Examples for the second midterm: Software model checking with abstraction. Modeling with Petri nets. Properties of Petri nets.

Ákos Hajdu, István Majzik
BME Department of Measurement and Information Systems

Software model checking with abstraction

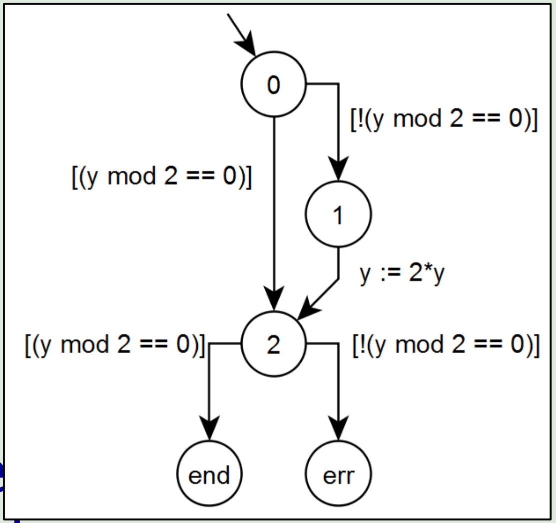
```
      y : int
0:    if !((y mod 2) ==0) {
1:          y := 2*y;
      }
2:    assert((y mod 2) == 0);
```

- Draw the Control Flow Automaton (CFA) corresponding to the code!
  - Use the number of the lines (0, 1, 2) for the locations!
  - Represent assertion violations with a location labeled err.
  - Represent the normal ending of the program with a location labeled end.
- We are using location and predicate abstraction with a single predicate *(y > 0)* for model checking.
  What are the possible initial states in the abstract state space if the value of *y* can be arbitrary in the beginning? Give them in the following form: (location, predicates)!
- Is the abstract path *(0, true)* → *(2, true)* → *(err, true)* a real or a spurious counterexample? Explain your answer!

Software model checking with abstraction: solution

- Draw the CFA!
  - Solution: to the right.

```
        y : int
0:      if !((y mod 2) ==0) {
1:              y := 2*y;
        }
2:      assert((y mod 2) == 0);
```



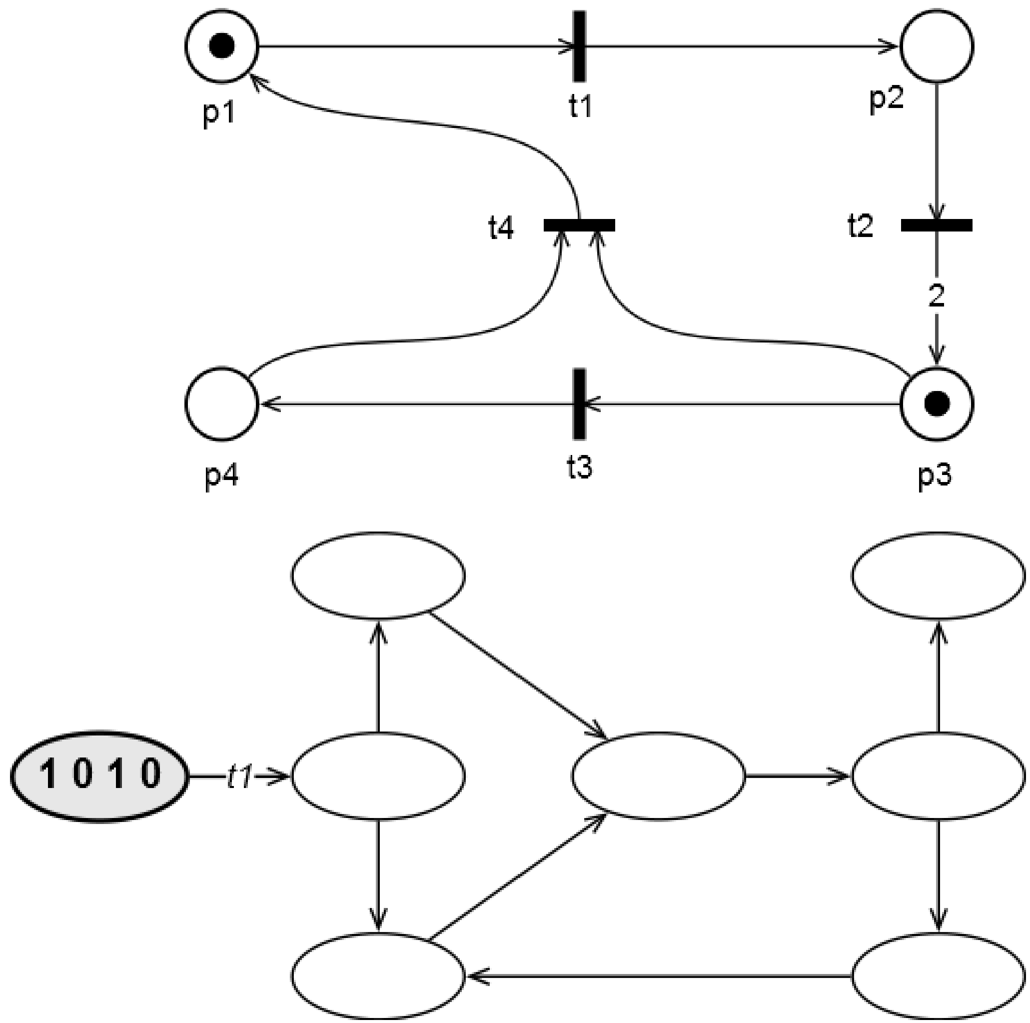
- We are using location and predicate abstraction with a single predicate *(y > 0)* for model checking. What are the possible initial states in the abstract state space if the value of *y* can be arbitrary in the beginning?
  - Solution: (0, true) and (0, false)
- Is the abstract path *(0, true)* → *(2, true)* → *(err, true)* a real or a spurious counterexample?
  - Solution: *(0, true)* → *(2, true)* transition requires (y>0) and (y mod 2 == 0) to hold. The former is the source state predicate, the latter is the transition. The transition *(2, true)* → *(err, true)* requires (y>0) and !(y mod 2 == 0) to hold, which contradicts the previous conditions.

State space of a Petri net

- Complete the reachability graph of the Petri net!

  The graph is missing edges, edge labels and state labels.

  The initial state is marked with a gray background.

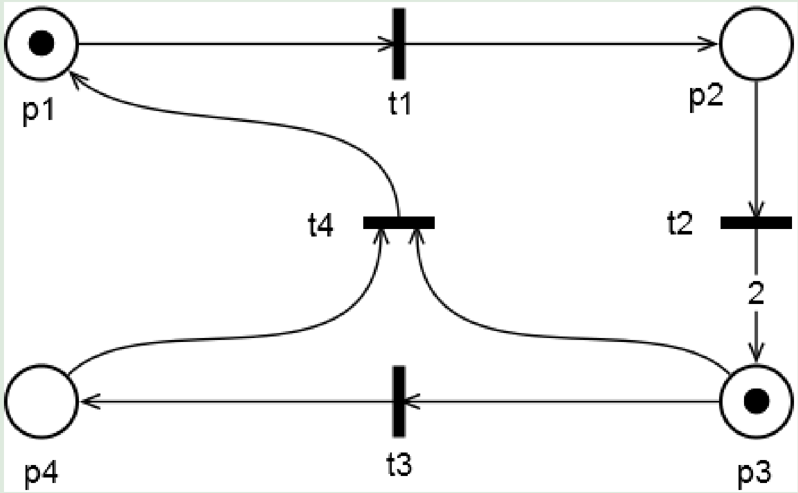


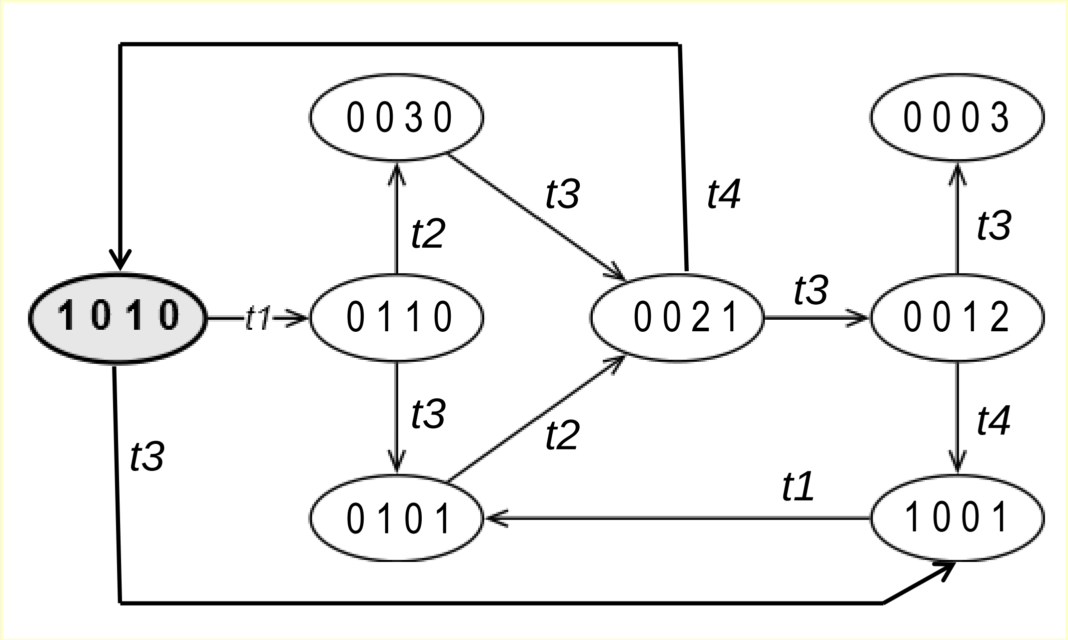
State space of a Petri net: solution

- Complete the reachability graph of the Petri net!

  The graph is missing edges, edge labels and state labels.

  The initial state is marked with a gray background.





Dynamic properties

- Observe the Petri net and its reachability graph from the previous example and decide if the following statements are true, false or not decidable!

(a) The reachability and coverability graphs of the net are identical

(b) The net is not persistent

(c) The net has a deadlock

(d) Transition t3 is L2-live

(e) The sequence t1, t2, t4 is a T-invariant

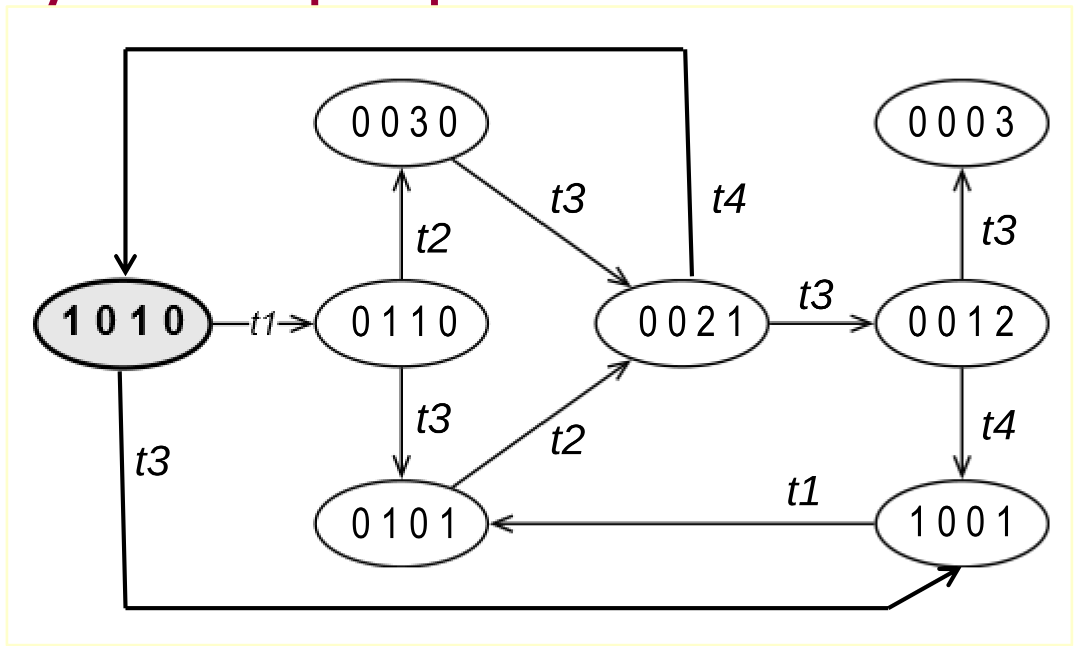
(f) Transitions t3 and t4 are bounded fair

(g) The state (0 1 0 1) is a home state

(h) The net is globally fair

Dynamic properties

The state space:



The properties:

(a) The reachability and coverability graphs of the net are identical

(b) The net is not persistent

(c) The net has a deadlock

(d) Transition t3 is L2-live

(e) The sequence t1, t2, t4 is a T-invariant

(f) Transitions t3 and t4 are bounded fair

(g) The state (0 1 0 1) is a home state

(h) The net is globally fair

Dynamic properties: solution

(a) The reachability and coverability graphs of the net are identical

(b) The net is not persistent

(c) The net has a deadlock

(d) Transition t3 is L2-live

(e) The sequence t1, t2, t4 is a T-invariant

(f) Transitions t3 and t4 are bounded fair

(g) The state (0 1 0 1) is a home state

(h) The net is globally fair

a) True, the net is bounded.
b) True, e.g., t3 and t4 for state (0 0 1 2).
c) True, e.g., state (0 0 0 3)
d) True, there is a loop containing t3.
e) False, there is no such loop.
f) True, because they do not appear in loops without eachother.
g) False, it cannot be reached from (0 0 0 3).
h) True, because each infinite sequence contains every transition.

Dynamic properties (1/2)

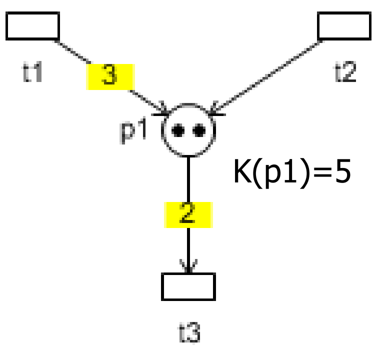
- Boundedness:
  - Finite reachability graph, no ω symbol in the coverability graph
  - Safeness: reachability graph only contains 0 or 1 in markings
- Reversibility:
  - The reachability graph is a single strongly connected component
- Home state:
  - The reachability graph contains a strongly connected component including the given state
- Fairness:
  - „One transition can only fire a finite number of times before the other fires.”:
    Counterexample: A loop with one of the transitions, excluding the other
- Persistency:
  - „Enabled transitions remain enabled until fired.”:
    Counterexample: Multiple transitions are enabled, but if we do not fire a given transition, it will not be enabled in the next state(s).

Dynamic properties (2/2)

- Transition L1, L2, L3-liveness
  - Enough to find a trajectory where it holds
- Transition L4-liveness
  - Check if it eventually becomes enabled from every state
- Net liveness
  - The net is live if every transition is L4-live
  - Enough to find a transition which is not L4-live
  - Deadlock freedom does not mean L-live

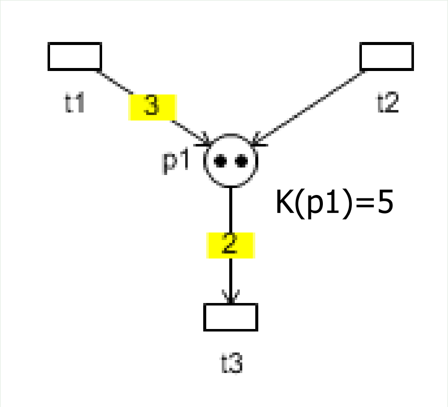
Petri nets with capacities

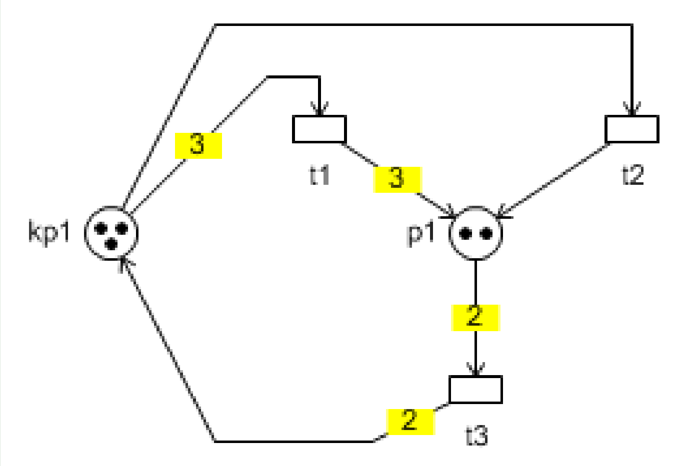
- What does it mean in a Petri net if a place has finite capacity?
- Draw an equivalent Petri net without using capacities!



Petri nets with capacities: solution

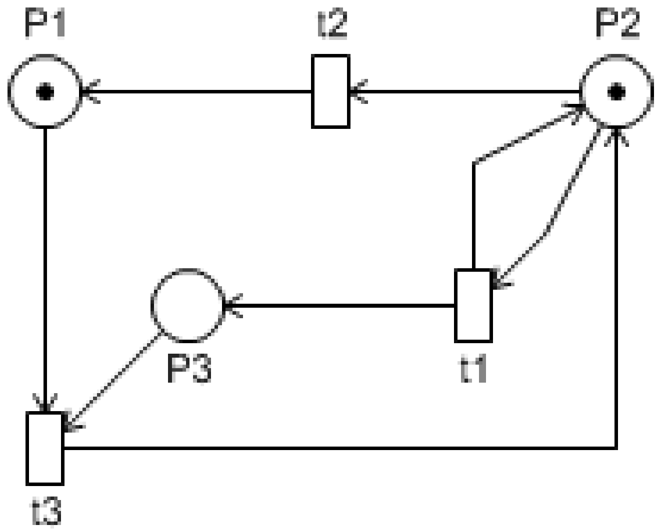
- What does it mean in a Petri net if a place has finite capacity?
  - The number of tokens cannot be greater in a place than its capacity.
- Draw an equivalent Petri net without using capacities!
  - Supplementary place kp1 to keep track of the free capacity.





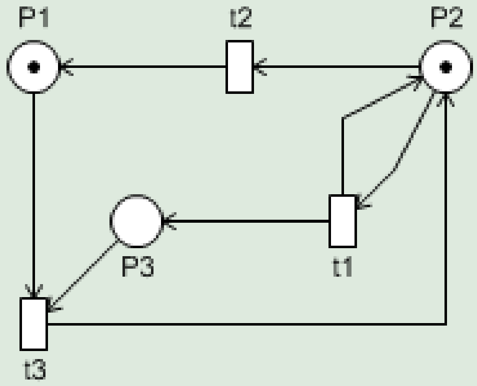
Coverability graph

- Draw the coverability graph for the following Petri net!
- How does the graph change if place P2 has a finite capacity of 1?

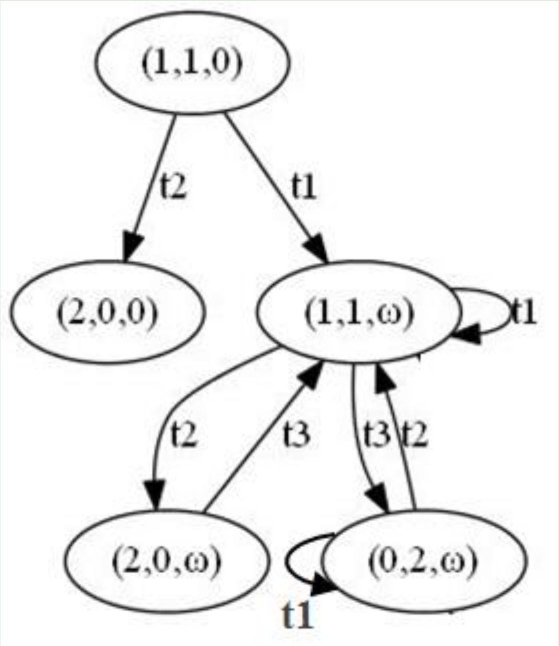


Coverability graph: solution

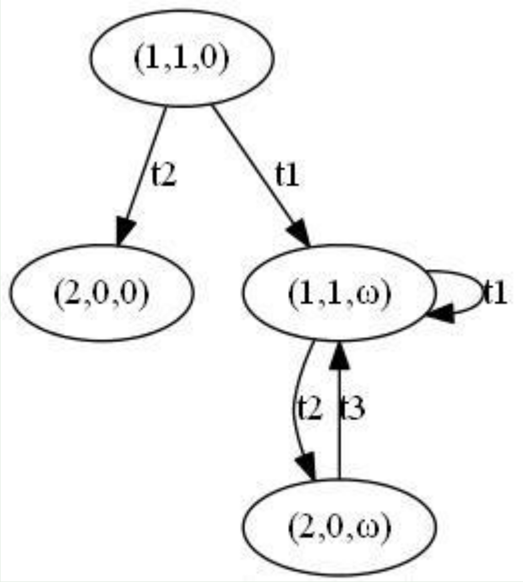
- Draw the coverability graph for the following Petri net!



Without capacity (P1, P2, P3):



With capacity (P1, P2, P3):



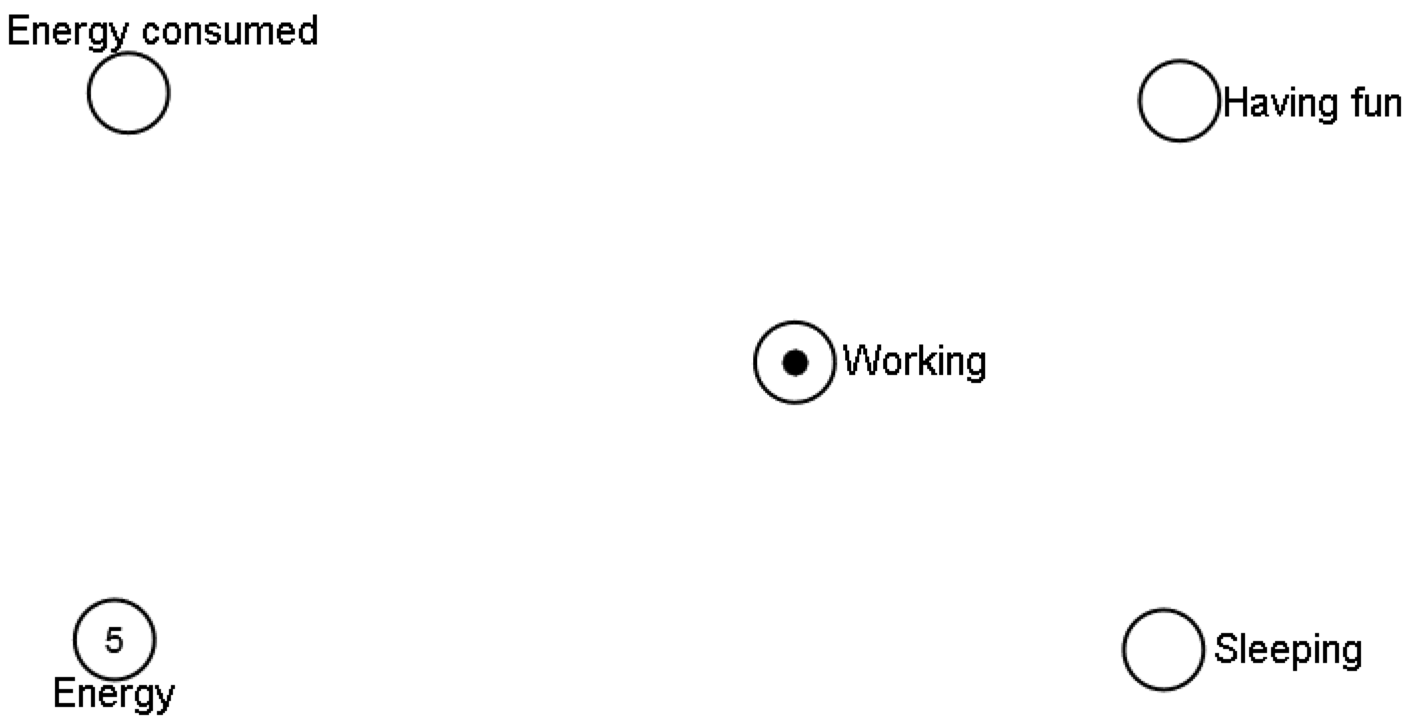
Modeling with Petri nets(1/2)

Create a non-colored Petri net model corresponding to a programmer based on the following description!

1. The programmer is working, sleeping or having fun.
2. The programmer has 5 units of energy for each day, and starts with work (this is the initial state).
3. If the programmer is working or having fun, he/she occasionally consumes a unit of energy.
4. If the programmer is working and has consumed at least 3 units of energy, he/she can start having fun.
5. If the programmer is having fun and has no energy left, he/she can start sleeping.
6. If the programmer is sleeping, one unit of energy is regained occasionally.
7. If the programmer is sleeping and all energy is available, he/she can start working.

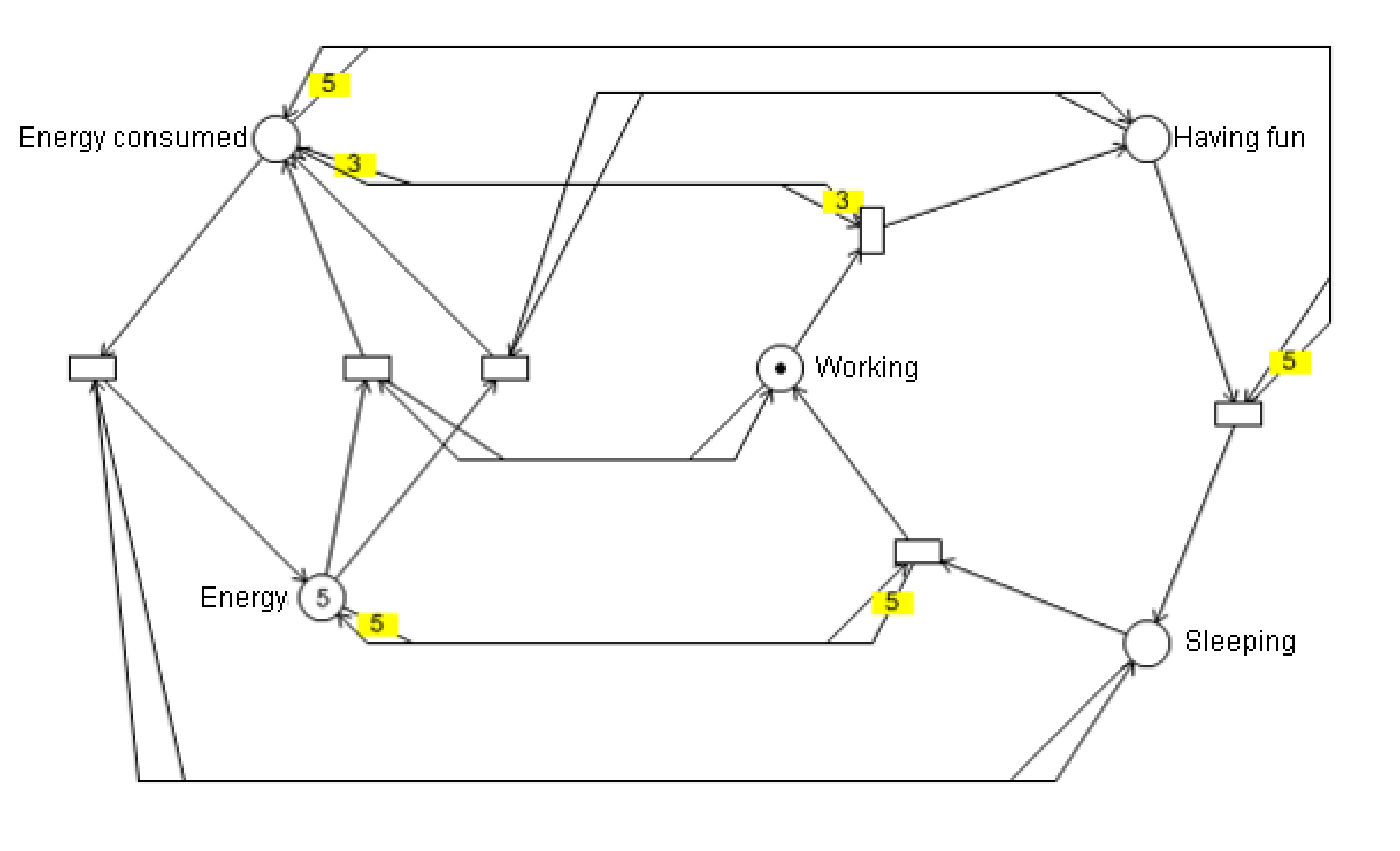
Modeling with Petri nets (2/2)

Extend the partial model below!



Modeling with Petri nets: solution

Extend the partial model below!

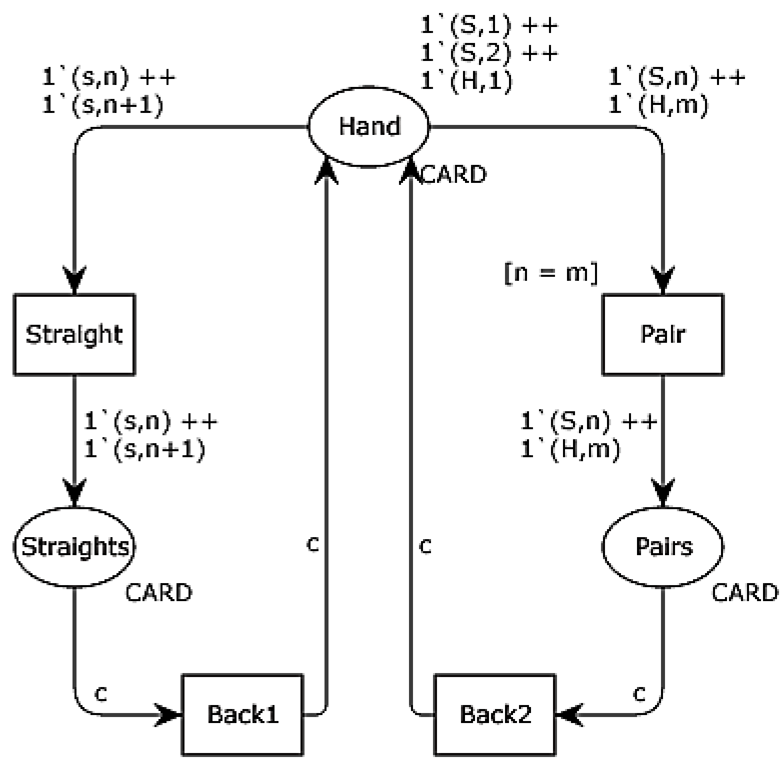


Modeling with colored Petri nets

The Petri net on the right is given with its definition block.

1. Enumerate enabled transitions with bindings under the given marking!
2. What are the possible markings after firing?
   Select one of the possible markings and enumerate the enabled bindings!
3. Is the net bounded with the given state?
4. Does the net have deadlock with the given state?
5. Is there a T-invariant in the net?

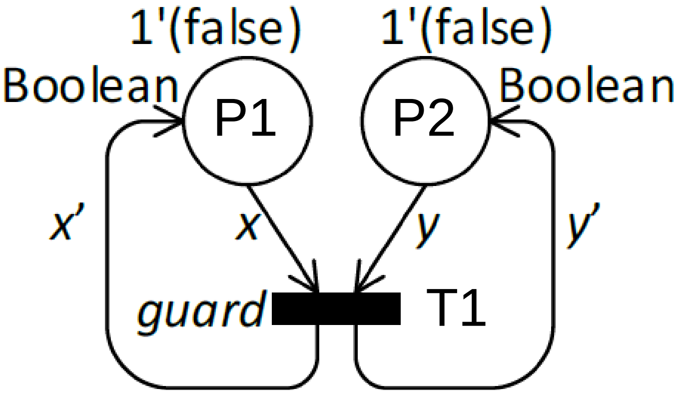
```
colset SUIT = with S | H;
colset NUM = int with 0..12;
colset CARD = product SUIT * NUM;
var s : SUIT;
var n, m : NUM;
var c : CARD;
```



Modeling with colored Petri nets: solution

1. Enabled:
   - **Straight**, s=S, n=1 binding
   - **Pair**, n=1, m=1 binding
2. Next markings:
   - **Straight** fires: **Hand** will have 1’(H,1), **Straights** will have 1’(S,1)++1’(S,2). Enabled: **Back1**, c=(S,1) or c=(S,2) binding
   - **Pair** fires: **Hand** will have 1’(S,2), **Pairs** will have 1’(S,1)++1’(H,1). Enabled: **Back2**, c=(S,1) or c=(H,1) binding
3. Bounded:
   - Each transition consumes/produces the same number of tokens, therefore the number of tokens does not change
4. Deadlock-free:
   - Transitions **Back1** or **Back2** can always put back the tokens consumed by **Straight** or **Pair**, making a cyclical behavior
5. T-invariants:
   - **Straight, Back1, Back1**
   - **Pair, Back2, Back2**

Unfolding colored Petri nets



- The Petri net in the right is given with the following definitions:
  var x, y, x',y': Boolean;

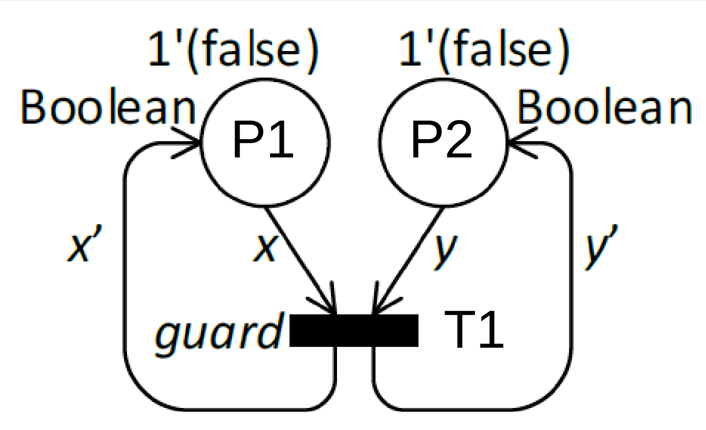
  The guard is the following:
  $(\neg x \wedge \neg y \wedge x' \wedge \neg y') \vee (x \wedge \neg y \wedge \neg x' \wedge y') \vee (\neg x \wedge y \wedge x' \wedge y')$
- Draw the equivalent non-colored Petri net, that is, the unfolding of the colored net!
- Is the colored Petri net and the unfolded non-colored net live and/or bounded with the given (or any) initial marking?

Unfolding colored Petri nets: solution

- The Petri net in the right is given with the following definitions :

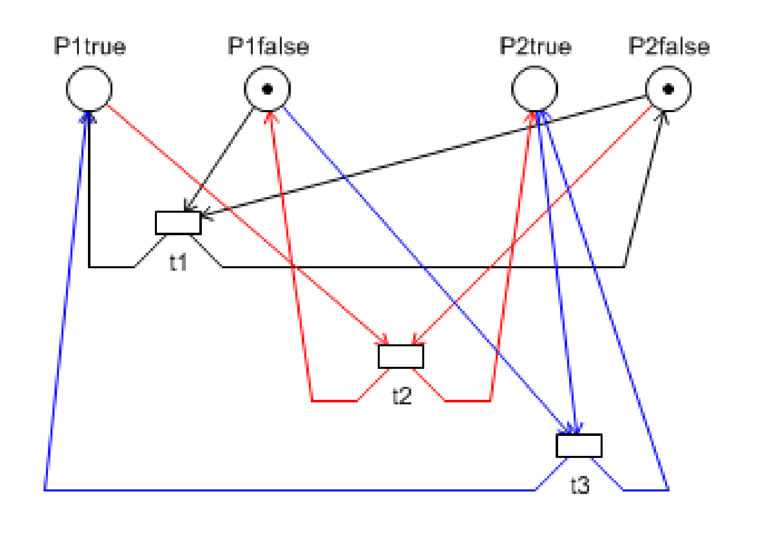
  var x, y, x’,y’: Boolean;

  The guard:



$$(\neg x \wedge \neg y \wedge x' \wedge \neg y') \vee (x \wedge \neg y \wedge \neg x' \wedge y') \vee (\neg x \wedge y \wedge x' \wedge y')$$

- Unfolding the net:
- Is the net live and/or bounded?
  - Not live (has deadlock)
  - Bounded: No transition can produce more tokens than it consumes

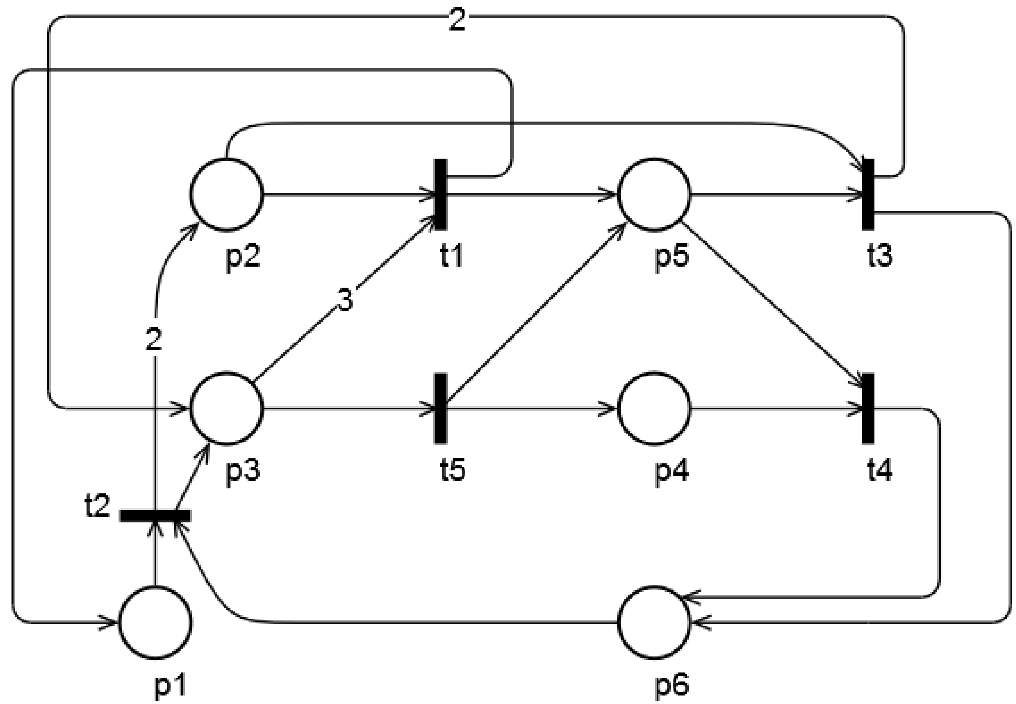


Structural properties (1)

The Petri net is given with its incidence matrix W^T where some elements are missing, denoted by letters.

Which numbers should replace the letters?

- A =
- B =
- C =
- D =



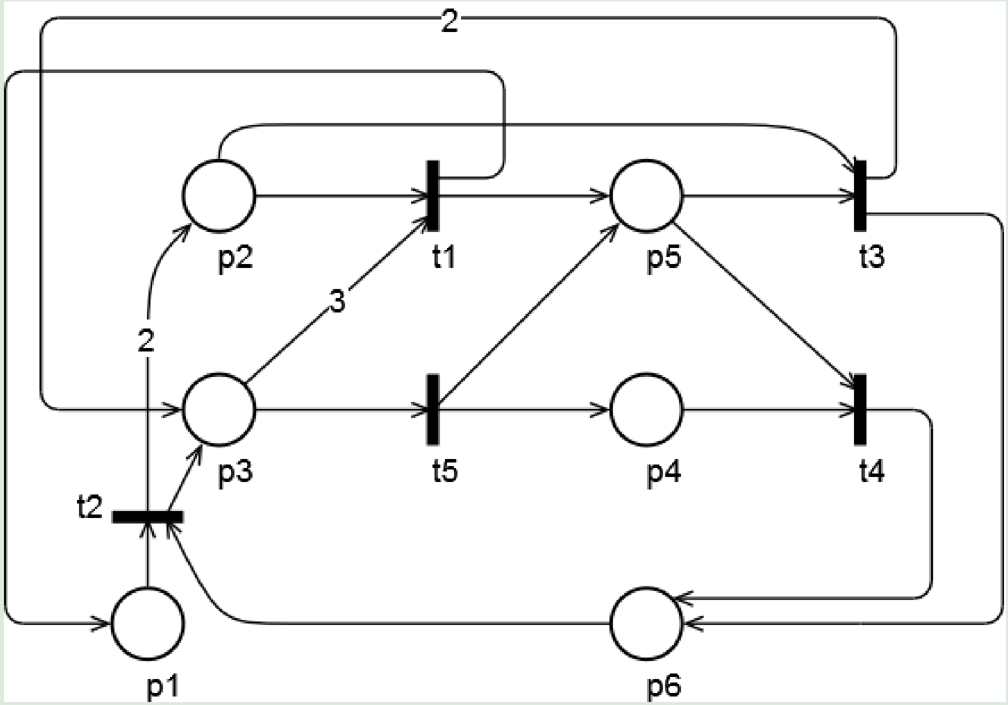
$$\mathbf{W}^T = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{array} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \hspace{-10em} \begin{array}{c} \\ \left[\begin{array}{ccccc} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & \mathbf{B} & 0 & 0 \\ \mathbf{A} & 1 & 2 & 0 & -1 \\ 0 & 0 & 0 & \mathbf{C} & 1 \\ 1 & 0 & -1 & -1 & 1 \\ 0 & -1 & 1 & 1 & \mathbf{D} \end{array}\right] \end{array}$$

Structural properties (1): solution

The Petri net is given with its incidence matrix W^T where some elements are missing, denoted by letters.

Which numbers should replace the letters?

- A = -3
- B = -1
- C = -1
- D = 0



$$W^T = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{array} \begin{bmatrix} t_1 & t_2 & t_3 & t_4 & t_5 \\ 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & \mathbf{B} & 0 & 0 \\ \mathbf{A} & 1 & 2 & 0 & -1 \\ 0 & 0 & 0 & \mathbf{C} & 1 \\ 1 & 0 & -1 & -1 & 1 \\ 0 & -1 & 1 & 1 & \mathbf{D} \end{bmatrix}$$

Structural properties (2)

Check if the following vectors are T-invariants in the net using the state equation!

$$
W^T = \begin{array}{c|ccccc} & t_1 & t_2 & t_3 & t_4 & t_5 \\ \hline p_1 & 1 & -1 & 0 & 0 & 0 \\ p_2 & -1 & 2 & \mathbf{B} & 0 & 0 \\ p_3 & \mathbf{A} & 1 & 2 & 0 & -1 \\ p_4 & 0 & 0 & 0 & \mathbf{C} & 1 \\ p_5 & 1 & 0 & -1 & -1 & 1 \\ p_6 & 0 & -1 & 1 & 1 & \mathbf{D} \end{array}
$$

- $(2,2,2,0,0)^T$
- $(0,1,0,1,3)^T$
- $(1,2,1,1,3)^T$

Structural properties (2): solution

Check if the following vectors are T-invariants in the net using the state equation!

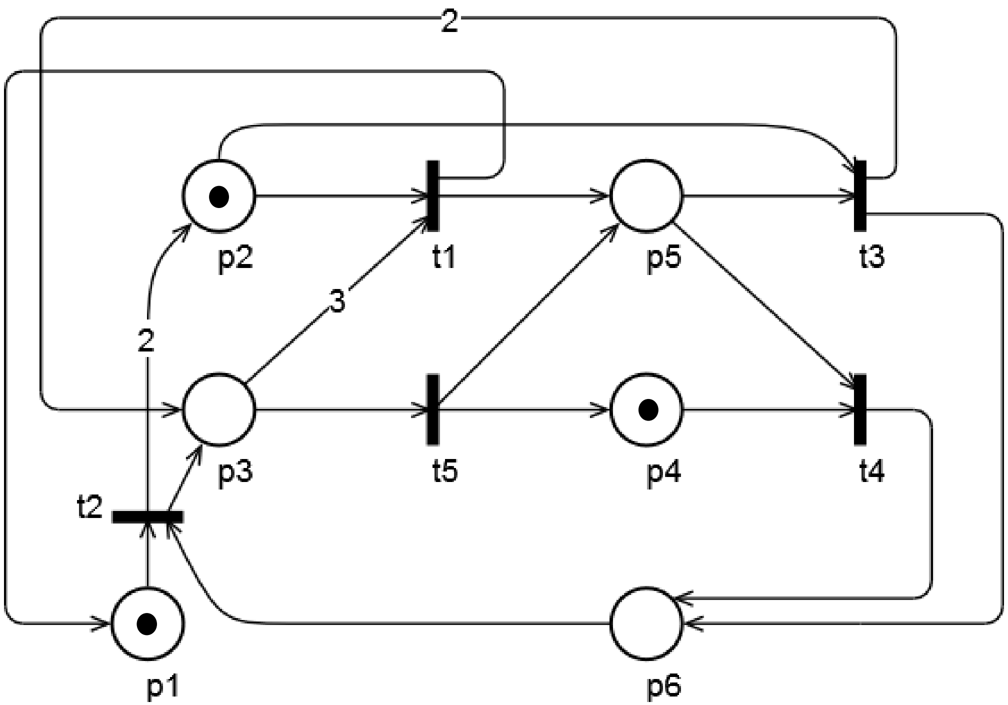
$$
\mathbf{W}^{\mathbf{T}} = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{array}
\begin{bmatrix} t_1 & t_2 & t_3 & t_4 & t_5 \\ 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & \mathbf{B} & 0 & 0 \\ \mathbf{A} & 1 & 2 & 0 & -1 \\ 0 & 0 & 0 & \mathbf{C} & 1 \\ 1 & 0 & -1 & -1 & 1 \\ 0 & -1 & 1 & 1 & \mathbf{D} \end{bmatrix}
$$

- $(2,2,2,0,0)^T$ Invariant
- $(0,1,0,1,3)^T$ Not invariant
- $(1,2,1,1,3)^T$ Not invariant

State equation check: $\mathbf{W}^{\mathrm{T}}\vec{\sigma}_T = 0$

Temporal properties

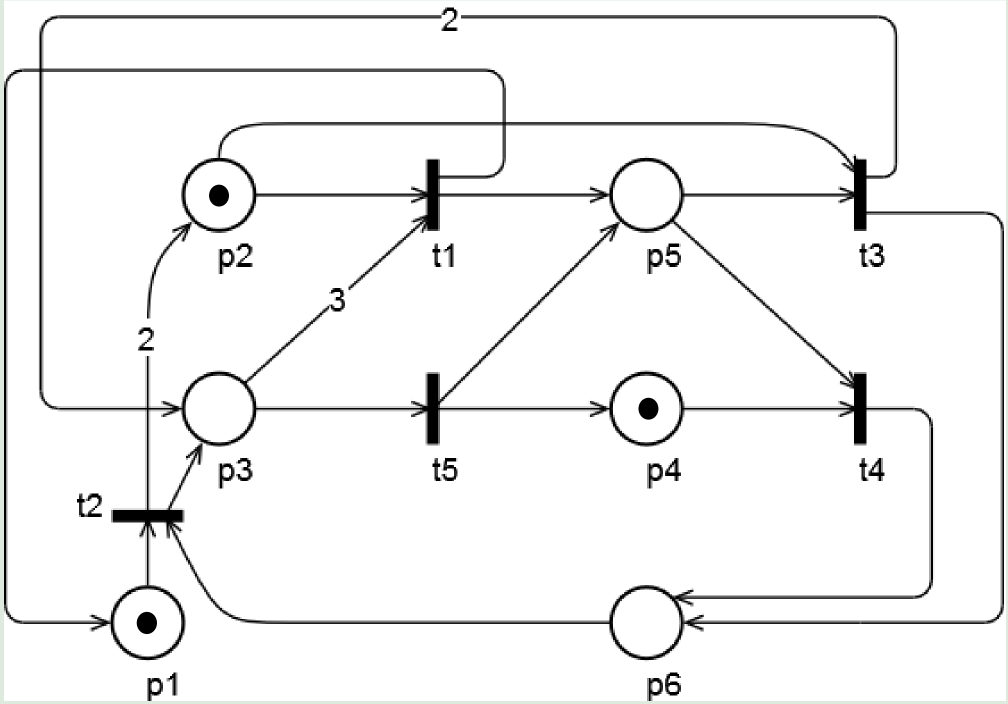
Does the following CTL expression hold for the given Petri net with the initial marking M(1,1,0,1,0,0)?



- **AG** (m(p1) + m(p2) + m(p3) + m(p5) + m(p6) = 2)

Temporal properties: solution

Does the following CTL expression hold for the given Petri net with the initial marking M(1,1,0,1,0,0)?



- **AG** (m(p1) + m(p2) + m(p3) + m(p5) + m(p6) = 2)

  In this example the whole state space only contains the initial marking for which it holds.

  In general: the initial state and the P-invariant described by the sum of tokens has to be checked.

Theoretical questions

1. Give the formal definition of P-invariants (explaining the symbols in the definition), and give an example on their practical relevance!
2. Draw a source transition and a sink transition! Explain why these transitions can endanger the liveness and safeness of the net!

Theoretical questions: solution

1. Give the formal definition of P-invariants (explaining the symbols in the definition), and give an example on their practical relevance!
   – P-invariants: The weighted sum of tokens (described by the weight vector $\boldsymbol{\mu_P}$) is constant:

$$\vec{\mu}_P^{\mathrm{T}} M = \text{constant}$$

   – Practical relevance: weighted sum of resources in a workflow is constant.
2. Draw a source transition and a sink transition! Explain why these transitions can endanger the liveness and safeness of the net!
   – Source transition: Has only outgoing edges. It can only produce tokens, possibly endangering boundedness and safeness.
   – Sink transition: Has only incoming edges. It can only consume tokens, possibly endangering liveness.