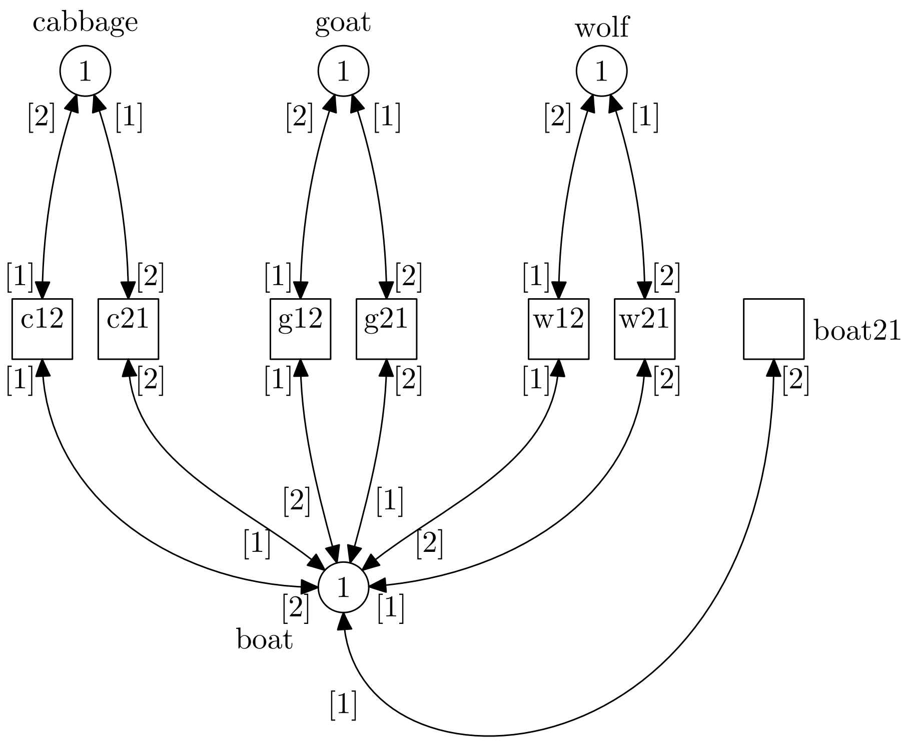
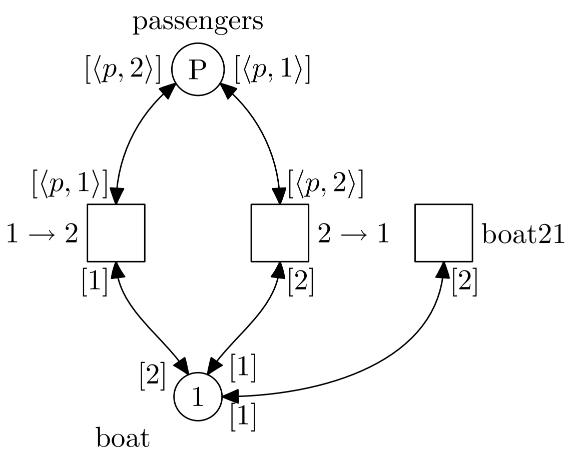
Laboratory for Theoretical Computer Science — Answers for tutorial 3
T-79.231 Parallel and Distributed Digital Systems — 10th October 2003

1. Below is presented a high-level net modeling the cabbage-goat-wolf problem. The net was made by folding the places modeling the location of passengers and boat. The data type for all places is $T = \{1, 2\}$.



In the following picture the net has been folded further. All places modeling the passengers have been folded together, but the place modeling the boat is still kept separate. We denote the set of passengers by $P = \{cabbage, goat, wolf\}$. The data type for the place passengers can now be written as $D = P \times T$.

The transitions $1 \to 2$ and $2 \to 1$ could be folded. The place boat could also be folded with place passengers. Then, the arc expression on the input arc of the folded transition would be $[\langle p, x\rangle, \langle boat, x\rangle]$. The expression on output arc would be $[\langle p, x \oplus 1\rangle, \langle boat, x \oplus 1\rangle]$, where $x \oplus 1 = (x \bmod 2) + 1$.

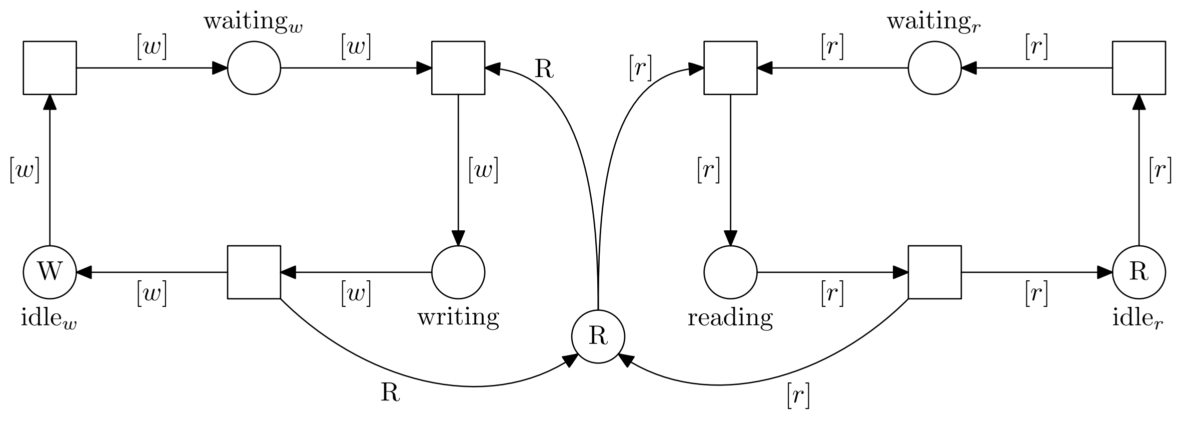


The reachability graph of the net is generated in basically the same way as with low-level nets. However, when firing the transitions, one must remember to take in the account the bindings of transition variables. One high-level transition can fire with different bindings from one marking.

The first three steps in generating the reachability graph are presented in the picture. For clarity, only the first letter is used for the passengers. The boxed markings are rejected markings, therefore the reachability graph is not generated further from such markings.

$$\begin{tikzcd}
\begin{array}{l}\text{pass:}\langle c,1\rangle,\langle g,1\rangle,\langle w,1\rangle\\ \text{boat:1}\end{array} \arrow[r, "1\to 2:[p=c]"] \arrow[d, "1\to 2:[p=g]"] \arrow[dr, "1\to 2:[p=w]"] & \boxed{\begin{array}{l}\text{pass:}\langle c,2\rangle,\langle g,1\rangle,\langle w,1\rangle\\ \text{boat:2}\end{array}} \\
\begin{array}{l}\text{pass:}\langle c,1\rangle,\langle g,2\rangle,\langle w,1\rangle\\ \text{boat:2}\end{array} \arrow[u, bend left, "2\to 1:[p=g]"] \arrow[d, "\text{boat21}"] & \boxed{\begin{array}{l}\text{pass:}\langle c,1\rangle,\langle g,1\rangle,\langle w,2\rangle\\ \text{boat:2}\end{array}} \\
\begin{array}{l}\text{pass:}\langle c,1\rangle,\langle g,2\rangle,\langle w,1\rangle\\ \text{boat:1}\end{array} \arrow[r, "1\to 2:[p=w]"] \arrow[d, "1\to 2:[p=c]"] & \begin{array}{l}\text{pass:}\langle c,1\rangle,\langle g,2\rangle,\langle w,2\rangle\\ \text{boat:2}\end{array} \\
\begin{array}{l}\text{pass:}\langle c,2\rangle,\langle g,2\rangle,\langle w,1\rangle\\ \text{boat:2}\end{array} &
\end{tikzcd}$$

2. The solution to the problem is presented below. The set of readers is denoted by $R = [r_1, r_2, \ldots, r_m]$ and the set of writers $W = [w_1, w_2, \ldots, w_n]$. The data type of places idle$_r$, waiting$_r$ and reading is $D_r = \{r_1, r_2, \ldots, r_m\}$. The data type of places idle$_w$, waiting$_w$ and writing is $D_w = \{w_1, w_2, \ldots, w_n\}$. The data type of the place controlling reading and writing is $D_r = \{r_1, r_2, \ldots, r_m\}$.



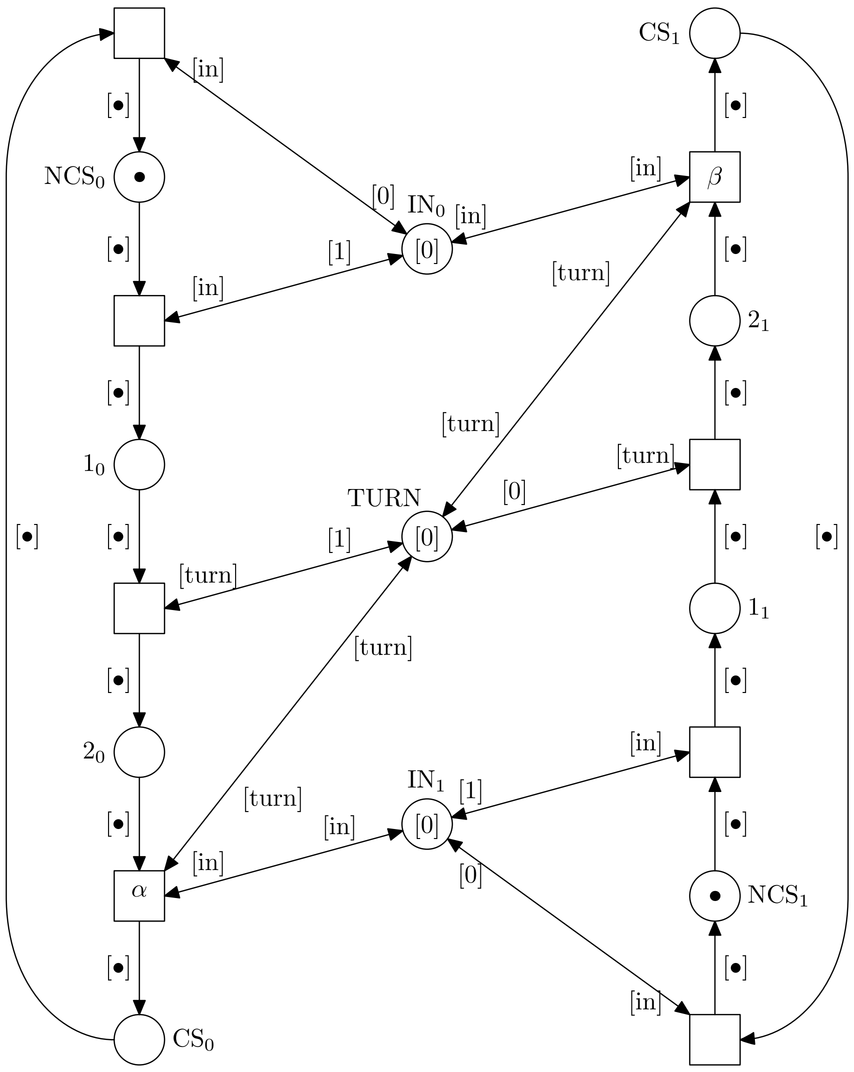
3. Below is a picture of the algebraic net modeling the Peterson's algorithm. The gate α is in $= 0 \vee$ turn $= 0$ and the gate β on in $= 0 \vee$ turn $= 1$.

   A suitable fact transition for checking the mutual exclusion would be one whose preplaces would be CS_0 and CS_1.

   If a reject formula is used, a suitable formula could be

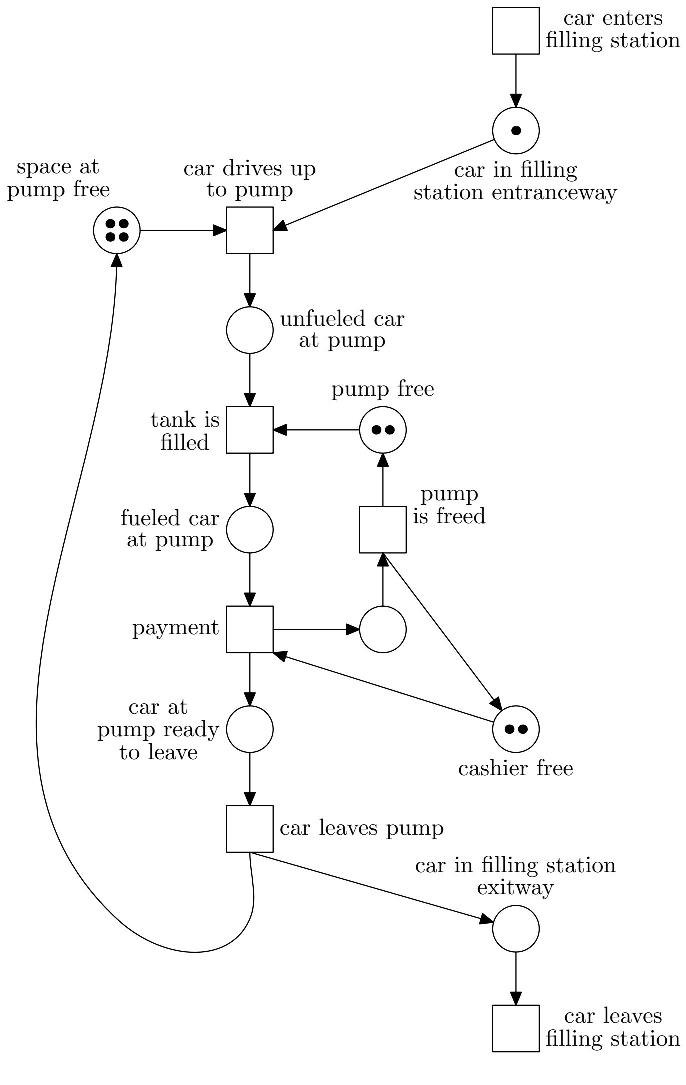
   ```
   reject place cs0 equals is token_t {} &&
          place cs1 equals is token_t {}
   ```

   It is assumed in the formula presented above that the black token (•) is defined in the Maria analyzer as:
   `typedef struct {} token_t;`



4. In the following picture is presented a simplified P/T system modeling the fueling station. The modifications suggested in part b) are also added. The initial marking of the place “space at pump free” has been increassed from two to four and the initial marking of the place “cashier free” from one to two. There are two tokens in the place “pump free”, because it now models both pumps at the station.

   We can make the system finite by adding a place that restricts the number of tokens in each place to some finite number n. The place will be a postplace of transition “car leaves filling station” and a preplace of transition “car enters filling station”. The place will thus restrict the firing of transition “car enters filling station”. If we model a fueling station for a garage for n cars, we could give the restricting place an initial marking of as many tokens as the garage has cars.


car enters filling station
car in filling station entranceway
space at pump free
car drives up to pump
unfueled car at pump
pump free
tank is filled
pump is freed
fueled car at pump
payment
car at pump ready to leave
cashier free
car leaves pump
car in filling station exitway
car leaves filling station