Problema1

January 17, 2021

1 Problema 1

Grupo 8 - Anabela Pereira - A87990 - André Gonçalves - A87942

$$\frac{\{\phi\}\mathsf{skip}\{\theta\} \quad \{\theta \wedge b\}C\{\theta\} \quad \{\theta \wedge \neg b\}\mathsf{skip}\{\psi\}}{\{\phi\}\mathsf{while}\ b\ \mathsf{do}\ C\{\psi\}}$$

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

```
inv = And(x*y+r==m*n,m>=0,n>=0,n>=y,y>=0)
```

1.0.1 a.

```
{ m >= 0 and n >= 0 and r == 0 and x == m and y == n }
0: while y > 0:
1:     if y & 1 == 1:
2:         y , r  = y-1 , r+x
3:     x , y = x<<1  ,  y>>1
4: stop
```

O estado inicial é caracterizado pelo seguinte predicado:

$$pc = 0 \land r = 0 \land m \geq 0 \land n \geq 0 \land x = m \land y = n$$

As transições possíveis no FOTS são caracterizadas pelo seguinte predicado:

$$\begin{array}{c}
(pc = 0 \land y > 0 \land pc' = 1 \land x' = x \land y' = y \land r' = r) \\
\lor \\
(pc = 0 \land y <= 0 \land pc' = 4 \land x' = x \land y' = y \land r' = r) \\
\lor \\
(pc = 1 \land y \ \& \ 1 = 1 \land pc' = 2 \land x' = x \land y' = y \land r' = r) \\
\lor \\
(pc = 1 \land y \ \& \ 1 \neq 1 \land pc' = 3 \land x' = x \land y' = y \land r' = r) \\
\lor \\
(pc = 2 \land pc' = 3 \land x' = x \land y' = y - 1 \land r' = r + x) \\
\lor \\
(pc = 3 \land pc' = 0 \land x' = x << 1 \land y' = y >> 1 \land r' = r) \\
\lor \\
(pc = 4 \land pc' = 4 \land x' = x \land y' = y \land r' = r)
\end{array}$$

[1]:

```
from z3 import *

def bmc_always(declare,init,trans,inv,K):
    for k in range(1,K+1):
        s = Solver()
        state = [declare(i) for i in range(k)]
        s.add(init(state[0]))

        for i in range(k-1):
            s.add(trans(state[i],state[i+1]))

        s.add(Not(inv(state[k-1])))
        if s.check() == sat:
            m = s.model()
            for i in range(k):
                print(i)
                for x in state[i]:
                    print(x,'=',m[state[i][x]])

            return
    print ("Property is valid up to traces of length "+str(K))

def declare(i):
    state = {}
    s = BitVecSort(16)
    state['pc'] = Int('pc'+str(i))
    state['x'] = Const('x'+str(i),s)
    state['y'] = Const('y'+str(i),s)
    state['r'] = Const('r'+str(i),s)
    state['m'] = Const('m'+str(i),s)
    state['n'] = Const('n'+str(i),s)
    return state

def init(s):
    return And(s['pc'] == 0, s['m']>=0, s['n']>=0,s['r']==0,s['x']==s['m'],s['y']==s['n'])

def trans(c,p):
    t01 = And(c['pc'] == 0,c['y']>0,p['pc']==1,p['x']==c['x'],p['y']==c['y'],p['r']==c['r'])
    t04 = And(c['pc'] == 0,c['y']<=0,p['pc']==4,p['x']==c['x'],p['y']==c['y'],p['r']==c['r'])
    t12 = And(c['pc'] == 1,c['y'] & 1 == 1,p['pc']==2,p['x']==c['x'],p['y']==c['y'],p['r']==c['r'])
    t13 = And(c['pc'] == 1,c['y'] & 1 != 1,p['pc']==3,p['x']==c['x'],p['y']==c['y'],p['r']==c['r'])
    t23 = And(c['pc'] == 2,p['pc']==3,p['x']==c['x'],p['y']==c['y'] -1 ,p['r']==c['r'] + c['x'])

    t30 = And(c['pc'] == 3,p['pc']==0,p['x']==c['x'] << 1,p['y']==c['y'] >> 1,p['r']==c['r'])
    t44 = And(c['pc'] == 4,p['pc']==4,p['x']==c['x'],p['y']==c['y'],p['r']==c['r'])
    return And(Or(t01,t04,t12,t13,t23,t30,t44),p['m']==c['m'],p['n']==c['n'])

def inv(s):
    return Implies(s['pc']==1,And(s['x']*s['y']+s['r']==s['m']*s['n'],s['m']>=0,s['n']>=s['y'],s['y']>=0))

def termina(s):
    return s['pc']==4

bmc_always(declare,init,trans,inv,4)
```

```
Property is valid up to traces of length 4
```

[2]:

```
def gera_traco(declare,init,trans,k):
    s = Solver()
    state = [declare(i) for i in range(k)]
    s.add(init(state[0]))

    for i in range(k-1):
        s.add(trans(state[i],state[i+1]))

    if s.check() == sat:
        m = s.model()
        for i in range(k):
            print(i)
            for x in state[i]:
                print(x,'=',m[state[i][x]])




gera_traco(declare,init,trans,33)
```

```
0
pc = 0
x = 2050
```

```
y = 255
r = 0
m = 2050
n = 255
1
pc = 1
x = 2050
y = 255
r = 0
m = 2050
n = 255
2
pc = 2
x = 2050
y = 255
r = 0
m = 2050
n = 255
3
pc = 3
x = 2050
y = 254
r = 2050
m = 2050
n = 255
4
pc = 0
x = 4100
y = 127
r = 2050
m = 2050
n = 255
5
pc = 1
```

```
x = 4100
y = 127
r = 2050
m = 2050
n = 255
6
pc = 2
x = 4100
y = 127
r = 2050
m = 2050
n = 255
7
pc = 3
x = 4100
y = 126
r = 6150
m = 2050
n = 255
8
pc = 0
x = 8200
y = 63
r = 6150
m = 2050
n = 255
9
pc = 1
x = 8200
y = 63
r = 6150
m = 2050
n = 255
10
```

```
pc = 2
x = 8200
y = 63
r = 6150
m = 2050
n = 255
11
pc = 3
x = 8200
y = 62
r = 14350
m = 2050
n = 255
12
pc = 0
x = 16400
y = 31
r = 14350
m = 2050
n = 255
13
pc = 1
x = 16400
y = 31
r = 14350
m = 2050
n = 255
14
pc = 2
x = 16400
y = 31
r = 14350
m = 2050
n = 255
```

```
15
pc = 3
x = 16400
y = 30
r = 30750
m = 2050
n = 255
16
pc = 0
x = 32800
y = 15
r = 30750
m = 2050
n = 255
17
pc = 1
x = 32800
y = 15
r = 30750
m = 2050
n = 255
18
pc = 2
x = 32800
y = 15
r = 30750
m = 2050
n = 255
19
pc = 3
x = 32800
y = 14
r = 63550
m = 2050
```

```
n = 255
20
pc = 0
x = 64
y = 7
r = 63550
m = 2050
n = 255
21
pc = 1
x = 64
y = 7
r = 63550
m = 2050
n = 255
22
pc = 2
x = 64
y = 7
r = 63550
m = 2050
n = 255
23
pc = 3
x = 64
y = 6
r = 63614
m = 2050
n = 255
24
pc = 0
x = 128
y = 3
r = 63614
```

```
m = 2050
n = 255
25
pc = 1
x = 128
y = 3
r = 63614
m = 2050
n = 255
26
pc = 2
x = 128
y = 3
r = 63614
m = 2050
n = 255
27
pc = 3
x = 128
y = 2
r = 63742
m = 2050
n = 255
28
pc = 0
x = 256
y = 1
r = 63742
m = 2050
n = 255
29
pc = 1
x = 256
y = 1
```

```
r = 63742
m = 2050
n = 255
30
pc = 2
x = 256
y = 1
r = 63742
m = 2050
n = 255
31
pc = 3
x = 256
y = 0
r = 63998
m = 2050
n = 255
32
pc = 0
x = 512
y = 0
r = 63998
m = 2050
n = 255
```

1.0.2 b.

[227]:

```
def prove(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])
```

WPC

1ª regra

```
[assume m >= 0 and n >= 0 and r == 0 and x == m and y == n; skip; assert inv]
(m >= 0 and n >= 0 and r == 0 and x == m and y == n) -> [assert inv]
(m >= 0 and n >= 0 and r == 0 and x == m and y == n) -> (inv)
```

2ª regra

```
pre = inv and y>0

[assume pre; if y & 1 == 1 then (y = y-1; r = r + x;) x = x << 1; y = y >> 1; assert inv]
pre -> [if y & 1 == 1 then (y = y-1; r = r + x;) x = x << 1; y = y >> 1; assert inv]
pre -> [((assume y & 1 == 1; y = y-1; r = r + x; assert inv) || (assume y & 1 != 1;)) x = x << 1; y = y >> 1; assert inv]
pre -> [(assume y & 1 == 1; y = y-1; r = r + x; x = x << 1; y = y >> 1; assert inv) || (assume y & 1 != 1; x = x << 1; y = y >> 1; assert inv)]
pre -> [(assume y & 1 == 1; y = y-1; r = r + x; x = x << 1; y = y >> 1; assert inv) || (assume y & 1 != 1; x = x << 1; y = y >> 1; assert inv)]
pre -> [(assume y & 1 == 1; y = y-1; r = r + x; x = x << 1; y = y >> 1; assert inv)] and [(assume y & 1 != 1; x = x << 1; y = y >> 1; assert inv)]
pre -> (y & 1 == 1 ->  [y = y-1; r = r + x; x = x << 1; y = y >> 1; assert inv]) and (y & 1 != 1 -> [x = x << 1; y = y >> 1; assert inv])
pre -> (y & 1 == 1 ->  [assert inv][y-1/y][r+x/r][x<<1/x][y>>1/y]) and (y & 1 != 1 -> [assert inv][x<<1/x][y>>1/y])
pre -> (y & 1 == 1 ->  inv[y-1/y][r+x/r][x<<1/x][y>>1/y]) and (y & 1 != 1 -> inv[x<<1/x][y>>1/y])
```

3ª regra

```
[inv and y<=0; skip; assert r == m*n]
(inv and y<=0) -> [assert r == m*n]
(inv and y<=0) -> (r == m*n)
```

[28]:

```
# WPC
N = 16
s = BitVecSort(N)
m,n,r,x,y = Consts("m n r x y",s)
```

```
inv= And(x*y+r==m*n,m>=0,n>=0,n>=y,y>=0)
# 1 regra
prove(Implies(And(m>=0,n>=0,r==0,x==m,y==n),inv))

# 2 regra
pre = And(y>0,inv)
left = Implies(y & 1 == 1,substitute(substitute(substitute(substitute(inv,(y,y>>1)),(x,x<<1)),(r,r+x)),(y,y-1)))
right = Implies(y & 1 != 1,substitute(substitute(inv,(y,y>>1)),(x,x<<1)))

f2 = Implies(pre,And(left,right))

#prove(f2) #nao acaba

f3 = Implies(And(inv,y<=0),r==m*n)

prove(f3)
```

```
proved
proved
```

Usando havoc e WPC

```
assume m >= 0 and n >= 0 and r == 0 and x == m and y == n;
assert inv;
havoc y; havoc r; havoc x;
((assume y>0 and inv; if y & 1 == 1 then (y = y-1; r = r + x;) x = x<<1; y = y>>1;) || (assume y<=0 and inv));
assert r == m * n;


pre = m >= 0 and n >= 0 and r == 0 and x == m and y == n
inv = x*y+r==m*n and m>=0 and n>=0 and n>=y and y>=0
pos = r == m * n
```

$$\text{pre} \rightarrow (\text{inv} \land [\text{Ciclo}])$$

$$\equiv$$

$$\text{pre} \rightarrow (\text{inv} \land \forall y.\forall r.\forall x.[\text{Corpo}])$$

$$\equiv$$

$$\text{pre} \rightarrow (\text{inv} \land \forall y.\forall r.\forall x.(y > 0 \land inv \rightarrow (y\&1 = 1 \rightarrow inv[y-1/y][r+x/r][x << 1/x][y >> 1/y]) \land (y\&1 \neq 1 \rightarrow inv[x << 1/x][y >> 1/y]))) \land (y \leq 0 \land inv \rightarrow pos)$$

[16]:

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

f1 = Implies(y & 1 == 1,substitute(substitute(substitute(substitute(inv,(y,y>>1)),(x,x<<1)),(r,r+x)),(y,y-1)))
f2 = Implies(y & 1 != 1,substitute(substitute(inv,(y,y>>1)),(x,x<<1)))
f3 = Implies(And(y>0,inv),And(f1,f2))
f4 = Implies(And(y<=0,inv),pos)

prove(Implies(pre,And(inv,f3,f4)))
```

```
proved
```

Usando SPC e unfold

```
assume m>=0 and n>=0 and r==0 and x==m and y==n;

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

||

assume y>0;
assume y&1==1;
y0 = y - 1;
r0 = r + x;
x0 = x;
y1 = y0 >> 1;
x1 = x0 << 1;
```

```
r1 = r0;
    ||
assume y&1!=1;
y1 = y >> 1;
x1 = x << 1;
r1 = r;

assume y1<=0;
assert y1*x1+r1 = n*m;

...
```

[38]:

```
def unfold(N):

    pre = (And(m>=0,n>=0,r==0,x==m,y==n))
    l1 = []
    v = {}
    for i in range(2*N+1):
        v['x'+str(i)] = Const('x'+str(i),s)
        v['y'+str(i)] = Const('y'+str(i),s)
        v['r'+str(i)] = Const('r'+str(i),s)

    l1.append(And(v['y0'] == y,v['x0'] == x,v['r0'] == r))
    for i in range(1,2*N+1,2):

        y0 = v['y'+str(i-1)]
        y1 = v['y'+str(i)]
        y2 = v['y'+str(i+1)]
        x0 = v['x'+str(i-1)]
        x1 = v['x'+str(i)]
        x2 = v['x'+str(i+1)]
        r0 = v['r'+str(i-1)]
        r1 = v['r'+str(i)]
        r2 = v['r'+str(i+1)]
        k = And(y0>0,Or(And(y0&1==1,y1==y0-1,r1==r0+x0,x1==x0,y2==y1>>1,x2==x1<<1,r2==r1),And(y0&1!=1,y2==y0>>1,x2==x0<<1,r0==r2)))
        l1.append(k)

    r0 = v['r'+str(2*N)]

    return Implies(And(pre,And(l1)),r0==m*n)

prove(unfold(N))
```

```
proved
```