CPN Assignment PMS 2IH10 (2018/2019 Q1)

Part 1

Version of September, 27th

1 Overview

This assignment belongs to the course Process Modelling and Simulation (PMS).

On the following pages you will find the description of a process with various requirements. The basic environment and definitions of the process as well as the general assignment are explained in Section 2; on CANVAS, you can find a file **`baseModel_ID<YOURSTUDENTID>.cpn`** that already contains this environment and definitions. In this assignment, you are asked to extend the provided model to satisfy a number of different requirements. The deadline for the assignment is **8th October, 22:00.**

This assignment is worth maximum 7 points, which will be complemented by a second assignment of 3 points. The second assignment will be centered on simulating a correct and complete solution of this assignment that will be provided on CANVAS after the deadline of this assignment.

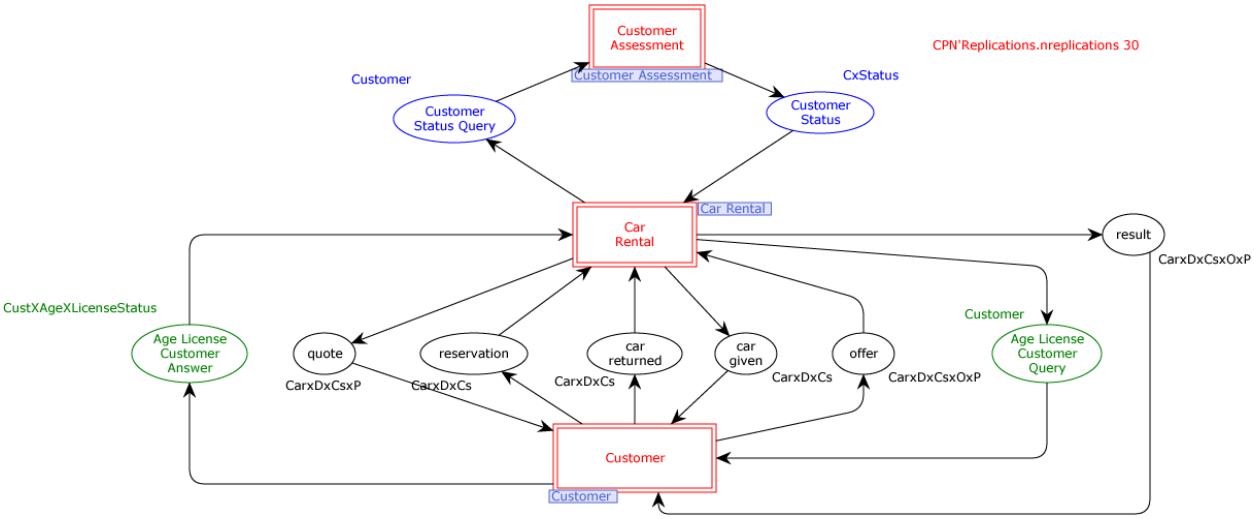
The points obtained in the CPN assignment remain valid for the rest of the academic year, i.e., a resit of the exam cannot be used to improve points obtained in the CPN assignment.

Models will be completely evaluated by a grader software that automatically checks all requirements of the assignment. The grader requires the model to not contain syntactic errors, which are highlighted in red in CPN-Tools. Any submitted model with syntactic errors will be immediately discarded and receives 0 points. The software only works correctly if every place and transition is named and all the names of transitions and places are different.

The grader will be available on CANVAS not later than Monday, September 24th.

2 Assignment Description

The setting of the assignment is a car rental agency. Part of the model is already given in CPN Tools. The top level model of the system is shown below. The purpose of the system is to model the procedure that enables customers to request a reservation for car rental.



The subnets *Customer* and *Customer Assessment* are already given. The subnet *Car Rental* is empty. The three nets are connected through a number of places as in the figure above. Color sets, variables, and basic functions are given under *Declarations* in Section *Provided Declarations*:

```
▼Provided Declarations
  ▼colset TUNIT=UNIT timed;
  ▼colset Car = with Compact | Minivan | Sport timed;
  ▼colset Customer = INT;
  ▼colset Cs = list Customer;
  ▼colset Age = INT;
  ▼colset CustomerStatus = with Legitimate |  Blacklisted;
  ▼colset LicenseStatus = with VALID | INVALID | RECENT;
  ▼colset Duration = INT;
  ▼colset Price = INT;
  ▼colset Offer = BOOL;
  ▼colset CarxD = product Car * Duration timed;
  ▼colset CarxDxCs = product Car * Duration * Cs timed;
  ▼colset CarxDxCsxOxP = product Car * Duration * Cs * Offer * Price timed;
  ▼colset CarxDxCsxP = product Car * Duration * Cs * Price timed;
  ▼colset CarxCs = product Car * Cs timed;
  ▼colset CxStatus = product Customer * CustomerStatus timed;
  ▼colset AgeXLicenseStatus=product Age * LicenseStatus;
  ▼colset CustXAgeXLicenseStatus=product Customer * Age * LicenseStatus timed;
  ▼var car: Car;
  ▼var c: Customer;
  ▼var d,d1: Duration;
  ▶var licStatus
  ▶var offer
  ▼var cs:Cs;
  ▼var s : CustomerStatus;
  ▼var a:Age;
  ▼var i,j,k : INT;
  ▶var b
  ▼globref numCompact = 25;
  ▼globref numMinivan = 25;
  ▼globref numSport = 25;
  ▼fun adjust(x)=if (x>1 orelse x<(~1)) then abs(x) else 1;
  ▼fun costDailyRent(car : Car) : Price = if (car=Compact) then 30 else if (car=Minivan) then 40 else 70;
  ▼fun costDailyInsurance(car : Car) : Price = Real.trunc(Real.fromInt(costDailyRent(car)) *0.5);
  ▼fun genLicStatus()=if (discrete(0,4)<4) then VALID else if (discrete(0,4)=0) then INVALID else RECENT;
▶Additional Declarations
```

If you need to add more declarations, list them in Section *Additional Declarations.*

The subnet *Customer Assessment* models the procedure to assess the reliability of the customers that request a reservation, whereas *Customer* focuses on modelling the behavior of customers and provides information about the customer age and driving license. A customer is identified by a customer identifier, which is a positive number. Note that, in the provided model, the simulation time unit corresponds to one hour.

The car rental agency operates as follows. A reservation is requested by a set of customers and is modelled by producing a token in the *reservation* place that consists of a tuple with the required car category, the rental duration in days (beginning from the reservation day) and the list of customers who will drive the car during the rental duration of the current reservation; e.g. **`(Minivan,2,[100,891])`**. The rental agency offers three categories of cars for rent to customers: Compact, Minivan and Sport. The rental agency has 25 cars of each of the three category. To simplify the task of modelling, assume that a customer can only be involved in one reservation a time: (s)he can request a new reservation only when the previous is concluded.

For each received reservation, the rental agency first checks whether a car of the request type is available.

If no car of the requested type is available, the reservation is cancelled and a token is generated in place *result* associated with a tuple where the fourth element of the tuple is false and the fifth contains a negative value. This concludes the management of the reservation. For instance, for the running example, it is **`(Minivan,2,[100,891],false,~1)`**.[1]

If a car of the requested type is available, this is preemptively allocated and the status of each customer is checked. This is queried to the Customer Assessment by generating one token for each customer of the reservation in place *Customer Status Query*. In the running example, two tokens are generated associated with values **`100`** and **`891.`** Eventually, the customer assessment will provide an answer for each customer.[2] For each customer of the reservation, the answer can be that (s)he is legitimate or, because of the past history, is blacklisted. The answer is provided by generating a token in place *Customer Status* of form, e.g., **`(100,Legitimate)`** or **`(100,Blacklisted)`**.

If any of the reservation customer is blacklisted, the reservation is cancelled and a token is generated in place *result* with false as forth element of the tuple and a price of zero. This concludes the management of the reservation. For the running example, it is **`(Minivan,2,[100,891],false,0)`**.

If every reservation customer is legitimate, the status of the driving license and the age of each customer is checked. This is performed by generating one token for each customer in place *Age License Customer Query*. The status of the driving license can be **`VALID`**, **`INVALID`** or **`RECENT`** as defined in colorset **`LicenseStatus`**. In the running example, two tokens are generated associated with values **`100`** and **`891.`** Eventually, an answer is received for each customer as a token in place *Age License Customer*

[1] In CPN Tools, the minus symbol is replaced by a tilde, such as **`~1`** instead of **`-1`**.

[2] The query response time is modelled in net *Customer Assessment* as a delay uniformly distributed between 1 and 4 hours.

Answer.[3] For instance, for the example in question, a token for customer 100 could be of color **`(100,17,RECENT)`**.

Depending on the age and the status of the driving license of the customers, the following scenarios are possible:

1. **At least one customer is less than 18 years old.** The reservation is cancelled and a token is generated in place *result* with false as fourth element of the tuple and a price of zero. This concludes the management of the reservation. For the running example, the token is of form **`(Minivan,2,[100,891],false,0)`**;
2. **The driving license of one or more customers is invalid.** Same as the above;
3. **For at least one customer, the driving license is recent and the age is between 18 and 24.** Same as the above;[4]
4. **No customer falls into the categories above and, for at least one customer, the driving license is recent or the age is between 18 and 24.** The reservation requires an additional insurances whose price is computed by leveraging on function **`costDailyInsurance`**, which is provided among the declarations. In particular, the price of any additional insurance for a reservation of **`d`** days for a car of type **`c`** is computed as **`d*costDailyInsurance(c)`**. For our working example, this becomes **`2*costDailyInsurance(Minivan)`**. Buying/adding the additional insurance takes 1 hour.
5. **All reservation customers are older than 25 years old and their driving licenses are valid.** See below.

For the 4th case after buying the additional insurance and for the 5th case, the reservation process continues as follows. The price is computed through the function **`costDailyRent`**, which is provided among the declarations. In particular, the price of any additional insurance for a reservation of **`d`** days for a car of type **`c`** is computed as **`d*costDailyRent(c)`**. For our working example, this becomes **`2*costDailyRent(Minivan)`**. Of course, if the additional insurance was bought, the reservation cost is the sum of the cost of the rental and of the additional insurance. Computing the price for the reservation takes 2 hours (not for the addition insurance, which takes 1 hour as mentioned above at point 4).

Once the cost is computed a token is accordingly generated in place *quote* where the last component of the tuple associated to the token contains the cost just computed. For instance, for the working example, assuming that **`costDailyRent(Minivan)`** returns 40 and the additional insurance is not necessary, this corresponds to generating a token **`(Minivan,2,[100,891],80)`**.

At this point the offer needs to be either accepted or rejected. This is done through the *Customer* subnet, which eventually generates a token in place *offer* where the four component of the tuple associated

[3] The age is uniformly distributed between 17 and 65 years. Also, on average, 80% of the licenses are VALID, 16% are RECENT, and nearly 4% are INVALID. The answers to the queries on age and drive license takes an amount of time that is normally distributed with average 5 hours and variance 2 hours. For further information, one can analyze the subnet *Customer* (with special focus on the red-colored transitions at the bottom-right corner).
[4] In this assignment, "between" includes the extremes. So, "between 18 and 24" means 18,19,…,23,24.

indicates whether the offer has been accepted or rejected.[5] For instance, for the running example, a token **`(Minivan,2,[100,891],true,80)`** will be generated if the offer is accepted or **`(Minivan,2,[100,891],false,80)`** if the offer is rejected.

If the offer is rejected, the *Car Rental* subnet is expected to forward the token to the place *result* and the execution concludes (note that the colorsets of places *offer* and *result* are the same).

If the offer is accepted, the car is ready to be given out, which should be delayed of a number of hours that are uniformly distributed between 1 and 3. The action of giving the car is made evident through the production of a token in place *car given*, such as **`(Minivan,2,[100,891])`**. After the expected number of days, the car is returned. The *Customer* subnet models this by producing a token in place *car returned*. When the car is returned, the car rental releases the car, which becomes available again for new reservations and produces a token in place *result*, such as **`(Minivan,2,[100,891],true,80)`**. This concludes the process of managing a car reservation.

3 Assignment Preparation and Submission

The CPN assignment consists in completing the partial model provided as a file "**`baseModel_IDxxxxxxx.cpn`**" via CANVAS. The file with your student id in the name **must be used** to model the **individual** tasks. This is to prevent frauds where students "pass" a (partial) solution to each other. Therefore, do not use a base model designated for another student; otherwise, it will be considered as fraud with all the direct consequences.

The submission is done through CANVAS and consists in submitting your personal file "**`baseModel_IDxxxxxxx.cpn`**", where you extend the partial model so as to meet all the requirements.

The number of cars of the different types must be configurable by varying the `globref` constants `numCompact`, `numMinivan`, `numSport`. Conversely, the possible types of cars are fixed.[6]

The number of tokens in all the places of subnet *Car Rental* should not grow indefinitely, which is equivalent to say that the number of tokens in *Car Rental* at the end of any simulation is the same as the number at the beginning of the simulation.

Bear in mind the following:

- ☐ Always use the base file specific for you, do not use base files specific to other students.
- ☐ Only modify the subnet *Car Rental* of your base model
  - ☐ Do not change any of the provided declarations or other subnets. Models in which the given environment (*Customer Assessment* and *Customer*) or the given declarations have been changed will be automatically evaluated with 0 points.

[5] The *Customer* subnet simulates a scenario in which 40% of the offers are rejected and 60% are accepted. For further information, one can analyze the subnet *Customer* (with special focus on the green-colored transitions at the bottom-right corner).

[6] The `globref` constants can be read and referenced through the name preceded by an exclamation mark (such as `!numSport`).

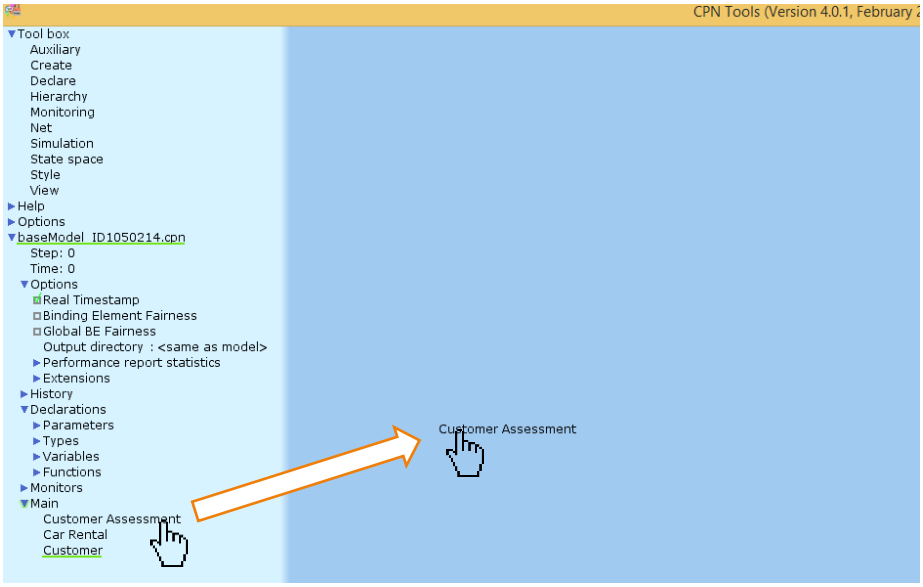
- ☐ We recommend that you use an extra declaration block, for instance the provided block called "`Additional Declarations`", for the declarations you add.
- ☐ Unless specified otherwise, your solution should be *generic* and independent of the number of cars and the specific behavior modelled in the *Customer Assessment* and *Customer* subnets.
- ☐ Models will be judged on correctness (i.e., conformance to specification).
  - ☐ Use any feedback you receive to improve your solutions;
  - ☐ Keep the model simple: use simple data structures and avoid introducing functions when you can prevent it;
  - ☐ It should be possible for a fellow student to understand your model within a few minutes;
  - ☐ **Each submission is individual and cannot result from a group work**;
  - ☐ **Models that cannot be simulated automatically or models of which the simulation does not terminate are evaluated with 0 points**.

Make use of the student-assistant sessions to get support to execute the assignment. Also, use the discussion section on CANVAS to seek for hints from fellow students, student assistants and occasionally the lecturers. When using the discussion section on CANVAS, be specific on the questions. For instance, such questions as "Why does CPN Tools report a syntax error here?" and "Why do tokens remain blocked here?" are legitimate. Conversely, very generic questions such as "How to implement this requirement?" will not be answered via the CANVAS's discussion; conversely, use the student-assistant sessions to get support for more "high-level" questions and/or ask the lecturers.

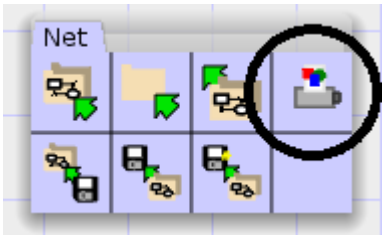
Important note: The above process description and also the subsequent tasks are partly – and on purpose – **underspecified**. The goal and purpose of this assignment is that **you resolve this underspecification** and come up with a concrete model satisfying all requirements. The purpose of the instructions is that **you ask the instructor and the student assistants** about your ideas and assumptions as you come up with a solution.

4 Hints

1. Initially all subnets are closed and, thus, when you open your model, it will "look empty". To open any subnet drags the name of the subnet and drop on the darker-blue, right-hand side:



2. Check the CPN Tools Web page (http://cpntools.org/) if you have questions about the CPN syntax and how to model with CPN Tools.
3. If you want to take a picture of your model, you can use print feature available in the "Net tool box" and click on the button as shown below:



This will generate an EPS file, which you can convert into the appropriate format to add to your reports.

4. Running a single simulation of a CPN model on a modern computer takes 1-2 minutes. If the simulation takes several minutes, then this is a hint that your model is too complex (e.g., you have places containing thousands of tokens) or the simulation does not even stop (i.e., in the `replication_report.txt` file in `output/rep_x/` no simulation is reported as completed). In this case, simulate your model manually to figure out the problem.

5. If you experience that your simulation of 30 sub-runs is incomplete, i.e., the model is not simulated until the end and some of the reservations of the 2000 customers are completed in each run, then there can be two reasons: your model has an error that prevents completion of every reservation, or the simulator is set to stop after a specific number of steps before reaching the reservations for all customers. To check which one is the case, open the “Simulation tool box” and click on the “fast-forward” simulation. The “maximum number of steps” to simulate is highlighted. Set it to a large number, e.g., 500000, and run the simulation once.

This will set the simulator to stop after 500000 steps (or when there is no transition enabled anymore). If you now run your simulation of 52 sub-runs you should get complete simulation results. If not, your model contains an error. Use manual simulation/fast-forward simulation of 50 steps to find the issue.

5 Running Simulations

CPN Tools allows to automatically simulate your model and to collect statistics about the simulation in a report. In order to properly collect the statistics, your CPN model should have the following options selected (see left and top right in the screenshots below. In case you obtain empty simulation reports, please check these options first); **in particular also check the "Fairness" options**. To start the simulation go to the *Main* subnet of your model, right-click on "`CPN'Replications.nreplications 30`" and select "Evaluate ML". See http://cpntools.org for details.