Lógica Computacional - TP3 Exercício 1 - G01

Bruno Dias da Gião A96544, João Luis da Cruz Pereira A95375, David Alberto Agra A95726

November 22, 2023

1 Exercício 1 - Enunciado

Considere-se de novo o algoritmo estendido de Euclides apresentado no TP2 mas usando o tipo dos inteiros e um parâmetro $N > 0$

```
INPUT  a, b : Int
assume  a > 0 and b > 0 and a < N and b < N
r, r', s, s', t, t' = a, b, 1, 0, 0, 1
while r' != 0
  q = r div r'
  r, r', s, s', t, t' = r', r - q × r', s', s - q × s', t', t - q × t'
OUTPUT r, s, t
```

Este exercício é dirigido às provas de segurança do algoritmo acima.

1. Construa um FOTS $\Sigma \equiv \langle \mathsf{X}, \mathsf{I}, \mathsf{T} \rangle$ usando este modelo nos inteiros.
2. Considere como propriedade de segurança `safety = (r > 0) and (r < N) and (r = a*s + b*t)` Prove usando k-indução que esta propriedade se verifica em qualquer traço do FOTS
3. Prove usando "Model-Checking" com interpelantes e invariantes prove também que esta propriedade é um invariante em qualquer traço de Σ.

2 Exercício 1 - Solução

2.1 Construção do FOTS

[39]:
```
from pysmt.shortcuts import *
from pysmt.typing import INT
import itertools
from random import randint
```

Consideremos a seguinte variação, trivialmente, equivalente do pseudocódigo apresentado, que ajudará na implementação do sistema de tal forma a que seja aceitável pelo solver `MathSAT` e de tal forma a que as atribuições, no cíclo, sejam lineares.

```
INPUT  a, b : Int
assume  a > 0 and b > 0 and a < N and b < N
r, r_prime, s, s_prime, t, t_prime = a, b, 1, 0, 0, 1
while r_prime != 0
    q = 0, cx = r, rx = 0, sx = 0, tx = 0
```

```
    while cx >= r_prime:
        cx -= r_prime, q += 1, rx += r_prime, sx += s_prime, tx += t_prime
    r, r_prime, s, s_prime, t, t_prime = r_prime, r - q × rx, s_prime, s - q × sx, t_prime, t -
OUTPUT r, s, t
```

Começemos, naturalmente, pela construção do FOTS Σ.

Ora, definemos:

- espaço de variáveis como $X_0, X_1, X_2, X_3, X_4, X_5, X_6 := (pc, r, r\_prime, s, s\_prime, t, t\_prime, q, cx, rx, sx, tx)$

[27]:
```
X = ["pc", "r", "r_prime", "s", "s_prime", "t", "t_prime", "q", "cx", "rx",␣
 ↪"sx", "tx"]
def genState(var,s,i):
    state = {}
    for v in var:
        state[v] = Symbol(v+'!'+s+str(i),INT)
    return state
```

- estados iniciais sendo determinados pelo predicado

$$init(a, b, N) = pc = 0 \wedge a > 0 \wedge b > 0 \wedge N > 0 \wedge N > a \wedge N > b$$

[28]:
```
def init(state, a, b, N):

    # Pre:
    A = Equals(state['pc'], Int(0))
    B = GT(Int(a), Int(0))
    C = GT(Int(b), Int(0))
    D = GT(Int(N), Int(0))
    E = GT(Int(N), Int(a))
    F = GT(Int(N), Int(b))
    return And(A,B,C,D,E,F)
```

- e transições:

$$\delta(0,1), \delta(1,2), \delta(1,f), \delta(2,3), \delta(3,4), \delta(4,3), \delta(3,5), \delta(5,1) \text{ e } \delta(6,6)$$

tais que:

[29]:
```
def trans(curr, prox, a, b):

    t01 = And(
        Equals(curr['pc'], Int(0)),
        Equals(prox['pc'], Int(1)),
        Equals(curr['r'], Int(a)),
        Equals(curr['r_prime'], Int(b)),
        Equals(curr['s'], Int(1)),
        Equals(curr['s_prime'], Int(0)),
        Equals(curr['t'], Int(0)),
```

```
        Equals(curr['t_prime'], Int(1)),
        Equals(curr['q'], Int(0)),
        Equals(curr["cx"], Int(0)),
        Equals(prox["rx"], Int(0)),
        Equals(prox["sx"], Int(0)),
        Equals(prox["tx"], Int(0)),
        Equals(curr["cx"], prox["cx"]),
        Equals(prox["rx"], curr['rx']),
        Equals(prox["sx"], curr['sx']),
        Equals(prox["tx"], curr['tx']),
        Equals(curr['r'], prox['r']),
        Equals(curr['r_prime'], prox['r_prime']),
        Equals(curr['s'], prox['s']),
        Equals(curr['s_prime'], prox['s_prime']),
        Equals(curr['t'], prox['t']),
        Equals(curr['t_prime'], prox['t_prime']),
        Equals(curr['q'], prox['q'])
    )

    t1f = And(
        Equals(curr['pc'], Int(1)),
        Equals(prox['pc'], Int(6)),
        Equals(curr['r_prime'], Int(0)),
        Equals(prox['r_prime'], curr['r_prime']),
        Equals(curr['r'], prox['r']),
        Equals(curr['s'], prox['s']),
        Equals(curr['s_prime'], prox['s_prime']),
        Equals(curr['t'], prox['t']),
        Equals(curr["cx"], prox["cx"]),
        Equals(prox["rx"], curr['rx']),
        Equals(prox["sx"], curr['sx']),
        Equals(prox["tx"], curr['tx']),
        Equals(curr["q"], prox["q"]),
        Equals(curr['t_prime'], prox['t_prime'])
    )
    t12 = And(
        Equals(curr['pc'], Int(1)),
        Equals(prox['pc'], Int(2)),
        Not(Equals(curr['r_prime'], Int(0))),
        Equals(curr['r_prime'], prox['r_prime']),
        Equals(curr['q'], prox['q']),
        Equals(curr['r'], prox['r']),
        Equals(curr['s'], prox['s']),
        Equals(curr['s_prime'], prox['s_prime']),
        Equals(curr['t'], prox['t']),
        Equals(curr["cx"], prox["cx"]),
        Equals(prox["rx"], curr['rx']),
```

```
        Equals(prox["sx"], curr['sx']),
        Equals(prox["tx"], curr['tx']),
        Equals(curr["q"], prox["q"]),
        Equals(curr['t_prime'], prox['t_prime'])
    )
    t23 = And(
        Equals(curr['pc'], Int(2)),
        Equals(prox['pc'], Int(3)),
        Equals(prox["cx"], curr['r']),
        Equals(prox["rx"], Int(0)),
        Equals(prox["sx"], Int(0)),
        Equals(prox["tx"], Int(0)),
        Equals(prox['q'], Int(0)),
        Equals(curr['r'], prox['r']),
        Equals(curr['r_prime'], prox['r_prime']),
        Equals(curr['s'], prox['s']),
        Equals(curr['s_prime'], prox['s_prime']),
        Equals(curr['t'], prox['t']),
        Equals(curr['t_prime'], prox['t_prime']),

    )
    t34 = And(
        Equals(curr['pc'], Int(3)),
        Equals(prox['pc'], Int(4)),
        GE(curr["cx"], curr['r_prime']),
        Equals(curr['r_prime'], prox['r_prime']),
        Equals(curr['q'], prox['q']),
        Equals(curr['r'], prox['r']),
        Equals(curr['s'], prox['s']),
        Equals(curr['s_prime'], prox['s_prime']),
        Equals(curr['t'], prox['t']),
        Equals(curr['t_prime'], prox['t_prime']),
        Equals(curr["cx"], prox["cx"]),
        Equals(prox["rx"], curr['rx']),
        Equals(prox["sx"], curr['sx']),
        Equals(prox["tx"], curr['tx'])
    )
    t43 = And(
        Equals(curr['pc'], Int(4)),
        Equals(prox['pc'], Int(3)),
        Equals(prox['q'], Plus(curr['q'], Int(1))),
        Equals(prox["cx"], Minus(curr["cx"], curr['r_prime'])),
        Equals(prox["rx"], Plus(curr['rx'], curr['r_prime'])),
        Equals(prox["sx"], Plus(curr['sx'], curr['s_prime'])),
        Equals(prox["tx"], Plus(curr['tx'], curr['t_prime'])),
        Equals(curr['r'], prox['r']),
        Equals(curr['r_prime'], prox['r_prime']),
```

```
        Equals(curr['s'], prox['s']),
        Equals(curr['s_prime'], prox['s_prime']),
        Equals(curr['t'], prox['t']),
        Equals(curr['t_prime'], prox['t_prime'])
    )
    t35 = And(
        Equals(curr['pc'], Int(3)),
        Equals(prox['pc'], Int(5)),
        LT(curr["cx"], curr['r_prime']),
        Equals(prox['r_prime'], curr['r_prime']),
        Equals(curr['r'], prox['r']),
        Equals(curr['s'], prox['s']),
        Equals(curr['s_prime'], prox['s_prime']),
        Equals(curr['t'], prox['t']),
        Equals(curr['t_prime'], prox['t_prime']),
        Equals(prox['q'], curr['q']),
        Equals(curr["cx"], prox["cx"]),
        Equals(prox["rx"], curr['rx']),
        Equals(prox["sx"], curr['sx']),
        Equals(prox["tx"], curr['tx'])

    )


    t51 = And(
        Equals(curr['pc'], Int(5)),
        Equals(prox['pc'], Int(1)),
        Equals(prox['r'], curr['r_prime']),
        Equals(prox['r_prime'], Minus(curr['r'], curr['rx'])),
        Equals(prox['s'], curr['s_prime']),
        Equals(prox['s_prime'], Minus(curr['s'], curr['sx'])),
        Equals(prox['t'], curr['t_prime']),
        Equals(prox['t_prime'], Minus(curr['t'], curr['tx'])),
        Equals(prox['q'], curr['q']),
        Equals(curr["cx"], prox["cx"]),
        Equals(prox["rx"], curr['rx']),
        Equals(prox["sx"], curr['sx']),
        Equals(prox["tx"], curr['tx'])
    )

    t66 = And(
        Equals(curr['pc'], Int(6)),
        Equals(prox['pc'], Int(6)),
        Equals(curr['r'], prox['r']),
        Equals(curr['r_prime'], prox['r_prime']),
        Equals(curr['s'], prox['s']),
        Equals(curr['s_prime'], prox['s_prime']),
```

```
        Equals(curr['t'], prox['t']),
        Equals(curr['t_prime'], prox['t_prime']),
        Equals(curr['q'], prox['q']),
        Equals(curr['cx'], prox['cx']),
        Equals(prox["rx"], curr['rx']),
        Equals(prox["sx"], curr['sx']),
        Equals(prox["tx"], curr['tx'])
    )

    return Or(t01, t1f, t12, t23, t34, t43, t35, t51, t66)
```

Definemos uma função que gera o traço de Σ:

[30]:
```
def genTrace(var,init,trans,post,n, a, b, N):
    with Solver(name="z3") as s:
        X = [genState(var,'X',i) for i in range(n+1)]   # cria n+1 estados (com␣
 ↪etiqueta X)
        I = init(X[0], a, b , N)
        Tks = [trans(X[i],X[i+1], a, b) for i in range(n)]

        error = Equals(X[-1]['pc'], Int(6))

        if s.solve([I,And(Tks), error]):      # testa se I /\ T^n  é␣
 ↪satisfazível
            for i in range(n+1):
                print("Estado:",i)
                for v in X[i]:
                    print("          ",v,'=',s.get_value(X[i][v]))
            # OUTPUT
            print(f"r = {s.get_value(X[-1]['r'])}, s = {s.
 ↪get_value(X[-1]['s'])}, t = {s.get_value(X[-1]['t'])}")
        else:
            print("not sat")
```

As seguintes definição são codificações das condições que pretendemos provar:

[31]:
```
def safety(state, a, b, N):
    A = GT(state['r'], Int(0))
    B = GT(Int(N), state['r'])
    C = Equals(state['r'], Plus(Times(Int(a), state['s']), Times(Int(b),␣
 ↪state['t'])))
    return And(A, B, C)

def stronger(state, a, b, N):
    return And(safety(state,a,b,N), init(state,a,b,N));

def error(state, a, b, N):
    return Not(safety(state, a, b, N))
```

Definemos também uma função que implemente k-induction:

[32]:
```
def kinduction_always(var, init, trans, error, n, a, b, N, inv):
    with Solver(name="z3") as solver:
        X = [genState(var, 'X', i) for i in range(n+1)]
        I = init(X[0], a, b, N)
        solver.add_assertion(I)

        for i in range(n-1):
            solver.add_assertion(trans(X[i], X[i+1], a, b))

        solver.add_assertion(Equals(X[-1]['pc'], Int(6)))

        for i in range(n):
            solver.push()
            solver.add_assertion(Not(inv(X[i], a, b, N)))
            if solver.solve():
                print(f"> Contradição! O invariante não se verifica nos k␣
↪estados iniciais.")
                for i, state in enumerate(X):
                    print(f"> State {i}: pc = {solver.
↪get_value(state['pc'])}\nq = {solver.get_value(state['q'])}\ns = {solver.
↪get_value(state['s'])}\nt = {solver.get_value(state['t'])}\nr = {solver.
↪get_value(state['r'])}\ns' = {solver.get_value(state['s_prime'])}\nt' =␣
↪{solver.get_value(state['t_prime'])}\nr' = {solver.
↪get_value(state['r_prime'])}")
                return
            solver.pop()

        X2 = [genState(var, 'X', i+n) for i in range(n+1)]

        for i in range(n):
            solver.add_assertion(inv(X2[i], a, b, N))
            solver.add_assertion(trans(X2[i],X2[i+1], a, b))

        solver.add_assertion(Not(inv(X2[-1], a, b, N)))

        solver.add_assertion(Equals(X2[-1]['pc'], Int(6)))

        if solver.solve():
            print(f"> Contradição! O passo indutivo não se verifica.")
            for i, state in enumerate(X):
                print(f"> State {i}: pc = {solver.get_value(state['pc'])}\nq =␣
↪{solver.get_value(state['q'])}\ns = {solver.get_value(state['s'])}\nt =␣
↪{solver.get_value(state['t'])}\nr = {solver.get_value(state['r'])}\ns' =␣
↪{solver.get_value(state['s_prime'])}\nt' = {solver.
↪get_value(state['t_prime'])}\nr' = {solver.get_value(state['r_prime'])}")
```

```
            return

        print(f"> A propriedade verifica-se por k-indução (k={n}).")
```

Continuemos agora para a realização do Model Checking com uso de interpolantes e invariantes.

[33]:
```
def baseName(s):
    return ''.join(list(itertools.takewhile(lambda x: x!='!', s)))

def rename(form,state):
    vs = get_free_variables(form)
    pairs = [ (x,state[baseName(x.symbol_name())]) for x in vs ]
    return form.substitute(dict(pairs))

def same(state1,state2):
    return And([Equals(state1[x],state2[x]) for x in state1])

def invert(trans, a, b):
    return lambda prev, next_ : trans(next_, prev, a, b)
```

[34]:
```
def model_checking(var, init, trans, error, Nb, Mb, a, b, N):
    with Solver(name="z3") as solver:

        # Criar todos os estados que poderão vir a ser necessários.
        X = [genState(var, 'X', i) for i in range(Nb+1)]
        Y = [genState(var, 'Y', i) for i in range(Mb+1)]
        transt = invert(trans, a, b)

        # Estabelecer a ordem pela qual os pares (n, m) vão surgir. Por exemplo:
        order = sorted([(a, b) for a in range(1, Nb+1) for b in range(1,␣
 ↪Mb+1)], key=lambda tup:tup[0]+tup[1])

        # Step 1 implícito na ordem de 'order' e nas definições Rn, Um.
        for (n, m) in order:
            # Step 2
            I= init(X[0], a, b , N)
            Tn = And([trans(X[i], X[i+1], a, b) for i in range(n)])
            Rn = And(I, Tn)

            E = error(Y[0], a, b, N)
            Bm = And([transt(Y[i], Y[i+1]) for i in range(m)])
            Um = And(E, Bm)

            Vnm = And(Rn, same(X[n], Y[m]), Um)
            if solver.solve([Vnm]):
                print("> O sistema é inseguro.")
                return
```

```
            else:
                # Step 3
                A = And(Rn, same(X[n], Y[m]))
                B = Um
                C = binary_interpolant(A, B)

                # Salvaguardar cálculo bem-sucedido do interpolante.
                if C is None:
                    print("> O interpolante é None.")
                    break

                # Step 4
                C0 = rename(C, X[0])
                T = trans(X[0], X[1], a, b)
                C1 = rename(C, X[1])

                if not solver.solve([C0, T, Not(C1)]):
                    # C é invariante de T.
                    print("> O sistema é seguro.")
                    return
                else:
                    # Step 5.1
                    S = rename(C, X[n])
                    while True:
                        # Step 5.2
                        T = trans(X[n], Y[m], a, b)
                        A = And(S, T)
                        if solver.solve([A, Um]):
                            print("> Não foi encontrado majorante.")
                            break
                        else:
                            # Step 5.3
                            C = binary_interpolant(A, Um)
                            Cn = rename(C, X[n])
                            if not solver.solve([Cn, Not(S)]):
                                # Step 5.4.
                                # C(Xn) -> S é tautologia.
                                print("> O sistema é seguro.")
                                return
                            else:
                                # Step 5.5.
                                # C(Xn) -> S não é tautologia.
                                S = Or(S, Cn)
    print("> Não foi provada a segurança ou insegurança do sistema.")
```

O seguinte exemplo visualizará a boa execução do código:

[35]:
```
K = 30
N = 20
```

[36]:
```
n = 13
m = 13
```

[37]:
```
a = 10
b = 8
```

[38]:
```
genTrace(X, init, trans, error, K, a, b, N)
kinduction_always(X, init, trans, error, K, a, b, N, safety)
model_checking(X, stronger, trans, error, n, m, a, b, N)
```

```
Estado: 0
           pc = 0
           r = 10
           r_prime = 8
           s = 1
           s_prime = 0
           t = 0
           t_prime = 1
           q = 0
           cx = 0
           rx = 0
           sx = 0
           tx = 0
Estado: 1
           pc = 1
           r = 10
           r_prime = 8
           s = 1
           s_prime = 0
           t = 0
           t_prime = 1
           q = 0
           cx = 0
           rx = 0
           sx = 0
           tx = 0
Estado: 2
           pc = 2
           r = 10
           r_prime = 8
           s = 1
           s_prime = 0
           t = 0
           t_prime = 1
           q = 0
```

```
           cx = 0
           rx = 0
           sx = 0
           tx = 0
Estado: 3
           pc = 3
           r = 10
           r_prime = 8
           s = 1
           s_prime = 0
           t = 0
           t_prime = 1
           q = 0
           cx = 10
           rx = 0
           sx = 0
           tx = 0
Estado: 4
           pc = 4
           r = 10
           r_prime = 8
           s = 1
           s_prime = 0
           t = 0
           t_prime = 1
           q = 0
           cx = 10
           rx = 0
           sx = 0
           tx = 0
Estado: 5
           pc = 3
           r = 10
           r_prime = 8
           s = 1
           s_prime = 0
           t = 0
           t_prime = 1
           q = 1
           cx = 2
           rx = 8
           sx = 0
           tx = 1
Estado: 6
           pc = 5
           r = 10
           r_prime = 8
           s = 1
```

```
            s_prime = 0
            t = 0
            t_prime = 1
            q = 1
            cx = 2
            rx = 8
            sx = 0
            tx = 1
Estado: 7
            pc = 1
            r = 8
            r_prime = 2
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 2
            rx = 8
            sx = 0
            tx = 1
Estado: 8
            pc = 2
            r = 8
            r_prime = 2
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 2
            rx = 8
            sx = 0
            tx = 1
Estado: 9
            pc = 3
            r = 8
            r_prime = 2
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 0
            cx = 8
            rx = 0
            sx = 0
            tx = 0
Estado: 10
```

```
            pc = 4
            r = 8
            r_prime = 2
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 0
            cx = 8
            rx = 0
            sx = 0
            tx = 0
Estado: 11
            pc = 3
            r = 8
            r_prime = 2
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 6
            rx = 2
            sx = 1
            tx = -1
Estado: 12
            pc = 4
            r = 8
            r_prime = 2
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 6
            rx = 2
            sx = 1
            tx = -1
Estado: 13
            pc = 3
            r = 8
            r_prime = 2
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 2
            cx = 4
```

```
           rx = 4
           sx = 2
           tx = -2
Estado: 14
           pc = 4
           r = 8
           r_prime = 2
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 2
           cx = 4
           rx = 4
           sx = 2
           tx = -2
Estado: 15
           pc = 3
           r = 8
           r_prime = 2
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 3
           cx = 2
           rx = 6
           sx = 3
           tx = -3
Estado: 16
           pc = 4
           r = 8
           r_prime = 2
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 3
           cx = 2
           rx = 6
           sx = 3
           tx = -3
Estado: 17
           pc = 3
           r = 8
           r_prime = 2
           s = 0
           s_prime = 1
```

```
           t = 1
           t_prime = -1
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 18
           pc = 5
           r = 8
           r_prime = 2
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 19
           pc = 1
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 20
           pc = 6
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 21
           pc = 6
```

```
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 22
           pc = 6
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 23
           pc = 6
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 24
           pc = 6
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
```

```
           sx = 4
           tx = -4
Estado: 25
           pc = 6
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 26
           pc = 6
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 27
           pc = 6
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
           t_prime = 5
           q = 4
           cx = 0
           rx = 8
           sx = 4
           tx = -4
Estado: 28
           pc = 6
           r = 2
           r_prime = 0
           s = 1
           s_prime = -4
           t = -1
```

```
            t_prime = 5
            q = 4
            cx = 0
            rx = 8
            sx = 4
            tx = -4
Estado: 29
            pc = 6
            r = 2
            r_prime = 0
            s = 1
            s_prime = -4
            t = -1
            t_prime = 5
            q = 4
            cx = 0
            rx = 8
            sx = 4
            tx = -4
Estado: 30
            pc = 6
            r = 2
            r_prime = 0
            s = 1
            s_prime = -4
            t = -1
            t_prime = 5
            q = 4
            cx = 0
            rx = 8
            sx = 4
            tx = -4
r = 2, s = 1, t = -1
> A propriedade verifica-se por k-indução (k=30).
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
```

```
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
```

```
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
```

```
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> O sistema é seguro.
```

3 Exercício 1 - Exemplos

[43]:
```
N = int(input("> N: ")) + 1
a = randint(1,N)
b = randint(1,N)
K = randint(20,50)
n = 15
m = 15
```

[44]:
```
genTrace(X, init, trans, error, K, a, b, N)
kinduction_always(X, init, trans, error, K, a, b, N, safety)
model_checking(X, stronger, trans, error, n, m, a, b, N)
```

```
Estado: 0
            pc = 0
            r = 3
            r_prime = 3
            s = 1
            s_prime = 0
            t = 0
            t_prime = 1
            q = 0
            cx = 0
            rx = 0
            sx = 0
            tx = 0
Estado: 1
            pc = 1
            r = 3
            r_prime = 3
            s = 1
            s_prime = 0
            t = 0
            t_prime = 1
            q = 0
            cx = 0
            rx = 0
            sx = 0
            tx = 0
Estado: 2
            pc = 2
            r = 3
            r_prime = 3
            s = 1
            s_prime = 0
            t = 0
            t_prime = 1
            q = 0
```

```
            cx = 0
            rx = 0
            sx = 0
            tx = 0
Estado: 3
            pc = 3
            r = 3
            r_prime = 3
            s = 1
            s_prime = 0
            t = 0
            t_prime = 1
            q = 0
            cx = 3
            rx = 0
            sx = 0
            tx = 0
Estado: 4
            pc = 4
            r = 3
            r_prime = 3
            s = 1
            s_prime = 0
            t = 0
            t_prime = 1
            q = 0
            cx = 3
            rx = 0
            sx = 0
            tx = 0
Estado: 5
            pc = 3
            r = 3
            r_prime = 3
            s = 1
            s_prime = 0
            t = 0
            t_prime = 1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 6
            pc = 5
            r = 3
            r_prime = 3
            s = 1
```

```
            s_prime = 0
            t = 0
            t_prime = 1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 7
            pc = 1
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 8
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 9
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 10
```

```
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 11
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 12
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 13
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
```

```
           rx = 3
           sx = 0
           tx = 1
Estado: 14
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
Estado: 15
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
Estado: 16
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
Estado: 17
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
```

```
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 18
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 19
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 20
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 21
            pc = 6
```

```
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 22
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 23
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
            sx = 0
            tx = 1
Estado: 24
            pc = 6
            r = 3
            r_prime = 0
            s = 0
            s_prime = 1
            t = 1
            t_prime = -1
            q = 1
            cx = 0
            rx = 3
```

```
           sx = 0
           tx = 1
Estado: 25
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
Estado: 26
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
Estado: 27
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
Estado: 28
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
```

```
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
Estado: 29
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
Estado: 30
           pc = 6
           r = 3
           r_prime = 0
           s = 0
           s_prime = 1
           t = 1
           t_prime = -1
           q = 1
           cx = 0
           rx = 3
           sx = 0
           tx = 1
r = 3, s = 0, t = 1
> A propriedade verifica-se por k-indução (k=30).
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
```

```
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> Não foi encontrado majorante.
> O sistema é seguro.
```

[ ]: