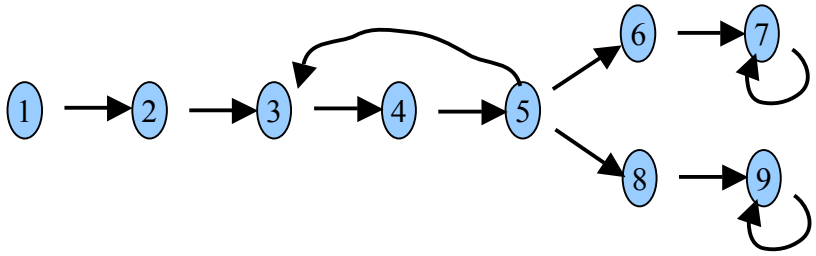
Exercise 4

Model Checking with Fairness Constraints and Construction of Counterexample/Witness

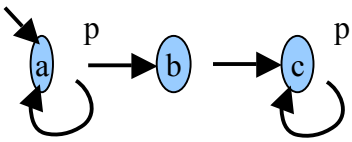
1) For the structure below, compute the set of fair states assuming that the two fairness constraints are as follows: $c_1=\{7\}$ and $c_2=\{4,9\}$.



Solution for 1)
fair = nu Z. EX((E Z U Z ∧ c1) ∧ (E Z U Z ∧ c2)) =
=nu Z. EX((mu Y. (Z ∧ c1) ∨ (Z ∧ EX(Y))) ∧ (mu Y. (Z ∧ c1) ∨ (Z ∧ EX(Y))))

Z0={1-9}
Y0={}, Y1={7}, Y2={6,7}, Y3={5,6,7}, Y4={4,5,6,7}, ... Y7={1-7}=Y8
Y0={}, Y1={4,9},Y2={3,4,8,9},Y3={2,3,4,5,8,9}, Y4={1,2,3,4,5,8,9}=Y5
Z1=EX{1-5}={1-5}
Y0={}=Y1
Y0={}, Y1={4}, Y2={3,4}, Y3={2,3,4,5}, Y4={1-5}=Y5
Z2=EX{}={}=fair

2) For the Kripke structure depicted below, assume that the fairness condition ¬a is given. Check whether, with this addition, K,a ⊨ AFAG p holds.



Solution for 2)
AFAG p = ! EG EF !p
[!p]={b}
EcF{b}=EF {b} (if b is fair)
Is b fair?
fair = nu Z.EX(E Z U Z ∧ !a), [!a]={b,c}
Z0={a,b,c}
Y0={}, Y1={b,c}, Y2={a,b,c}
Z1=EX{a,b,c}={a,b,c}
Yes, b is fair.

EF{b}= mu Y. {b} ∨ EX Y
Y0={},Y1={b},Y2={a,b}=Y3
EcG{a,b}=mu Z. {a,b} ∧ EX(nuY. (Z ∧ {b,c}) ∨ (Z ∧ EX Y))
Z0={a,b,c}
Y0={}, Y1={b,c}, Y2={a,b,c}

Z1={a,b}/\EX{a,b,c}={a,b}
Y0={}, Y1={b}, Y2={a,b}
Z2={a,b}/\EX{a,b}={a}
Y0={},Y1={}=Y0
Z3={}
So, [[EGEF !p]]={}, and therefore [[AFAG p]]={a,b,c} and so K models AFAG p.

3) Use VIS with the two models of the dining philosophers introduces in the lecture and check for
   - Mutual exclusion
   - Progress/deadlock
   - Starvation

   without fairness. Give a counterexample if the property fails, i.e., ask VIS for a counterexample using the command: `model_check -d 1 filename`.
   Construct meaningful fairness conditions and show that the second version satisfies all properties under fairness.

   Solution for 3)

   There are several ways to write these properties. Here are my versions.

   Properties:
   Mutual exclusion:

```
AG((!(s0=EATING)) + !(s1=EATING));
AG((!(s1=EATING)) + !(s2=EATING));
AG((!(s2=EATING)) + !(s3=EATING));
AG((!(s3=EATING)) + !(s0=EATING));
```

   These four properties hold in dining-v1.v and dining-v2.v.

   No deadlock:

```
AG( (s0=HUNGRY * s1=HUNGRY * s2=HUNGRY * s3=HUNGRY) ->
AF(!(s0=HUNGRY * s1=HUNGRY * s2=HUNGRY * s3=HUNGRY)));
```

   This property holds in dining-v2.v but is violated in dining-v1.v (with and without fairness).

   No Starvation:

```
AG(s0=HUNGRY -> AF(s0=EATING));
AG(s1=HUNGRY -> AF(s1=EATING));
AG(s2=HUNGRY -> AF(s2=EATING));
AG(s3=HUNGRY -> AF(s3=EATING));
```

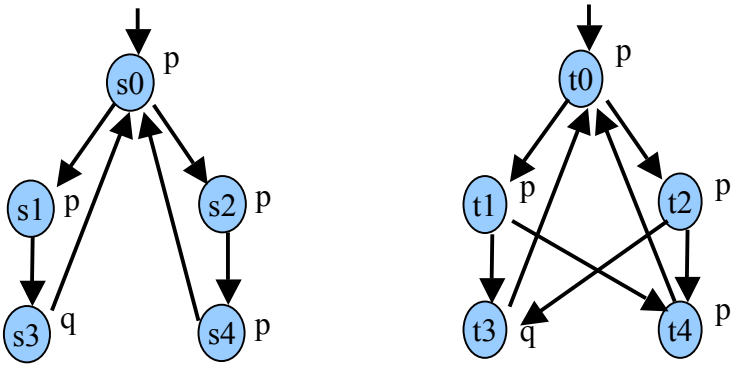
   These four properties hold only in dining-v2.v under the fairness conditions listed below.
   (In dining-v1.v they are always violated.)

   File with four fairness constraints

```
# no philosophers is eating forever
! (s0=EATING);
! (s1=EATING);
! (s2=EATING);
! (s3=EATING);
```

Bisimulation, Simulation, and Linear Temporal Logic

4) Check if the following two structures are (i) bisimilar or/and (ii) simulation equivalent.



Solution to 4)

Bisim	t0	t1	t2	t3	t4
s0	F	F	F	F	F
s1	F	F	F	F	F
s2	F	F	F	F	F
s3	F	F	F	F	F
s4	F	F	F	F	F

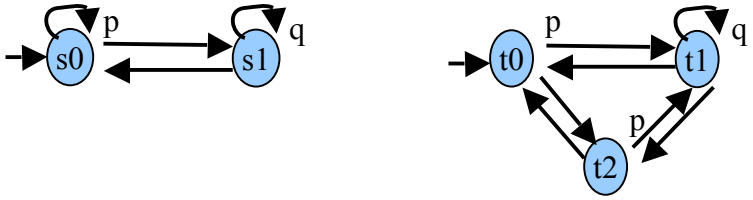
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6

s < t	t0	t1	t2	t3	t4
s0		F	F	F	F
s1	F			F	F
s2	F			F	F
s3	F	F	F		F
s4	F	F	F	F	

t < s	s0	s1	s2	s3	s4
t0	F	F	F	F	F
t1	F	F	F	F	F
t2	F	F	F	F	F
t3	F	F	F	F	F
t4	F	F	F	F	F

So, the two structures are neither bisimilar nor simulation equivalent but the right structure simulates the left one.

5) Check if the following two structures are (i) bisimilar or/and (ii) simulation equivalent.



The two structure are bisimilar (and therefore simulation equivalent) due to the following bisimulation relation: {(s0,t0),(s0,t2),(s1,t1)}

6) Consider the dining philosophers from the last lecture. Write LTL properties for
   1. Mutual exclusion
   2. Progress/deadlock
   3. Starvation

Use VIS with the two models introduces in the last lecture (Slide 9 and 11) and check if the models satisfy your properties. The LTL model checker of VIS is called with

```
ltl_model_check filename
```

Don't forget to read-in the verilog file first and build the transition graph:

```
read_verilog filename
init_verify
```

Write an LTL property that is true for all paths on which none of the philosophers is eating forever. Use this property and our previous properties to show that the model v2 is starvation free under the assumption that none of the philosophers is eating forever.

Solution of 6)
Properties:
Mutual exclusion:

```
G((!(s0=EATING)) + !(s1=EATING));
G((!(s1=EATING)) + !(s2=EATING));
G((!(s2=EATING)) + !(s3=EATING));
G((!(s3=EATING)) + !(s0=EATING));
```

These four properties hold in dining-v1.v and dining-v2.v.

No deadlock:

```
G(F(!(s0=HUNGRY * s1=HUNGRY * s2=HUNGRY * s3=HUNGRY)));
```

This property holds in dining-v2.v but is violated in dining-v1.v

No Starvation:

```
G(s0=HUNGRY -> F(s0=EATING));
G(s1=HUNGRY -> F(s1=EATING));
G(s2=HUNGRY -> F(s2=EATING));
G(s3=HUNGRY -> F(s3=EATING));
```

These four properties are violated in both versions.

No philosophers is eating forever:

```
G(F(!s0=eating)) * G(F(!s1=eating)) * G(F(!s2=eating)) * G(F(!s3=eating));
```

Version-v2 satisfies the following properties:

```
(G(F(!s0=eating)) * G(F(!s1=eating)) * G(F(!s2=eating)) * G(F(!s3=eating)))->
G(s0=HUNGRY -> F(s0=EATING));
(G(F(!s0=eating)) * G(F(!s1=eating)) * G(F(!s2=eating)) * G(F(!s3=eating))->
G(s1=HUNGRY -> F(s1=EATING));
(G(F(!s0=eating)) * G(F(!s1=eating)) * G(F(!s2=eating)) * G(F(!s3=eating))->
G(s2=HUNGRY -> F(s2=EATING));
(G(F(!s0=eating)) * G(F(!s1=eating)) * G(F(!s2=eating)) * G(F(!s3=eating))->
G(s3=HUNGRY -> F(s3=EATING));
```