Order shipment. Part 2
Colored Petri nets

abel.armas@ut.ee

Oftentimes, it is desirable to assess the performance of systems and evaluate their efficiency. For example, when we want to analyze the average time that it takes to execute one instance of our workflow. During this lab session, you will transform your workflow into a timed CPN model and set up the simulation configuration. Therefore, we will have to make some modifications, such as declaring the set of variables as timed, specify the time that each of the tasks last, and select the data we want to gather during simulation. In order to select the data we want to save, we will use some features of CPN tools, known as monitors. There are different types of monitors, but during this practice we will use two, one for collecting data and one for setting the stop condition of the simulation.

http://cpntools.org/documentation/tasks/performance/monitors/start

Generation of cases

1. Add the following construction to the beginning of your workflow

   In this case, it will be possible to generate tokes automatically.

2. The variables and functions are the following:

```
▼Practice2
  ▼colset TI = INT;
  ▼colset MS = STRING timed;
  ▼fun NextArrival() = round(erlang(10, 5.0));
  ▼colset CASES = INT timed;
  ▼var c : CASES;
  ▼fun NewCase(c) = (c, "p"^CASES.mkstr(c)^" ", IntInf.toInt(time()));
  ▼colset DATA = product CASES*MS*TI timed;
```

   Since the tokens of type ***CASES*** are ***timed***, they will have associated a second value, called timestamp. Once the transition ***Send*** is fired, a new token is set into the place ***NextCase*** with a new timestamp (in fact, the *timestamp of previous token + a value of an erlang distribution with values 10 and 5,* http://cpntools.org/documentation/tasks/performance/random/erlang). Additionally, transition ***Send*** will put a new token (of type ***DATA***) in the place ***start***. The new token is created with three parameters as defined in ***fun NewCase(c)***, i.e., the case identifier, a string and the time of arrival. The time in CPN tools is a non-negative integer and it specifies the time at which a token is ready to use. Finally, the function time() returns the current model time (as an infinite integer).

3. In the workflow, change the color sets of the places and the type of the variables accordingly.

Adding time to the transitions

1. In order to specify the time that it takes to execute one task, we can select the transition representing the task and press the key TAB twice (i.e., when ***@+*** symbols appear). In this case, we can put a numeric value or a generated value (random, normal distribution, erlang distributions, etc.). Set the following times for each of the tasks:

Task	Duration
A	5
B	0
C	Normal distribution 15 with standard deviation of 3 (i.e., normal(15.0,3.0))
D	0
E	Normal distribution 20 with standard deviation of 5
F	7
G	10
H	0
I	Normal distribution 15 with standard deviation of 3

2. You can fire some of the transitions from your model and observe how the time associated to each of the tokens changes throughout the workflow.

Adding probability to the transitions

1. Let us suppose that the path to execute task replenish (according to the simulation of the original model, it is transition ***2*** after transition ***B***) does not occur so frequently. On average, it is executed once every 10 jobs received (10% of probability to occur). In order to model such restriction, it is necessary to declare the following elements:

    val loopRate = 0.1;
    fun Success() = uniform(0.0,1.0) <= loopRate

2. Finally, you have to add a guard in transition ***2***, such that it is fired iff ***Success()*** is true.
3. Add the same probability of execution to option ***1*** after ***B*** (path leading to transition ***F***) and to task ***D***.

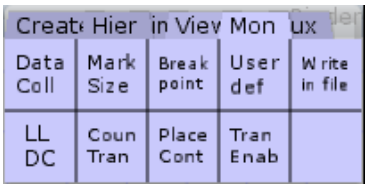
Monitors

A monitor is a mechanism in CPN Tools that is used to observe, inspect, control or modify a simulation of a CP-net. Many different monitors can be defined for a given net. Monitors can inspect both the marking of places and the occurring binding elements during the simulation, and they can make appropriate actions based on the observations. In this case, we will use two types of monitors: ***Data Collection*** and ***Breakpoint***.

1. Add the following transition and place at the end of your workflow

Declare the necessary variables for catching the values of a token of type DATA.

2. Open the Tool box Monitoring



3. Select ***Data Collection*** and click on the transition ***Log***.
   a. Observe that under ***Monitors*** in left-hand-side of the screen, there is a new monitor declaration. You can put any name you want.

   b. The Data Collection monitor has the following elements:

      i. **Initialization function** is called once before a simulation starts. It returns an optional numerical value (see below for more information about optional values). If the function returns NONE then nothing will happen. If the function returns a value of the form SOME x where x is a number, then x will be used to update statistics and it will be saved in a log file (if the Logging option for the data collector monitor is checked).
      ii. **Predicate function** is called after simulation steps. When the predicate function returns true, the observation and action functions are called.
      iii. **Observation function** examines the monitored nodes, and returns a numerical value.
      iv. **Stop function** is analogous to the initialization function, but it is called when Simulation stop criteria are met.

      http://cpntools.org/documentation/tasks/performance/monitors/data_collector_monitoring

   c. Modify the Observer function as follows:

```
Observer
    fun obs (bindelem) =
    let
      fun obsBindElem (Practice1'Log (1, {c,msg,tm})) = IntInf.toInt(time()) - tm
        | obsBindElem _ = 0
    in
      obsBindElem bindelem
    end
```

The function observer will compute the time required for a token to move from the ***start*** to the ***end*** place, i.e., *current time – time of arrival*.

Simulating your model

1. In order to automatically simulate your model
    a. Open the Tool box ***Auxiliary,*** and create a Text element.
    b. Write the following code in the created tag (hereinafter, it is referred to as **tag A**):
    CPN'Replications.nreplications 5

    The function ***CPN'Replications.nreplications*** can be used to automatically run a given number of simulations.

    http://cpntools.org/documentation/tasks/simulation/simulation_replications

2. Select ***Breakpoint*** from the Toolbox ***Monitoring*** and click on the place ***sink***. Set a name to the breakpoint defined.
    a. Modify the Predicate function as follows

    ```
    Predicate
      fun pred (Practice1'sink_1_mark : DATA tms) =
        size Practice1'sink_1_mark = 100
    ```

    Breakpoint monitors are used to stop simulations when certain conditions are fulfilled. In this example, we will stop the simulation when the marking in the place ***sink*** has size 100.

    b. Right-click over the **tag A** and select ***Evaluate ML***

3. Within the folder where your model is located, you can find a new folder "output/reps_***x***". This new folder contains all the results from the simulation. For instance, if you open the file PerfReportIID.html, then you will observe the following data:
    a. **count** represents the number times that values have been added to the monitor
    b. **min** is minimum of the observed values
    c. **max** is the maximum of the observed values
    d. **sum** represents the sum of the observed values
    e. **avrg** is the average of the observed values

Troubleshooting

1. Table in PerfReportIID.html is full of 0s
    a. A simulation will stop if one or more of the following conditions are fulfilled A simulation will stop if one or more of the following conditions are fulfilled
        i. There are no more enabled transitions
        ii. The number of steps specified for the Play or Fast forward tool has been executed
        iii. The Stop tool is applied after applying the Play tool
        iv. One of the Breakpoint monitors is fulfilled

b. By default, the ***Play*** and ***Fast forward*** tool have 50 steps. In this example, 50 steps are not enough for reaching the final place in the simulation. Thus, the simulation would be stopping before reaching the transition associated to the 'Data Collection' monitor.
c. To modify the amount of steps, right click over the tools ***Play*** and ***Fast forward*** (Figure 1) and select ***Set Options*** (Figure 2).
d. Modify the number of steps to a larger number, e.g., 1000.

Figure 1

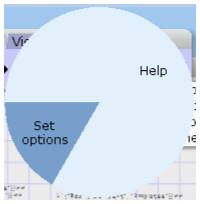


Figure 2