in1 := 1 ||
out1 := 1 ||
tab1 := {}
```

OPERATIONS

```
ll <-- long =
BEGIN
  ll := in1 - out1
END ;

add ( xx ) =
PRE
  xx : XX &
  in1 + 1 <= out1 + Max
THEN
  tab1(in1) := xx ||
  in1 := in1 + 1
END ;

xx <-- rmv =
PRE
  out1 + 1 <= in1
THEN
  xx := tab1(out1) ||
  out1 := out1 + 1
END

END
```

```
REFINEMENT
    fifo_2 (Max, XX)
REFINES
    fifo_1
VARIABLES
    in2, out2, tab2
INVARIANT
    in2 = in1 &
    out2 = out1 &
    tab2 : (0 .. Max - 1) --> XX &
    ! adr . (adr : (out1 .. in1 - 1) =>
              tab1(adr) = tab2(adr mod Max))
```

```
INITIALISATION
    in2 := 1 ||
    out2 := 1 ||
    tab2 :: (0 .. Max - 1) --> XX

  xx <-- rmv =
  BEGIN
     xx := tab2(out2
mod Max) ||
     out2 := out2 + 1
  END
END
```

```
OPERATIONS
   ll <-- long =
   BEGIN
      ll := in2 -
out2
   END ;
   add ( xx ) =
   BEGIN
      tab2(in2 mod
Max) := xx ||
      in2 := in2 + 1
   END ;
```

```
REFINEMENT

    fifo_3 (Max, XX)

REFINES

    fifo_2

VARIABLES

    in3, out3, tab3

INVARIANT

    out3 = out2 mod Max &
    in3 = out3 + in2 - out2
&
    tab3 = tab2
```

```
INITIALISATION
    in3 := 1 ;
    out3 := 1 ;
    tab3 :: (0 .. Max - 1) -->
XX
OPERATIONS
    ll <-- long =
    BEGIN
        ll := in3 - out3
    END ;
```

```
add ( xx ) =
   BEGIN
      IF in3 < Max THEN
         tab3(in3) := xx
      ELSE
         tab3(in3 - Max)
:= xx
      END ;
      in3 := in3 + 1
   END ;
```

```
xx <-- rmv =
   BEGIN
      xx := tab3(out3) ;
      out3 := out3 + 1 ;
      IF out3 = Max THEN
         in3 := in3 - Max ;
         out3 := out3 - Max
      END
   END
END
```

Une ménagère possède une pile de torchons dans un placards.
Les torchons sont tous identiques.
En fonction de ses besoins, la ménagère prend un ou plusieurs torchons sur le dessus de la pile.
De temps à autre, elle fait sa lessive. Les torchons lavés, séchés puis pliés sont mis en une pile qu'elle range EN DESSOUS de la pile des torchons non utilisés.
Cette manière de faire semble clairement meilleure que de les ranger par dessus afin répartir au mieux l'usure des torchons.
Que pouvez-vous en dire ?

Indications :
- l'approche statistique semble vouée à l'échec dès le départ au vu des problèmes de combinatoire,
- une approche de type "invariant" permet d'arriver à un résultat non trivial.

REFINEMENT

fifo_3 (Max, XX)

REFINES

fifo_2

VARIABLES

in3, out3, tab3

INVARIANT

out3 = out2 mod Max &

in3 = out3 + in2 - out2 &

tab3 = tab2

INITIALISATION

in3 := 1 ;
out3 := 1 ;
tab3 :: (0 .. Max - 1) --> XX

OPERATIONS

```
ll <-- long =
BEGIN
  ll := in3 - out3
END ;

add ( xx ) =
BEGIN
  IF in3 < Max THEN
    tab3(in3) := xx
  ELSE
    tab3(in3 - Max) := xx
  END ;
  in3 := in3 + 1
END ;
```

```
xx <-- rmv =
BEGIN
   xx := tab3(out3) ;
   out3 := out3 + 1 ;
   IF out3 = Max THEN
      in3 := in3 - Max ;
      out3 := out3 - Max
   END
END
```