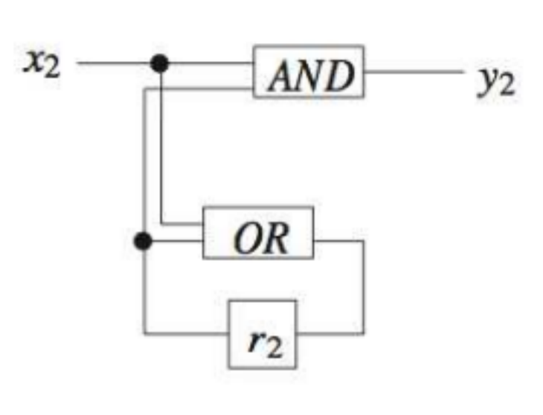
TASK 1&2 Tutorials FSE

Name: Heja Bibani
Student Number: 16301173

TASK 1

Consider the following Sequential hardware circuit:



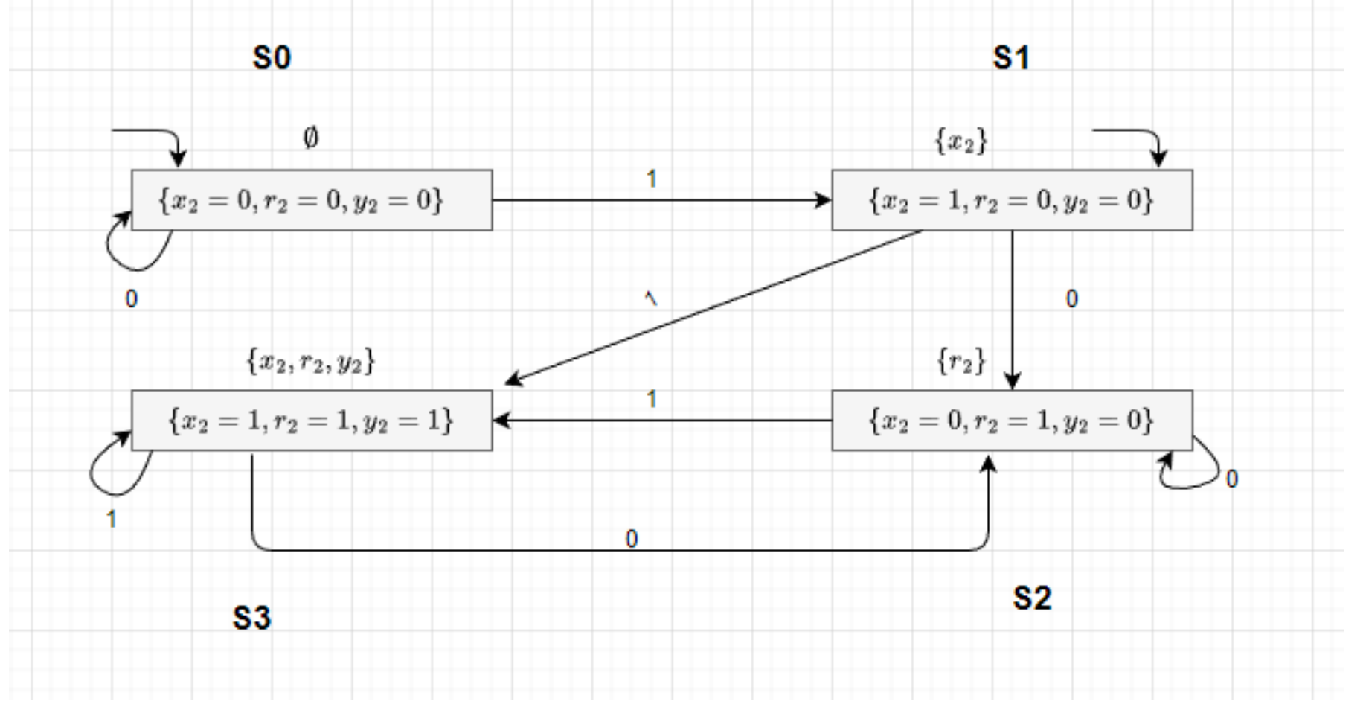
Where x_2 is an input variable, r_2 a register, and y_2 an output variable. The control function for output variable y_2 is defined by $\lambda_{y_2} = x_2 \wedge r_2$ and the register evaluation changes according to the circuit funciton $\delta_{r_2} = x_2 \vee r_2$. Initially, r_2 = 0, and x_2 can be 0 or 1. Once r_2 is evaluated to be 1 (according to $\delta_{r_2} = x_2 \vee r_2$), it will keep that value.

Specify a transition systems description

$TS = \{S, Act, \rightarrow I, AP, L\}$

$S = \{S_0, S_1, S_2, S_3\}$
$S_0 = \langle x_2 = 0, y_2 = 0, r_2 = 0\rangle$
$S_1 = \langle x_2 = 1, y_2 = 0, r_2 = 0\rangle$
$S_2 = \langle x_2 = 0, y_2 = 0, r_2 = 1\rangle$
$S_3 = \langle x_2 = 1, y_2 = 1, r_2 = 1\rangle$
$Act = \{input0, input1\}$
$\rightarrow = \{(S_0, input0, S_0), (S_0, input1, S_1), (S_1, input0, S_2), (S_1, input1, S_3), (S_2, input0, S_2), (S_2, input1, S_3), (S_3, input0, S_2), (S_3, input1, S_3)\}$
$I = \{S_0, S_1\}$
$AP = \{x_2, y_2, r_2\}$
$L(S_0) = \emptyset$
$L(S_1) = \{x_2\}$
$L(S_2) = \{r_2\}$
$L(S_3) = \{x_2, r_2, y_2\}$

Specify a transition systems diagram

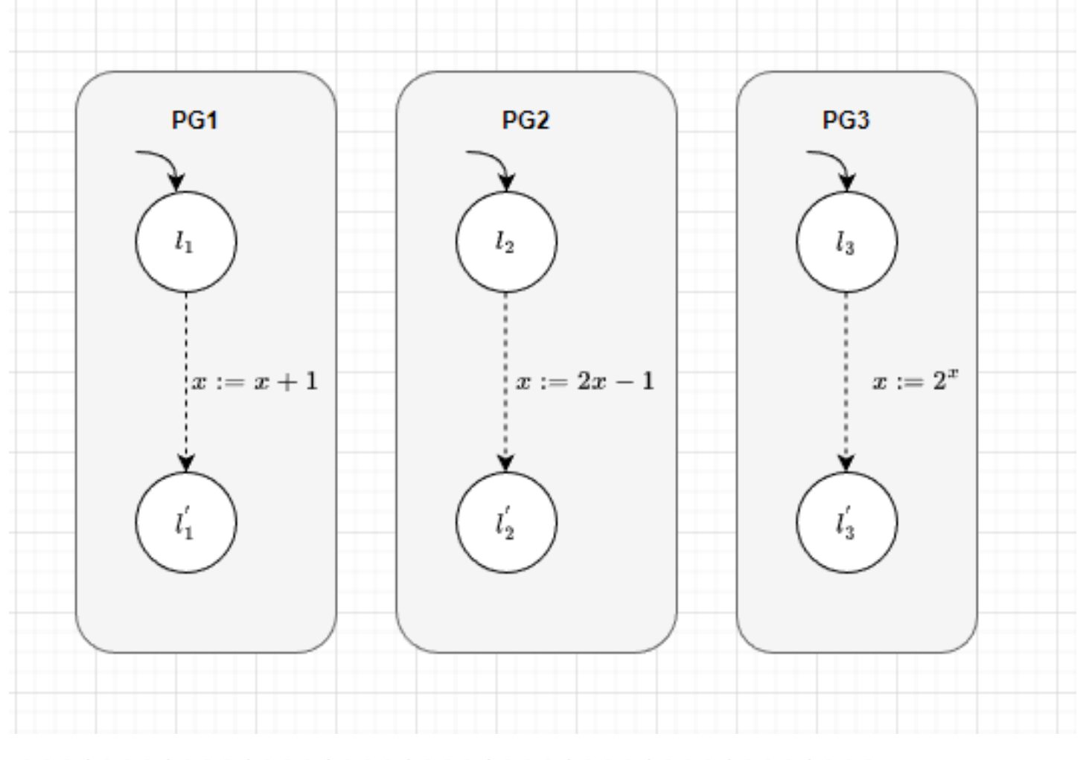


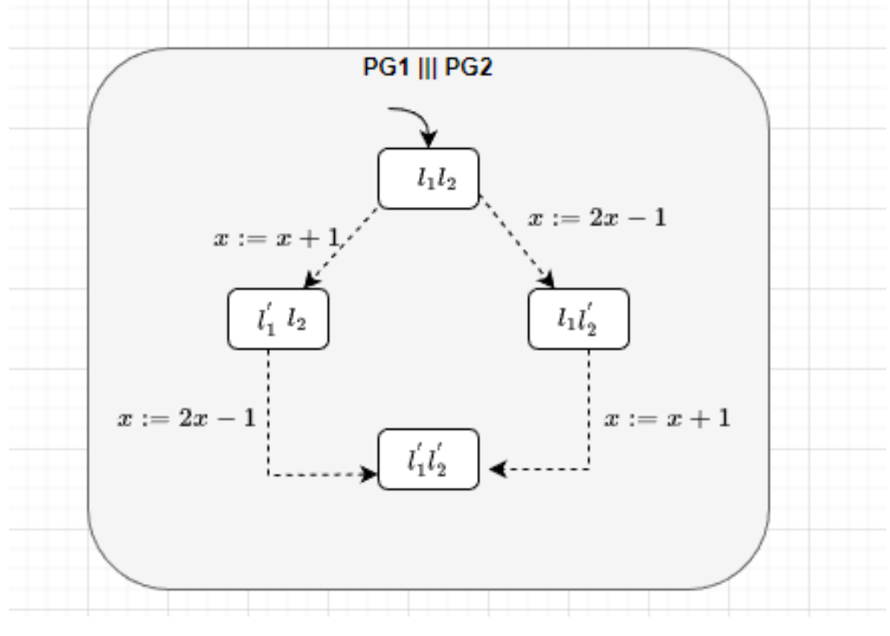
Consider the following Parallel Program

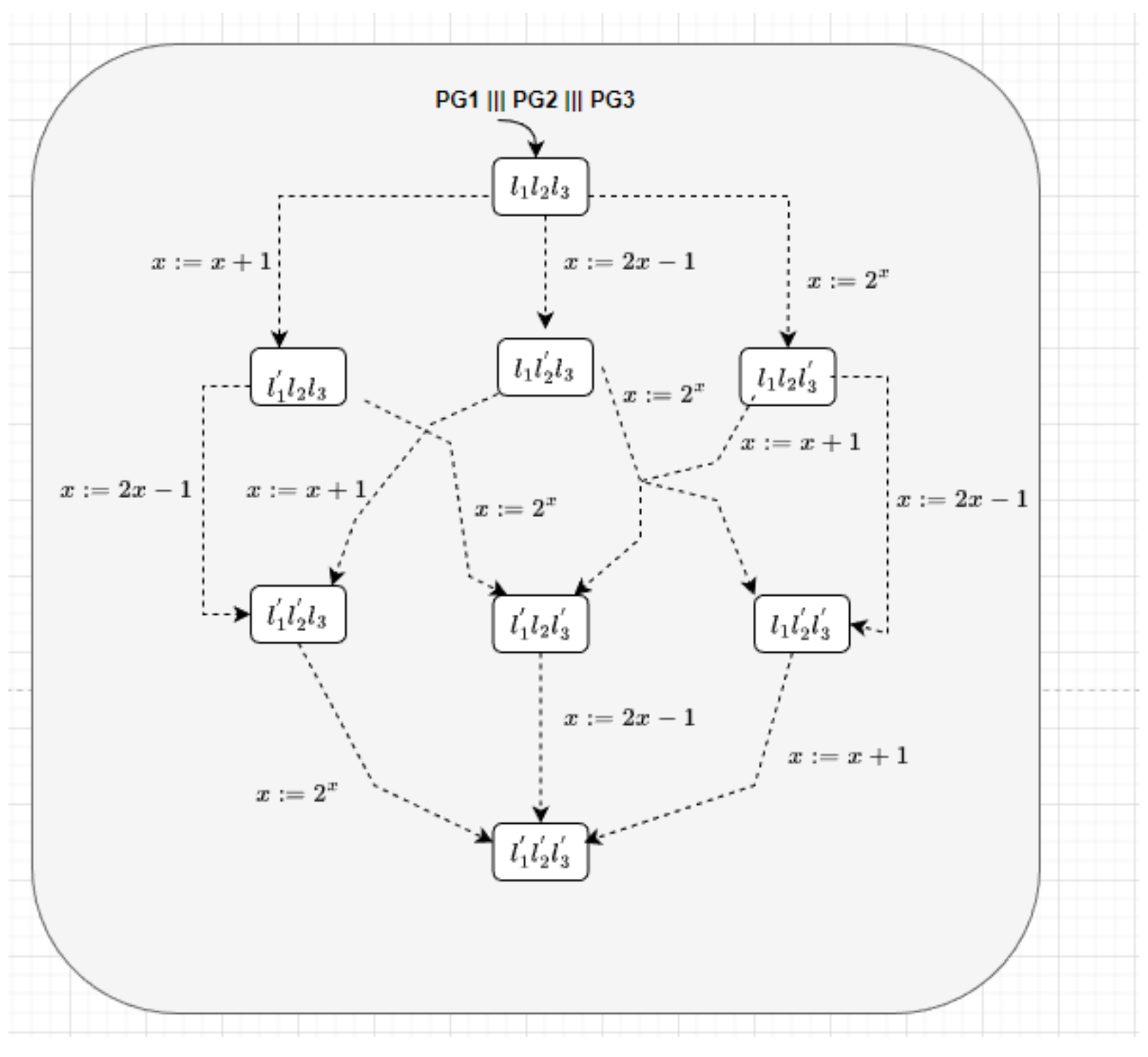
$x := x + 1 \;|||\; x := 2.x - 1 \;|||\; x := 2^x$

We assume that initially x = 3. Describe the program graph (graph only) and transition system of this parallel program. For the transition system, you need to provide both the transition system graph, and the transition system description.

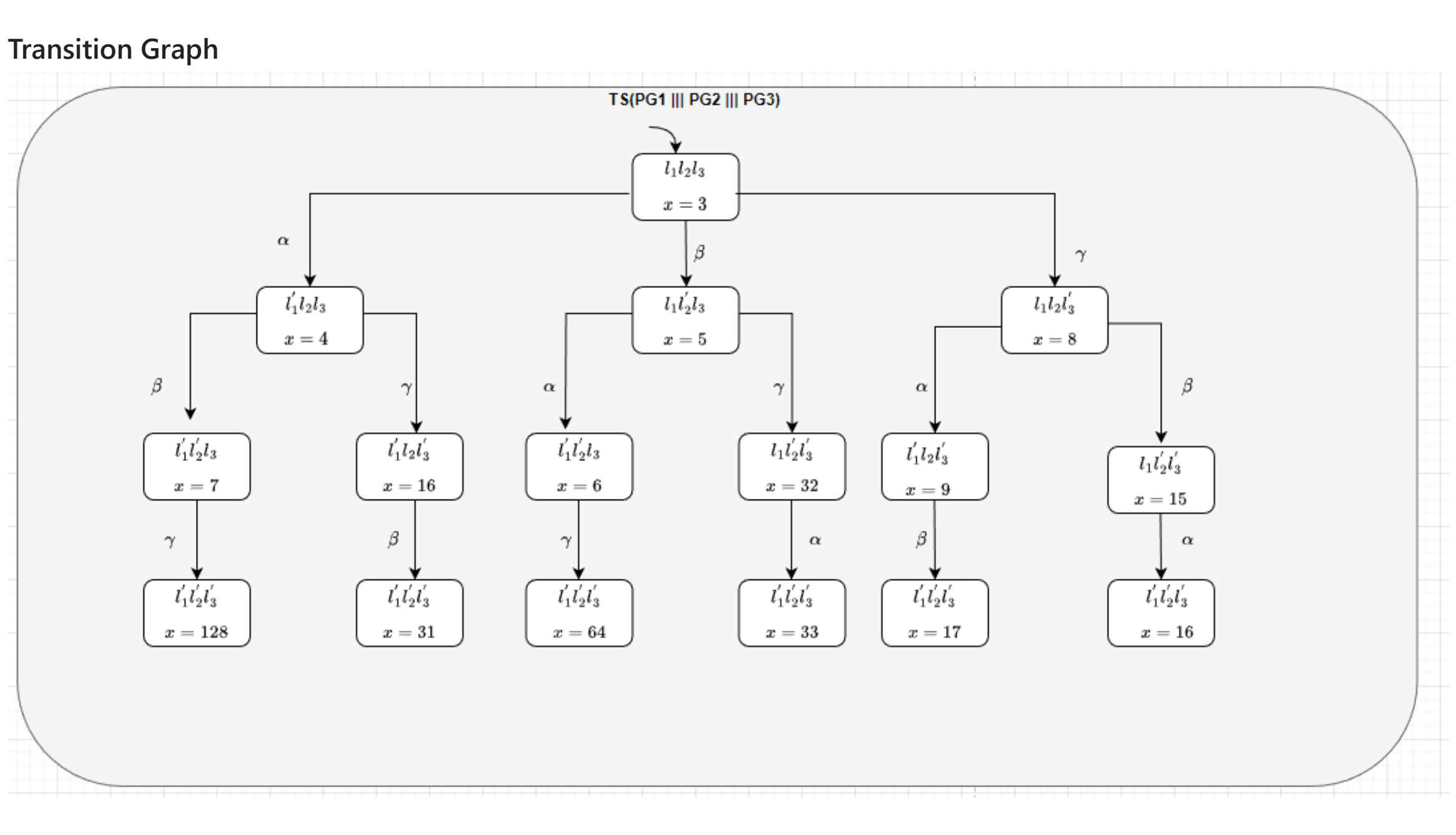
Program Graph







Transition Graph



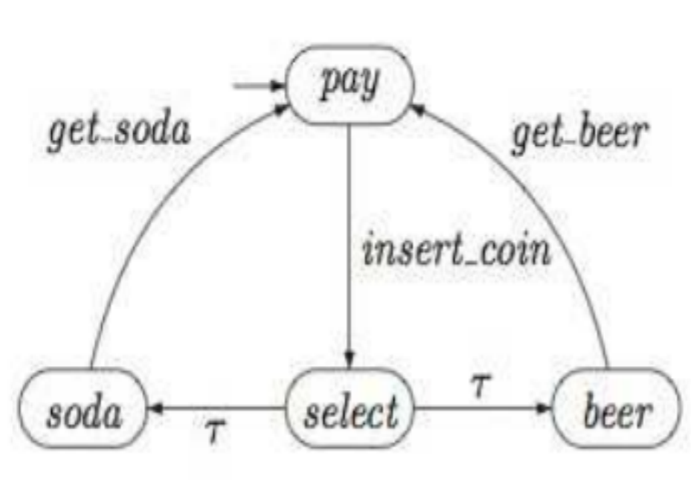
Transition System Description

$TS = \{S, Act, \rightarrow I, AP, L\}$
$S = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}\}$
$Where$
$S_0 = \langle l_1, l_2, l_3, x = 3\rangle$
$S_1 = \langle l_1', l_2, l_3, x = 4\rangle$
$S_2 = \langle l_1', l_2', l_3, x = 7\rangle$
$S_3 = \langle l_1', l_2', l_3', x = 128\rangle$
$S_4 = \langle l_1', l_2, l_3', x = 16\rangle$
$S_5 = \langle l_1', l_2', l_3', x = 31\rangle$
$S_6 = \langle l_1, l_2', l_3, x = 5\rangle$
$S_7 = \langle l_1', l_2', l_3, x = 6\rangle$
$S_8 = \langle l_1', l_2', l_3', x = 64\rangle$
$S_9 = \langle l_1, l_2', l_3', x = 32\rangle$
$S_{10} = \langle l_1', l_2', l_3', x = 33\rangle$
$S_{11} = \langle l_1, l_2, l_3', x = 8\rangle$
$S_{12} = \langle l_1', l_2, l_3', x = 9\rangle$
$S_{13} = \langle l_1', l_2', l_3', x = 17\rangle$
$S_{14} = \langle l_1, 1_2', l_3', x = 15\rangle$
$S_{15} = \langle l_1', l_2', l_3', x = 16\rangle$
$Act = \{\alpha, \beta, \gamma\}$
$Where\ \alpha \equiv x := x + 1\ , \beta \equiv x := 2x - 1, \gamma \equiv x := 2^x$
$\rightarrow = \{(S_0, \alpha, S_1), (S_1, \beta, S_2), (S_2, \gamma, S_3), (S_1, \gamma, S_4), (S_4, \beta, S_5), (S_0, \beta, S_6), (S_6, \alpha, S_7), (S_7, \gamma, S_8), (S_6, \gamma, S_9), (S_9, \alpha, S_{10}), (S_0, \gamma, S_{11}), (S_{11}, \alpha, S_{12}), (S_{12}, \beta, S_{13}), (S_{11}, \beta, S_{14}), (S_{14}, \alpha, S_{15})\}$
$I = \{S_0\}$
$AP = \{l_1, l_2, l_3, l_1', l_2', l_3', x = 3, x = 4, x = 5, x = 6, x = 7, x = 8, x = 9, x = 15, x = 16, x = 17, x = 31, x = 32, x = 33, x = 64, x = 128\}$
$L(S_0) = \{l_1, l_2, l_3, x = 3\}$
$L(S_1) = \{l_1', l_2, l_3, x = 4\}$
$L(S_2) = \{l_1', l_2', l_3, x = 7\}$
$L(S_3) = \{l_1', l_2', l_3', x = 128\}$
$L(S_4) = \{l_1', l_2, l_3', x = 16\}$
$L(S_5) = \{l_1', l_2', l_3', x = 31\}$
$L(S_6) = \{l_1, l_2', l_3, x = 5\}$
$L(S_7) = \{l_1', l_2', l_3, x = 6\}$
$L(S_8) = \{l_1', l_2', l_3', x = 64\}$
$L(S_9) = \{l_1, l_2', l_3', x = 32\}$
$L(S_{10}) = \{l_1', l_2', l_3', x = 33\}$
$L(S_{11}) = \{l_1, l_2, l_3', x = 8\}$
$L(S_{12}) = \{l_1', l_2, l_3', x = 9\}$
$L(S_{13}) = \{l_1', l_2', l_3', x = 17\}$
$L(S_{14}) = \{l_1, 1_2', l_3', x = 15\}$
$L(S_{15}) = \{l_1', l_2', l_3', x = 16\}$

Task 2

Question 1

Consider the following transition system T S of the simple beverage vending machine:



Prove that $TS \models \Box\Diamond(soda \vee beer), TS \not\models \Box soda$ and $TS \not\models \Box\Diamond(soda \wedge beer)$
$TS = \{S, Act, \rightarrow I, AP, L\}$
$S = \{pay, select, beer, soda\}$
$Act = \{insert\_coin, \tau, get\_soda, get\_beer\}$
$\rightarrow = \{pay, insert\_coin, select), (select, \tau, soda), (soda, get\_soda, pay), (select, \tau, beer), (beer, get\_beer, pay)\}$
$I = \{pay\}$
$AP = \{paid, soda, beer\}$
$L(pay) = \emptyset$
$L(soda) = \{paid, soda\}$
$L(beer) = \{paid, beer\}$
$L(select) = \{paid\}$

We note the following to be true and we will use this as a means to prove the three satisfication relations.

$Lets \in S$, we write $s \models \varphi$, iff for every execution path π of TS starting at s, that is, $\pi \in Paths(s)$, we have $\pi \models \varphi. TS \models \varphi$ iff for all initial states $s_0 \in I, s_0 \models \varphi$

To prove that TS satisfies or does not satisfy a LTL formula we have to find a path which is a counter-example. Thus, for each of the proof we will proves these in this way. For this transition system all paths start at pay. Thus,

----

(1) $TS \models \Box\Diamond(soda \vee beer)$ iff for all initial states $pay \in I, pay \models \Box\Diamond(soda \vee beer)$, $\forall\pi \in Paths(pay)$

A path of transition system TS is an initial, maximal path fragment. There are two main paths in the transition system that always lead to beer or soda, and the re-arrangement of the paths are arbitrary considering only these two outcomes are possible. The only difference is that some of the paths lead to both outcomes of soda and beer. These are the main paths that are in the transition system:

Let $Paths(pay) = \{\pi_1, \pi_2, \pi_3, \pi_4\}$

1. $\pi_1 = [(pay, select, beer, pay, select, beer, \ldots)] = (pay, select, beer)^\omega$
2. $\pi_2 = [(pay, select, soda, pay, select, soda, \ldots)] = (pay, select, soda)^\omega$
3. $\pi_3 = [(pay, select, soda, pay, select, beer, pay \ldots)] = ((pay, select, soda)^\omega, (pay, select, beer)^\omega)^\omega$
4. $\pi_4 = [(pay, select, beer, pay, select, soda, pay \ldots)] = ((pay, select, beer)^\omega, (pay, select, soda)^\omega)^\omega$

The traces for each path:

1. $trace(\pi_1) = (\emptyset, \{paid\}, \{paid, beer\})^\omega$
2. $trace(\pi_2) = (\emptyset, \{paid\}, \{paid, soda\})^\omega$
3. $trace(\pi_3) = ((\emptyset, \{paid\}, \{paid, soda\})^\omega, (\emptyset, \{paid\}, \{paid, beer\})^\omega)^\omega$
4. $trace(\pi_4) = ((\emptyset, \{paid\}, \{paid, beer\})^\omega, (\emptyset, \{paid\}, \{paid, soda\})^\omega)^\omega$

To prove that this satisifies the formula above we will look at each path to see if there is a counter-example, we are expecting that there should be no counter-example:

When we look at all the paths we notice that they must hold since $beer(state) \models beer$ and $soda(state) \models soda$. When we look at each path we see that they hold in each case:

1. Let $trace(\pi_1) = (\emptyset, \{paid\}, \{paid, beer\})^\omega$

This implies that $\pi_1 \models \Box\Diamond beer$ since $beer(state) \models beer$ which satsifies $\pi_1 \models \Box\Diamond(soda \vee beer)$ because now and for all future points, there is a point further up in the future where $beer$ will hold.

1. Let $trace(\pi_2) = (\emptyset, \{paid\}, \{paid, soda\})^\omega$

This implies that $\pi_2 \models \Box\Diamond soda$ since $soda(state) \models soda$ which satsifies $\pi_2 \models \Box\Diamond(soda \vee beer)$ because now and for all future points, there is a point further up in the future where $soda$ will hold.

1. Let $trace(\pi_3) = ((\emptyset, \{paid\}, \{paid, soda\})^\omega, (\emptyset, \{paid\}, \{paid, beer\})^\omega)^\omega$

This shows that $\pi_3 \models \Box\Diamond soda\ or\ \pi_3 \models \Box\Diamond beer$ since $beer(state) \models beer$ and $soda(state) \models soda$ which satisfies that $\pi_3 \models \Box\Diamond(soda \vee beer)$ because now and for all future points, there is a point further up in the future where $soda \vee beer$ will hold.

1. Let $trace(\pi_4) = ((\emptyset, \{paid\}, \{paid, beer\})^\omega, (\emptyset, \{paid\}, \{paid, soda\})^\omega)^\omega$

This shows that $\pi_4 \models \Box\Diamond soda\ or\ \pi_4 \models \Box\Diamond beer$ since $beer(state) \models beer$ and $soda(state) \models soda$ which satisfies that $\pi_4 \models \Box\Diamond(soda \vee beer)$ because now and for all future points, there is a point further up in the future where $soda \vee beer$ will hold.

Given that $Paths(pay) = \{\pi_1, \pi_2, \pi_3, \pi_4\}, \forall\pi \in Paths(pay),\ pay \models \Box\Diamond soda\ or\ pay \models \Box\Diamond beer$. In otherwords, all paths starting from the intitial state pay satisfy the formula.

Thus we have proved, $TS \models \Box\Diamond(soda \vee beer)$

----

(2) $TS \not\models \Box soda$ iff for all initial states $pay \in I, pay \not\models \Box soda$, $\exists\pi \in Paths(pay)$. From our intuition we expect that this satisification will not hold. What we are going to do is find a path which does not satisfy the condition always soda (counter-example).

If we look to $\pi_1 = [pay, select, beer, pay, select, beer, \ldots] = [pay, select, beer]^\omega, trace(\pi_1) = (\emptyset, \{paid\}, \{paid, beer\})^\omega$ we notice that $pay \not\models soda, select \not\models soda, beer(state) \not\models soda$ thus it is definately the case that $\pi_1 \not\models \Box soda$. This is a counter-example which reftues the satisification relation and:

Thus we have proved, $TS \not\models \Box soda$

----

(3) $TS \not\models \Box\Diamond(soda \wedge beer)$ iff for all initial states $pay \in I, pay \not\models \Box\Diamond(soda \wedge beer)$, $\exists\pi \in Paths(pay)$. In this case, all we want to do is find a path starting from the initial state in which the satisfication relation does not hold. We already notice that there is no state that holds either of the propositions to be true, so it would be impossible that this transition system can satisfy this state.

We only need to find one path and use π_1 to demonstrate it as a counter-example:

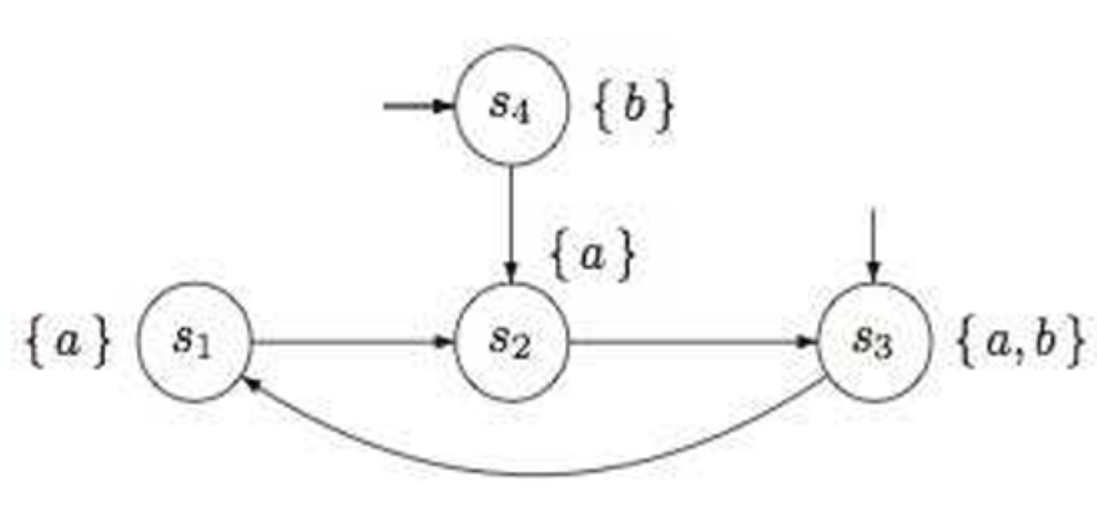
1. $\pi_1 = [pay, select, beer, pay, select, beer, \ldots] = [pay, select, beer]^\omega$
2. $trace(\pi_1) = (\emptyset, \{paid\}, \{paid, beer\})^\omega$

We notice that $pay \not\models soda \wedge beer, select \not\models soda \wedge beer, beer(state) \not\models soda \wedge beer$ thus we state that $\pi_1 \not\models \Box\Diamond(soda \wedge beer)$. This counter-example demonstrates that the satisification cannot hold and:

Thus we have proved, $TS \not\models \Box\Diamond(soda \wedge beer)$.

Question 2

Consider the following transition system TS of the simple beverage vending machine:



Decide for each of the following LTL formulas φ_i, whether $TS \models \varphi_i$. If $TS \models \varphi_i$, provide a proof; if $TS \not\models \varphi_i$, then provide a path π such that $\pi \not\models \varphi_i$.

There are two intial states I = {S4, S3}. A path of transition system TS is an initial, maximal path fragment. There are two paths on this transition system:

1. Let $Paths(S4) = \{\pi_1\}$
2. Let $Paths(S3) = \{\pi_2\}$

1. $\pi_1 = [S_4, S_2, S_3, S_1, S_2, S_3 \ldots] = [S_4, (S_2, S_3, S_1)^\omega]$
2. $\pi_2 = [S_3, S_1, S_2, S_3 \ldots] = [(S_3, S_1, S_2)^\omega]$

The traces of the path are the following:

1. $trace(\pi_1) = (\{b\}, (\{a\}, \{a, b\}, \{a\})^\omega$
2. $trace(\pi_2) = (\{a, b\}, \{a\}, \{a\})^\omega$

We must prove the following:

(1) $TS \models \Box\Diamond a$ iff for all initial states $s \in I, s \models \Box\Diamond a$, $\forall\pi \in Paths(s)$

This means that Always eventually a holds. In other words: Now and for all future points, there is a point further up in the future where a holds.

When we examine the traces we see indeed that it should satisfy the relation. We will move forward and prove that it is the case.

1. $trace(\pi_1) = (\{b\}, (\{a\}, \{a, b\}, \{a\})^\omega)$
2. $trace(\pi_2) = (\{a, b\}, \{a\}, \{a\})^\omega$

a. For $\pi_1 = [S_4, S_2, S_3, S_1, S_2, S_3 \ldots] = [S_4, (S_2, S_3, S_1)^\omega]$

$S_4 \not\models a$, but $S_2 \models a$, $S_3 \models a \wedge b$, $S_1 \models a$ thus $\pi_1 \models \Box\Diamond a$ this is because there will always be a point further up in time where a will hold and this will continue forever.

b. For $\pi_2 = [S_3, S_1, S_2, S_3 \ldots] = [(S_3, S_1, S_2)^\omega]$

$S_3 \models a \wedge b$, $S_1 \models a$, $S_2 \models a$ thus $\pi_2 \models \Box\Diamond a$, this is because there will always be a point further up in time where a will hold and this will continue forever.

We have shown for all paths that the satisftication relation holds and thus:

$TS \models \Box\Diamond a$

----

(2) $TS \models \circ a$ iff for all initial states $s \in I, s \models \circ a$, $\forall\pi \in Paths(s)$

This means that the on each of the paths next step must be a.

When we examine the traces we see indeed that it should satisfy the relation. We will move forward and prove that it is the case.

1. $trace(\pi_1) = (\{b\}, (\{a\}, \{a, b\}, \{a\})^\omega)$
2. $trace(\pi_2) = (\{a, b\}, \{a\}, \{a\})^\omega$

a. For $\pi_1 = [S_4, S_2, S_3, S_1, S_2, S_3 \ldots] = [S_4, (S_2, S_3, S_1)^\omega]$

$S_2 \models a$, thus $\pi_1 \models \circ a$ as S_2 is the next step in the path.

b. For $\pi_2 = [S_3, S_1, S_2, S_3 \ldots] = [(S_3, S_1, S_2)^\omega]$

$S_1 \models a$,thus $\pi_2 \models \circ a$ as S_1 is the next step in the path.

We have shown for all paths that the satisftication relation holds and thus:

$TS \models \circ a$

----

(3) $TS \not\models \circ(a \rightarrow \Box(a \wedge b))$ iff for all initial states $s \in I, s \not\models \circ(a \rightarrow \Box(a \wedge b)), \exists\pi \in Paths(s)$

a. For $\pi_1 = [S_4, S_2, S_3, S_1, S_2, S_3 \ldots] = [S_4, (S_2, S_3, S_1)^\omega]$

We have to prove that $\pi_1^1 \not\models (a \rightarrow \Box(a \wedge b)$ because this is the next step. And since $S_2 \models a$ it implies that $\pi_1^1 \models a$ and after this we test the implication and we notice that $S_3 \models a \wedge b$, but $S_1 \not\models a \wedge b$ and $S_2 \not\models a \wedge b$ and it is impossible the implication $\Box(a \wedge b)$ can be satisifed which means that $\pi_1^1 \not\models (a \rightarrow \Box(a \wedge b))$

We have shown a counter-example for a single path in which the satisfication relation does not hold and thus:

$TS \not\models \circ(a \rightarrow \Box(a \wedge b))$

----

(4)$TS \models \Diamond(a \cup b)$ iff for all initial states $s \in I, s \models \Diamond(a \cup b), \forall\pi \in Paths(s)$

This means eventually, it will reach a state in which a will hold until b hold.

When we examine the traces we see indeed that it should satisfy the relation. We will move forward and prove that it is the case.

1. $trace(\pi_1) = (\{b\}, (\{a\}, \{a, b\}, \{a\})^\omega)$
2. $trace(\pi_2) = (\{a, b\}, \{a\}, \{a\})^\omega$

a. For $\pi_1 = [S_4, S_2, S_3, S_1, S_2, S_3 \ldots] = [S_4, (S_2, S_3, S_1)^\omega]$

We see that $S_4 \models b$, and we can refer to the formula for equivalence, and note that $\phi \cup \psi \equiv \psi \vee (\phi \wedge \circ(\phi \cup \psi))$, thus S_4 satisfies this state. Thus, $\pi_1 \models \Diamond(a \cup b)$, since eventually is defined as now or at some future point in time and $\pi_1^0 \models (a \cup b)$ because $S_4 \models b$.

b. For $\pi_2 = [S_3, S_1, S_2, S_3 \ldots] = [(S_3, S_1, S_2)^\omega]$

We see that, $S_3 \models a \wedge b$ and we can refer to the formula for equivalence, and note that $\phi \cup \psi \equiv \psi \vee (\phi \wedge \circ(\phi \cup \psi))$, since $S_3 \models b$,thus S_3 satisfies this state. Thus, $\pi_2 \models \Diamond(a \cup b)$, since eventually is defined as now or at some future point in time and $\pi_2^0 \models (a \cup b)$ because $S_3 \models b$.

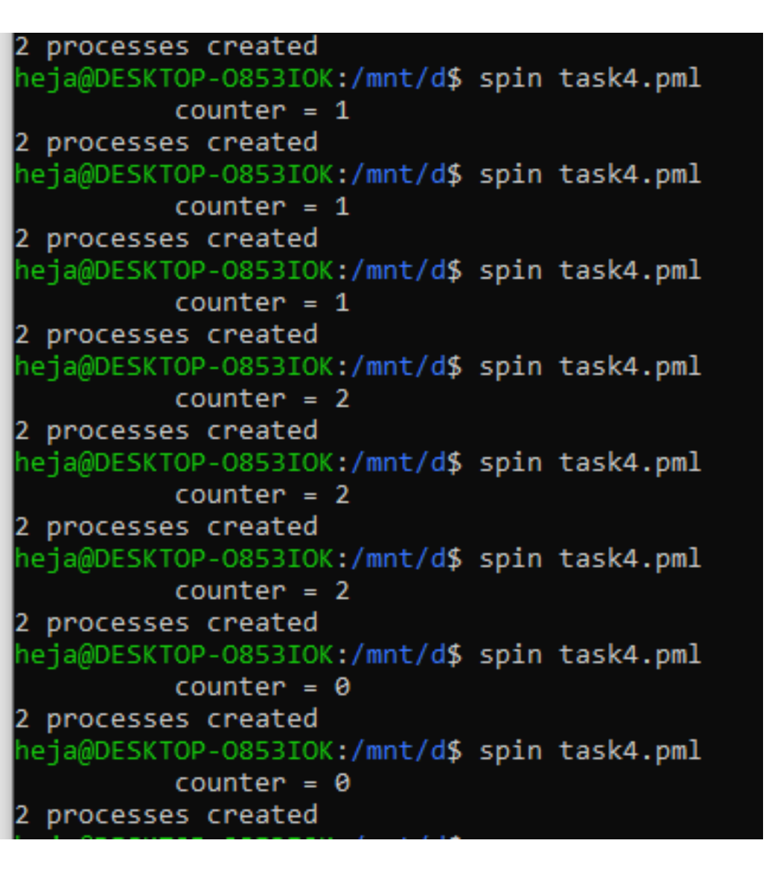
We have shown for all paths that the satisftication relation holds and thus:

$TS \models \Diamond(a \cup b)$

Question 4

When we are looking at the code, x is 1 to begin with and we notice that after the guard x !=0, the value x after the if statements are either added, subtracted or breaks randomly. This could lead to an execution path where the value = 0. We can test this a few ways, firstly we kept re-running the program and found that our intution was correct and there was a path where the value of x = 0. And then we used an assertion at the bottom of the program and verified this using spin. These are detailed below.

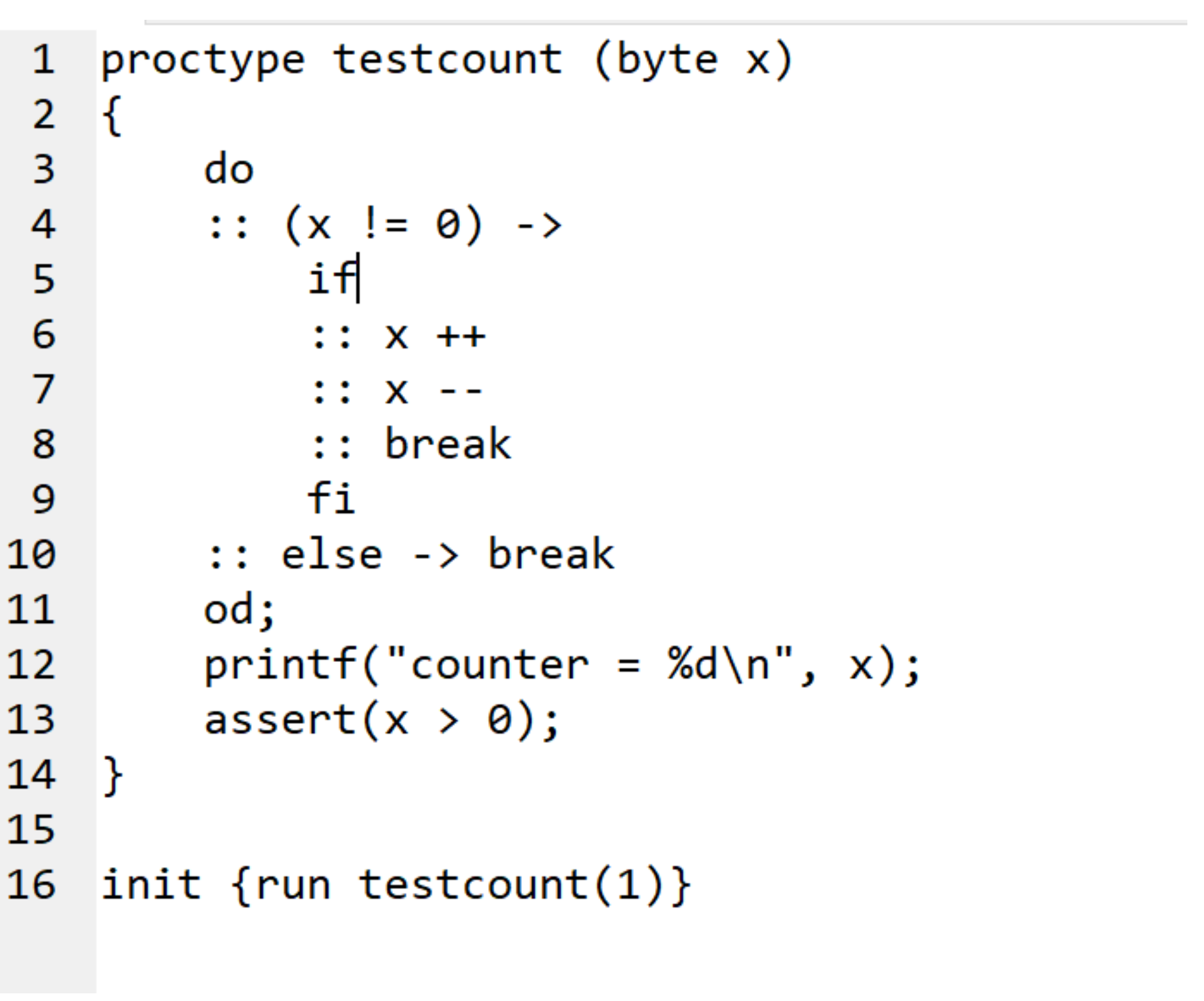
(1) Continue re-running the program and see if there is an execution path which leads to the value being 0.



```
2 processes created
heja@DESKTOP-0853IOK:/mnt/d$ spin task4.pml
        counter = 1
2 processes created
heja@DESKTOP-0853IOK:/mnt/d$ spin task4.pml
        counter = 1
2 processes created
heja@DESKTOP-0853IOK:/mnt/d$ spin task4.pml
        counter = 1
2 processes created
heja@DESKTOP-0853IOK:/mnt/d$ spin task4.pml
        counter = 2
2 processes created
heja@DESKTOP-0853IOK:/mnt/d$ spin task4.pml
        counter = 2
2 processes created
heja@DESKTOP-0853IOK:/mnt/d$ spin task4.pml
        counter = 2
2 processes created
heja@DESKTOP-0853IOK:/mnt/d$ spin task4.pml
        counter = 0
2 processes created
heja@DESKTOP-0853IOK:/mnt/d$ spin task4.pml
        counter = 0
2 processes created
```

We see that it does not hold for all execution paths.

(2) We can add an assertion function and verify that the value is greater than 0. This is more of an ideal way of doing it since for more complicated programs it would be too difficult to re-run the program and find an execution path which violates the assertion.



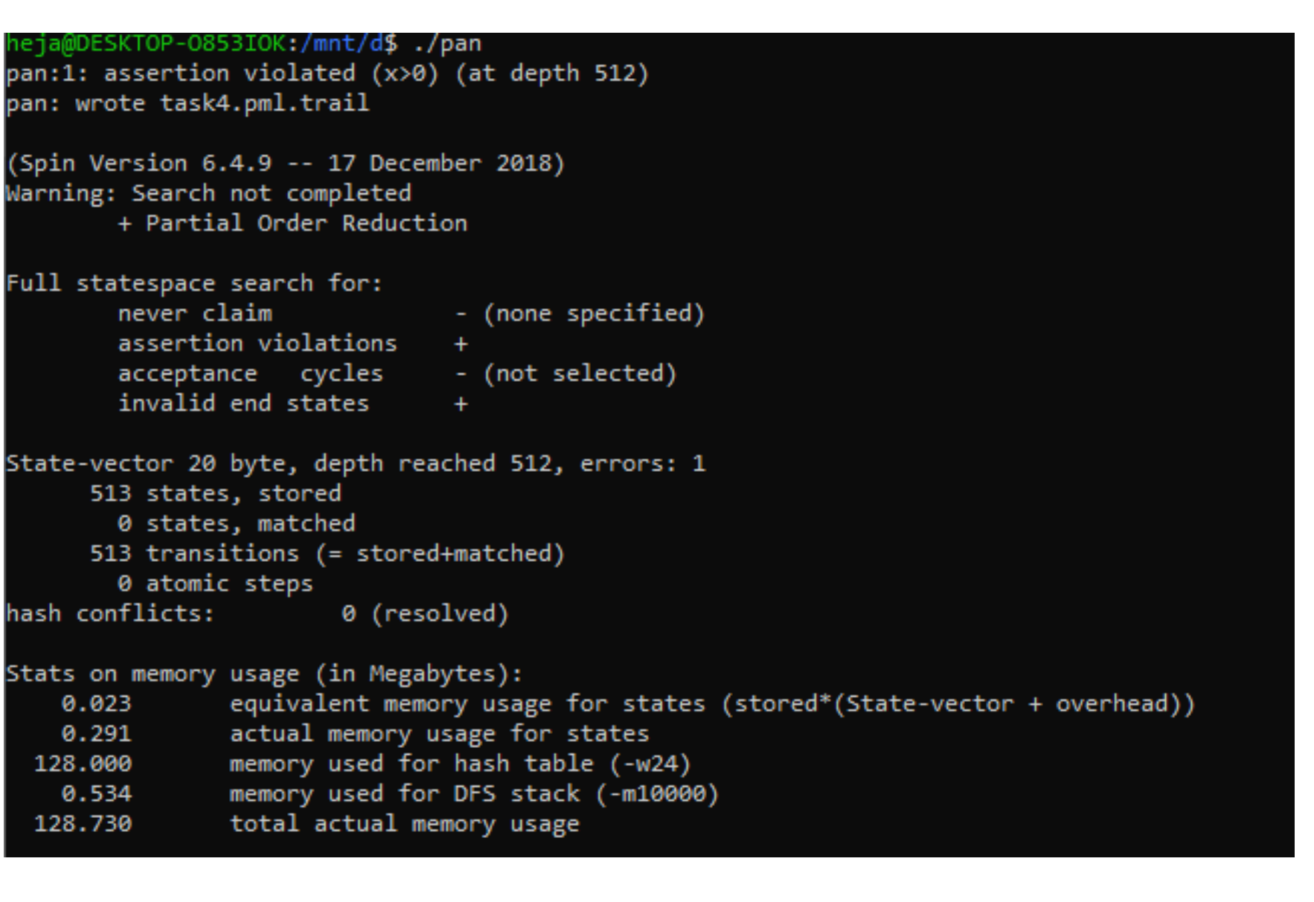
```
proctype testcount (byte x)
{
    do
    :: (x != 0) ->
        if
        :: x ++
        :: x --
        :: break
        fi
    :: else -> break
    od;
    printf("counter = %d\n", x);
    assert(x > 0);
}

init {run testcount(1)}
```

We can then verify the above program with the following commands:

a. spin -a task4.pml

b. gcc ./pan.c -o pan

c. ./pan



```
heja@DESKTOP-0853IOK:/mnt/d$ ./pan
pan:1: assertion violated (x>0) (at depth 512)
pan: wrote task4.pml.trail

(Spin Version 6.4.9 -- 17 December 2018)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        acceptance   cycles     - (not selected)
        invalid end states      +

State-vector 20 byte, depth reached 512, errors: 1
      513 states, stored
        0 states, matched
      513 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.023       equivalent memory usage for states (stored*(State-vector + overhead))
    0.291       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.730       total actual memory usage
```

We notice the following statement:

pan:1: assertion violated (x>0) (at depth 512)

Indicating that the value of x is not always greater than 0, as we have demonstrated earlier.