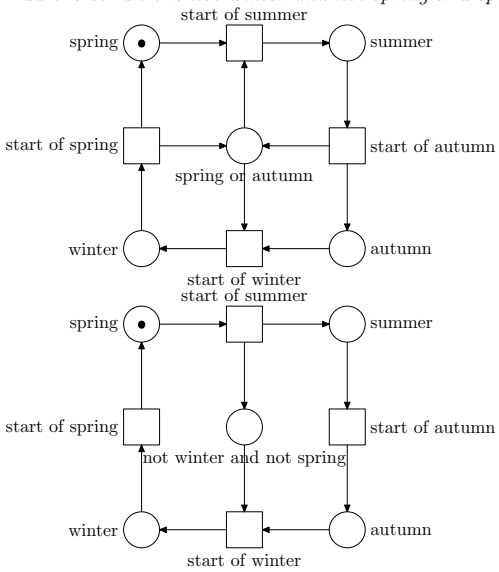
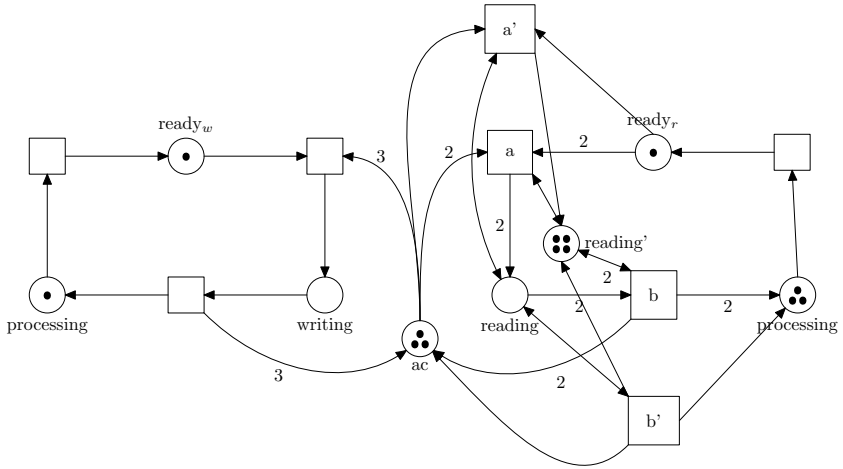
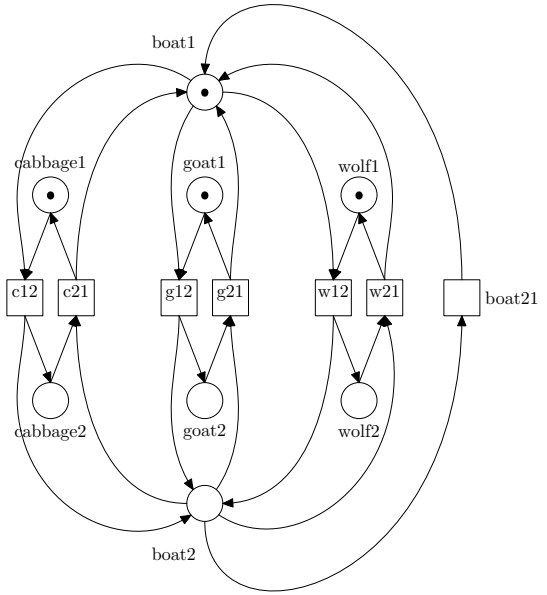
1.1 Add the conditions *not winter and not spring* and *spring or autumn.*



1.2 The net below has the necessary changes included. To keep the graphical representation readable, some arc weights have not been included in the picture. Following arcs of transition a' have weights different from one: (reading,a'):2; (a',reading):3. Following arcs of transition b' have weights different from one: (reading,b'):3; (b',reading):2. Following arcs of transition a have weights different from one: (reading',a):4;(a,reading'):2. Following arcs of transition b have weights different from one: (reading',b):2; (b,reading'):4.



2.1 Below is a C/E-system modelling the given problem. The conditions cabbage1, goat1 and wolf1 represent the situation where the creature has not yet crossed the river, while the conditions cabbage2, goat2 and wolf2 represent the situation where the creature has crossed the river. Similarly for the conditions boat1 and boat2.



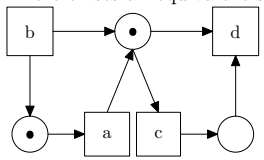
We can search the solution by playing the token game. The rejected cases are such that have conditions $\{\text{goat}_i, \text{cabbage}_i, \text{boat}_j\}$ or $\{\text{goat}_i, \text{wolf}_i, \text{boat}_j\}$, where $i, j \in \{1, 2\}$ and $i \neq j$. One possible solution is generated by the occurrence of the following events, in presented order: g12, boat21, w12, g21, c12, boat21, g12.

2.2 s_1 : spring or summer; s_2 : autumn or spring; s_3 : summer or autumn.

2.3 The systems are not equivalent. A first hint to the non-equivalence is the difference between events c and e: in the first net they are detached but in the second net they share a postcondition. This results in a different number of steps regarding those events (see proposition 2.4 (d)). We can prove the non-equivalence by showing that the case graphs are nonisomorphic (proposition 2.6 (d)). This turns out indeed to be the case, with the first net having 13 possible cases while the second net only has 8.

2.4 The case graph of the system consists of four cases which occur sequentially: $c_1 : \{s_1, s_4, s_6\}[c\rangle c_2 : \{s_1, s_3, s_7\}[a\rangle c_3 : \{s_2, s_4, s_7\}[d\rangle c_4 : \{s_2, s_5, s_6\}[b\rangle c_1$.

There exists an equivalent system with three conditions:



The case graph of the system presented above is $c_1 : \{s_1, s_2\}[c\rangle c_2 : \{s_1, s_3\}[a\rangle c_3 : \{s_2, s_3\}[d\rangle c_4 : \emptyset[b\rangle c_1$. The case graphs are isomorphic, and therefore the nets are also equivalent (proposition 2.6 (d)). The net with three conditions is also minimal w.r.t. number of conditions.

The maximum number of cases with n conditions is 2^n. Thus, we need at least two conditions.

Assume that we have exactly two conditions, q and r. From the equivalence requirement it follows that the cases are {}, {q}, {r} ja {q,r}. The enabledness and occurrence definitions of C/E-systems are taken into account on each row of the following table. The table lists all possible occurrences of events in a system with conditions p and q.

preconditions	postconditions	occurrences in case graph
$\{\}$	$\{q\}$	$\{\} \to \{q\}$ ja $\{r\} \to \{q, r\}$
$\{\}$	$\{r\}$	$\{\} \to \{r\}$ ja $\{q\} \to \{q, r\}$
$\{\}$	$\{q, r\}$	$\{\} \to \{q, r\}$
$\{q\}$	$\{\}$	$\{q\} \to \{\}$ ja $\{q, r\} \to \{r\}$
$\{q\}$	$\{r\}$	$\{q\} \to \{r\}$
$\{r\}$	$\{\}$	$\{r\} \to \{\}$ ja $\{q, r\} \to \{q\}$
$\{r\}$	$\{q\}$	$\{r\} \to \{q\}$
$\{q, r\}$	$\{\}$	$\{q, r\} \to \{\}$

It turns out that we can not choose four rows in such a way that the equivalence requirement is satisfied. Note that there must be exactly four events according to the proposition 2.4 (d).

2.6

Fig. 1 Contact free.

Fig. 2 Contact free.

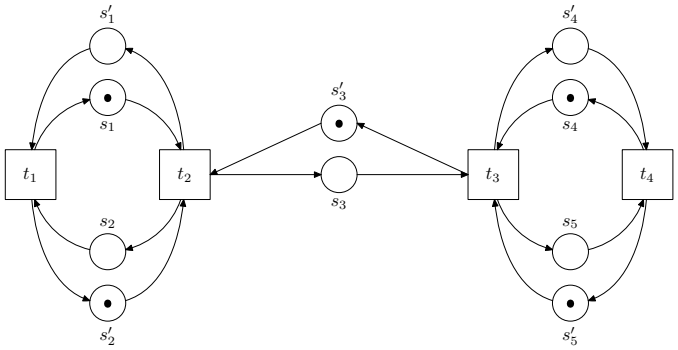
Fig. 21 Contact free.

Fig. 22 Contact free.

Fig. 24 Contact in the system presented in the figure; event *e1* can not occur even though the precondition is satisfied.

Fig. 25 Contact in the system presented in the figure; event *start of spring* can not occur even though the precondition (empty) is satisfied.

2.7 Complement construction is done by adding a complement condition for each condition in the system, with arcs "turned around".



2.8 Case graph contains all cases that can be reached by event occurrences, either forward or backward. One possibility is to first generate all cases reachable by forward event occurrences and then check each case for backward occurrences resulting in an undiscovered case. If such cases are found, iterate until no new cases are found neither by forward occurrences nor by backward occurrences. Because the case graph of a C/E-system is finite, the method will eventually terminate with the case graph fully constructed.