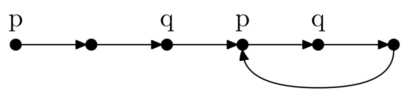
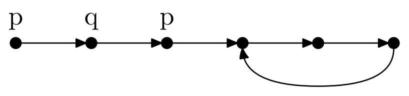
1. Only two finite automata are shown for each formula, one where the formula holds and one where it does not hold. There are also other possible automata for which the formula holds or does not hold.

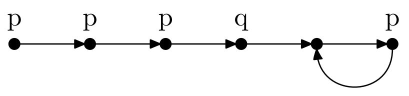
   (a) $\Box(p \rightarrow \Diamond q)$
   An automaton where the formula holds:



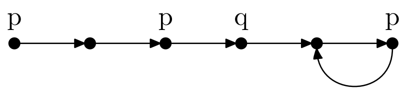
   An automaton where the formula does not hold:



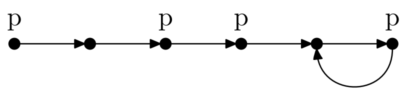
   (b) $(p \mathbin{\mathsf{U}} q) \vee (\Box\neg q)$
   An automaton where the formula holds:



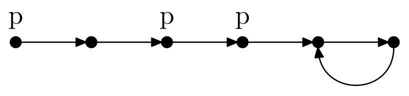
   An automaton where the formula does not hold:



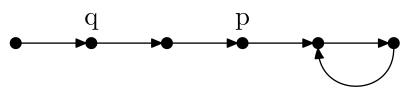
   (c) $\Box\Diamond p$
   An automaton where the formula holds:



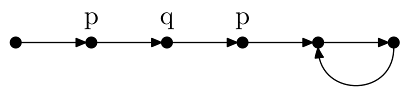
   An automaton where the formula does not hold:



   (d) $\Diamond p \rightarrow (\neg p \mathbin{\mathsf{U}} q)$
   An automaton where the formula holds:



   An automaton where the formula does not hold:



2. Define $P(\langle x, y\rangle, z)$ to be true, if there are tokens $\langle x, y\rangle$ and z in place P. The names of places and values of tokens have been abbreviated in the formulae. A formula holds in the model only if it holds *for all possible executions.*

   (a) All passengers reach the opposite shore safely:
   $\Diamond(\text{pass}(\langle c, 2\rangle) \wedge \text{pass}(\langle g, 2\rangle) \wedge \text{pass}(\langle w, 2\rangle))$
   The formula does not hold in the model. There is an execution $\{1 \rightarrow 2[p = goat], 2 \rightarrow 1[p = goat]\}^{\omega}$. In the execution one passenger is moved back and forth infinitely.

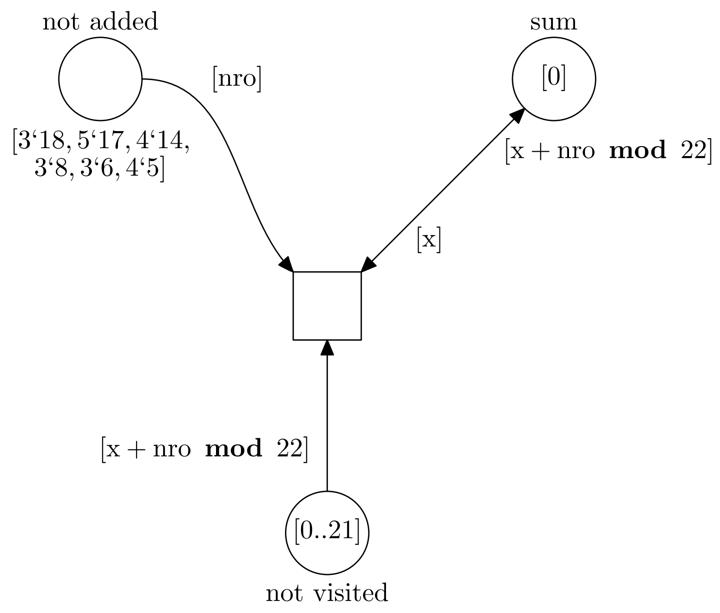
   (b) Wolf eats the goat:
   $\Diamond((\text{pass}(\langle g, 1\rangle, \langle w, 1\rangle) \wedge \text{boat}(2)) \vee (\text{pass}(\langle g, 2\rangle, \langle w, 2\rangle) \wedge \text{boat}(1)))$
   The formula does not hold in the model. The execution in (a) can be used as a counterexample.

   (c) It is not possible that a passenger is eaten:

   $$\begin{aligned}\Box\neg((&\text{pass}(\langle c, 1\rangle, \langle g, 1\rangle) \wedge \text{boat}(2)) \vee (\text{pass}(\langle g, 1\rangle, \langle w, 1\rangle) \wedge \text{boat}(2)) \vee \\ &(\text{pass}(\langle c, 2\rangle, \langle g, 2\rangle) \wedge \text{boat}(1)) \vee (\text{pass}(\langle g, 2\rangle, \langle w, 2\rangle) \wedge \text{boat}(1)))\end{aligned}$$

   The formula does not hold in the model. For example, by firing the transition $1 \rightarrow 2[p = w]$ an unwanted state is reached.

   (a) An algebraic net for solving the problem:



(b) Maria description of the algebraic net:

```
typedef unsigned(0..22) Int;
place NotAdded Int : 3#18,5#17,4#14,3#8,3#6,4#5;
place NotVisited Int : Int a (a <= 21) : a;
place Sum Int : 0;
trans Add
  in {
    place NotAdded : num;
    place NotVisited : (x+num)%22;
    place Sum : x;
  }
  out {
    Sum : (x+num)%22;
  };
```

(c) It is easy to add a reject formula which stops the reachability analysis when a solution is found. Because the task was to find a solution in exactly 22 steps a possible formula is

```
reject (cardinality place NotVisited == 0) && fatal;
```

There are initially exactly 22 tokens in place NotVisited, and every firing of the transition comsumes one of them. Also, the values will be consumed when they are visited, and the condition “Traverse all numbers $0 \leq i \leq 21$” will be satisfied.

The evaluation of the keyword `fatal` causes the analyzer to stop the generation of the reachability graph. Were the `fatal` not used, Maria would generate the complete reachability graph and report the markings which satisfy the reject formula.

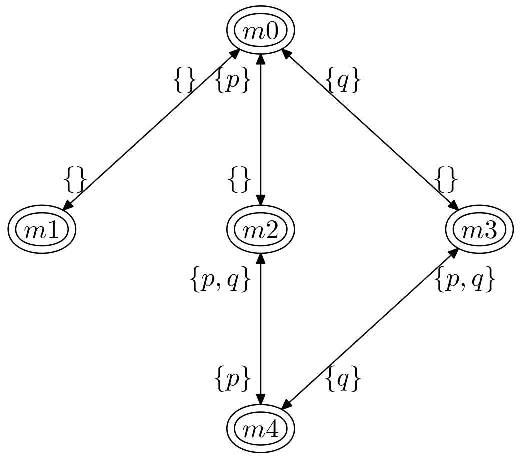
(d) To solve this problem reachability analysis is too powerful a tool. In reachability analysis, all aready reached states are stored, and their number can be exponential to the size of the input. Using the solution methods for combinatorial problems we can solve the problem in linear time related to the size of the input.

3. This example is for demonstration purposes only, as the subject is beyond the scope of the course.

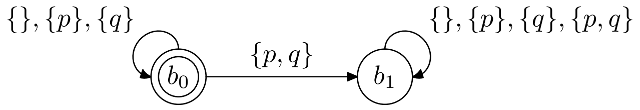
(a) The reachability graph of the net:

$$\begin{tikzcd}
 & \begin{array}{l} i_w : 1 \\ i_r : 1,2 \\ l : 1,2 \end{array} \arrow[dl, leftrightarrow] \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\begin{array}{l} w : 1 \\ i_r : 1,2 \\ l : \end{array} & \begin{array}{ll} i_w : 1 & \\ i_r : 2 & r : 1 \\ l : 2 & \end{array} \arrow[d, leftrightarrow] & \begin{array}{ll} i_w : 1 & \\ i_r : 1 & r : 2 \\ l : 1 & \end{array} \arrow[dl, leftrightarrow] \\
 & \begin{array}{l} i_w : 1 \\ r : 1,2 \\ l : \end{array} &
\end{tikzcd}$$

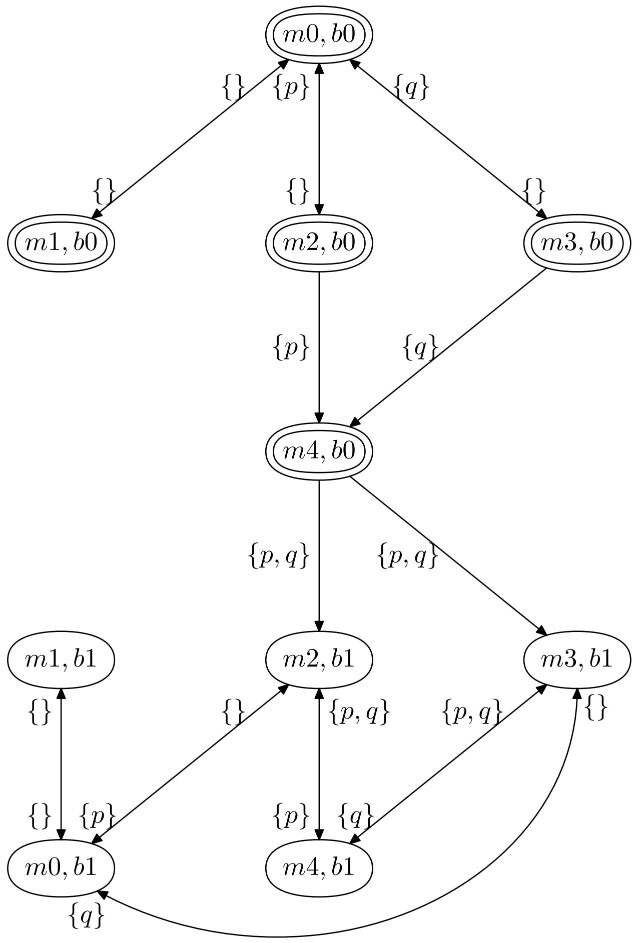
(b) A Büchi automaton $B(G)$ corresponding to the reachability graph (using the atomic propositions p and q)



A Büchi automaton $B(\neg A)$ for the negation $(\Box\neg(p \wedge q))$ of the formula $A = \Diamond(p \wedge q)$:



(c) Below is the product automaton $(B(G) \cap B(\neg A))$. The product automaton has an infinite run visiting an accepting state infinitely often: $(m0, b0) \rightarrow (m1, b0) \rightarrow (m0, b0)$ Therefore the system and the negation of the formula have shared infinite behaviour. Thus the original formula $\Diamond(p \wedge q)$ does not hold in the model.



$m0, b0$
$\{\}$
$\{p\}$
$\{q\}$
$\{\}$
$\{\}$
$\{\}$
$m1, b0$
$m2, b0$
$m3, b0$
$\{p\}$
$\{q\}$
$m4, b0$
$\{p, q\}$
$\{p, q\}$
$m1, b1$
$m2, b1$
$m3, b1$
$\{\}$
$\{\}$
$\{p, q\}$
$\{p, q\}$
$\{\}$
$\{\}$
$\{p\}$
$\{p\}$
$\{q\}$
$m0, b1$
$m4, b1$
$\{q\}$