Trabalho 4

Grupo 24

Pedro Faria - A72640

Hugo Costeira - A87976

1

```
assume m >= 0 and n >= 0 and r == 0 and x == m and y == n
    0: while y > 0:
    1:    if y & 1 == 1:
                y , r  = y-1 , r+x
    2:    x , y = x<<1  ,  y>>1
    3: assert r == m * n


invar = 0<=y<=n and  m * n = x * y + r

assume m >= 0 and n >= 0 and r == 0 and x == m and y == n; skip; 0<=n<=y and
m*n==x*y+r

assume y>0 and 0<=n<=y and  m * n == x * y + r;
    if y & 1 == 1 then y, r = y-1, r+x ;
    x , y = x<<1 , y>>1;
    0<=n<=y and  m * n = x * y + r

assume y<=0 and 0<=n<=y and  m * n == x * y + r; skip; assert r == m * n
```

In [ ]:
```
!pip install z3-solver
```

In [ ]:
```
from z3 import *

def prove1(f):
    s = Solver()
    s.add(Not(f))
    r = s.check()
    if r == unsat:
        print("Proved")
    else:
        print("Failed to prove")
        m = s.model()
        for v in m:
            print(v,'=', m[v])
```

In [ ]:

```
m, n, r, x, y  = Ints('m n r x y')

prove1(Implies(And(y<=0,0<=y,y<=n,m*n==x*y+r),r==m*n))
```

```
Proved
```

```
inicialização ciclo
assume m >= 0 and n >= 0 and r == 0 and x == m and y == n; skip; 0<=n<=y and
m*n==x*y+r
m >= 0 and n >= 0 -> 0<=n<=y and  m * n == x * y + r [y/n][x/m][r/0]
m >= 0 and n >= 0 -> 0<=n<=n and  m * n == m * n + 0

pos = r == m * n
ciclo = (assume m >= 0 and n >= 0 and r == 0 and x == m and y == n; skip;
assert 0<=n<=y
and  m * n = x * y + r) and (assume y>0 and 0<=n<=y and  m * n == x * y + r;
if y & 1 == 1 then y, r = y-1, r+x ; x , y = x<<1  ,  y>>1;  0<=n<=y
and  m * n = x * y + r) and (assume y<=0 and 0<=n<=y and
m * n == x * y + r; skip; assert r == m * n)

[while y>0 : if y & 1 == 1 then y , r  = y-1 , r+x ; assert pos; x , y =
x<<1 , y>>1;
assert pos;]

[ciclo; (assume  y & 1 == 1;  y , r  = y-1 , r+x; || assume ~y | 1 != 1;);
assert pos;
x , y = x<<1   ,  y>>1 ; assert pos]

[ciclo; (assume  y & 1 == 1;  y , r  = y-1 , r+x; x , y = x<<1,  y>>1 ;
assert pos; ||
assume ~y | 1 != 1; assert pos); x , y = x<<1   ,  y>>1 ; assert pos]

[ciclo; [y & 1 == 1 -> [ y , r  = y-1 , r+x; assert pos;]] or [~y | 1 != 1;
assert pos];
x , y = x<<1   ,  y>>1 ; assert pos;]

[ciclo; [y & 1 == 1 -> [ y , r  = y-1 , r+x; assert pos;]] or [~y | 1 != 1;
assert pos];
x , y = x<<1   ,  y>>1 ; assert pos;]

[ciclo; [y & 1 == 1 -> [ y , r  = y-1 , r+x; assert pos;]] or [~y | 1 != 1
assert pos];
x , y = x<<1   ,  y>>1 ; assert pos;]

[ciclo; y & 1 == 1 -> [[ y , r  = y-1 , r+x; assert pos;]] or [~y | 1 != 1;
assert pos];
x , y = x<<1   ,  y>>1 ; assert pos;]

[ciclo; y & 1 == 1 -> [[y=y-1; r=r+x; assert pos;]] or [~y | 1 != 1; assert
pos];
x , y = x<<1   ,  y>>1 ; assert pos]
```

```
[ciclo; y & 1 == 1 -> pos[y/(y>>1)][r/r+(x<<1)] or [~y | 1 != 1-> pos];
pos[x/x<<1][y/y<<2];]
```

In [ ]:

```
m, n, r, x, y = BitVecs("m n r x y", 16)

pre = And(m >= 0, n >= 0, r == 0, x == m, y == n)
pos = (r == m * n)
inv = And(0<=y,y<=n,m*n==x*y+r)

ifT=Implies(y & 1 == 1, substitute(substitute(substitute(inv, (x, x<<1)), (y,
ifF=Implies(Not(y & 1 == 1),substitute(substitute(inv, (x, x<<1)), (y, y>>1)))

ciclo = ForAll([x, y, r], Implies(And(y>0, inv), Or(ifT, ifF)))
final = Implies(And(Not(y>0), inv), pos)

prove(Implies(pre, And(inv, ciclo, final)))
```

```
proved
```