TD 1 : Modélisation avec la Méthode B

Preuves et Constructions Formelles

J. Christian Attiogbé

Université de Nantes, November 2020

Exercice 1 - pair, impair, premier

```
MACHINE
    OddEvenPrime
DEFINITIONS
    divides(nn,dd) == (#kk . (kk : NATURAL & (nn = kk * dd)))
    ; odd(nn) == not divides(nn,2)
    ; even(nn) == divides(nn,2)
    ; foo(xx, yy) == ( (xx <= yy) or (yy <= xx) )

CONSTANTS
    divisible
,   prime
,   pair // even
,   impair // odd
PROPERTIES /
   ...
ASSERTIONS
   ...
END
```

Exercice 1 (voici les propriétés)

```
PROPERTIES
  divisible : NATURAL * NATURAL --> BOOL
& !(nn,dd).((nn : NATURAL & dd : NATURAL & dd > 0)
    => (divisible(nn,dd) = bool(#kk . (kk : NATURAL &
                                        (nn = kk * dd)))))
& prime : NATURAL --> BOOL // a prime number
& !(nn).((nn : NATURAL) => ((prime(nn) =
        bool(!di.( (di : NATURAL & (divides (nn,di)))
           => ( (di = 1) or ((di = nn) & (di /=1 ))) )) )))
& pair : NATURAL --> BOOL
& !nn.((nn: NATURAL) =>
    (pair(nn) = bool(#kk . (kk : NATURAL & (nn = kk * 2)))))
& impair : NATURAL --> BOOL
& !nn.((nn: NATURAL) =>
    (impair(nn) = bool(not(#kk . (kk : NATURAL &
                                      (nn = kk * 2))))))
```


Exercice 1 (et voici nos assertions)

```
ASSERTIONS
    !nn.((nn : NATURAL) => ( even(nn) or odd(nn) ))
                              // tout entier est pair ou impair

;   !(nn,mm).((nn : NATURAL & mm: NATURAL) => foo(nn,mm))
// une autre façon
/*--------------------------------
!nn.((nn: NATURAL & (#kk . (kk : NATURAL &
                            (nn = kk * 2)))) => pair(nn) = TRUE)
;   !nn.((nn: NATURAL) =>
        (pair(nn) = bool(#kk . (kk : NATURAL & (nn = kk * 2)))))

;   !(nn).(((nn : NATURAL) &
             (!di.( (di : NATURAL & (divides (nn,di))) =>
                                    ( (di = 1) or (di = nn)))))
      => ((prime(nn) = TRUE))) -------------------------*/
```

Exercice 2 - Construire le carré (la machine abstraite)

```
MACHINE
    carre
OPERATIONS
res <-- carreN (nn) =
PRE nn : NAT
    & nn >= 1
    & nn < MAXINT
    &  nn*nn <= MAXINT
THEN
    ANY rr WHERE // substitution non-déterministe
        rr : NAT
    &   nn*nn <= MAXINT
    &   rr = nn*nn
    THEN
        res := rr
    END
END
```


Exercice 1 - carré (un raffinement)

```
IMPLEMENTATION carre_i
REFINES carre
OPERATIONS
res <-- carreN (nn) =
BEGIN
  VAR tmp, ii  IN
     ii := 0 ;  tmp := 0 ;
     WHILE (ii+1 < nn)
     DO   tmp := tmp + nn ;  ii := ii +1
     INVARIANT
        ii : NAT & ii <= MAXINT   & nn : NAT  & nn < MAXINT
      & ii < nn  & nn+nn <= MAXINT & tmp : NAT  & tmp <= MAXINT
      & tmp + nn <=  nn*nn  & tmp+nn <= MAXINT  & tmp = nn*(ii)
     VARIANT  nn - ii
     END;
     res := tmp + nn
     END
END END
```

Exercice 2 - carré des relatifs

```
res <-- carreZ (vv) = /* carre de Z */
PRE
    vv : INTEGER
    & vv < MAXINT
    & vv*vv <= MAXINT
    THEN
    ANY rr WHERE
    rr : NATURAL
    & rr = vv*vv
THEN
    res := rr
END
END
```


Exercice 3 - Calculatrice 8 bits

Spécification en B d'une machine à calculer rudimentaire qui manipule des valeurs entières stockables sur 8 bits.

- La machine à calculer dispose de deux registres : un registre principal nommé RP et un registre secondaire nommé RS.
- La machine propose les opérations suivantes : storeRP(val) ; storeRS(val) ; incRP1 ; decRP1 ; incrRS ; decrRS cmp ; getRP

Analyse : 8 bits ? contraintes ? Espace d'états ? quelles variables ? Les fonctionnalités ?

```
MACHINE Calc8B
  VARIABLES . . .
  INVARIANT . . .
  INITIALISATION . . .
  OPERATIONS . . .
END
```

Exercice 3 - Calculatrice 8 bits

```
VARIABLES
   rp // registre principal
,  rs // registre secondaire

INVARIANT
   rp : 0..255 // ou bien   rp : NAT & 0 <= rp & rp <= 255
&  rs : 0..255
INITIALISATION
   rp :=  ???
|| rs :=  ???
```


Exercice 3 - Calculatrice 8 bits

```
OPERATIONS
    storeRP(val) =  // stockage de val dans le registre rp
    PRE
       val : ???
    &  ... ?
    THEN
       rp := rp + val
    END
;
    incrRP1 =  // incrementer le registre rp de 1
    PRE
        ...
    THEN
        rp := rp + 1
    END
```

Exercice 3 - Calculatrice 8 bits

```
// OPERATIONS c'est la suite

    vr <-- getRP = // récupération dans vr, de la valeur de RP
    BEGIN
       vr := rp
    END
;
    vr <-- cmp =
     BEGIN
        vr := bool (rp = rs)  // noter bien ceci !!!
     END
END // de la MACHINE
```