Handout 4 - ICTiB/Process modelling

Pieter Kwantes

February 21, 2010

Contents

1 Exercises

1.1 Top-down and Bottom-up development

In this exercise we will present a very simple "toy-problem" just to get you acquainted with the hierarchy feature of CPN-tools. Consider a small supermarket with one cash-register. We want to model the process at the cash-register in a top-down fashion. That is, we just model the entire process as one transition and postpone the creation of a more detailed model until later.

1.1.1. We start by creating a very simple Coloured Petri net with the process at the cash register represented by just one transition, one input and one output place. The arrival of customers at the cash register is represented by tokens (we can use the simplest colourset UNIT) in the inputplace. If a case (i.e. a customer) is completely processed (the customer has paid its groceries), this is represented by a token in the output place.

1.1.2. Consider the Petri net in exercise 1.1.1. Now assume that we have more detailed information on the cash register process. We now can assume that it consists of two steps: "start" and "stop" and we can assume that there is one cashier which is either "busy" (after the step "start") or " free" (after the step "stop"). We will detail the cash register process in this way by creating a subpage linked to the substitution transition in the Petri net in exercise 1.1.1. You can use the "Move to subpage" option in the "Hierarchy" menu in CPN tools, shown in the figure below, and apply it to the substitution transition. Also read about "top down development" in the CPN tools help pages.

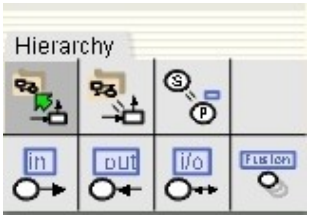

Figure 1: Move to subpage option in the Hierarchy menu of CPN tools

After you created the subpage, you amend the Petri net on that subpage to include the more detailed information about the cash register process mentioned above.

1.1.3. Extend the model you have created with an extra cash-register by adding one transition at the top-level. Then assign the detailed model of a cash register you created in exercise 1.1.2. to the added (substitution) transition on the top page by applying the "set subpage" option in the hierarchy menu of

CPN tools (shown in the figure below) first to the transition on the super page representing the extra cash register, and the to the subpage with the detailed model. Also read about "bottom up development" in the CPN tools help pages to perform this assignment.

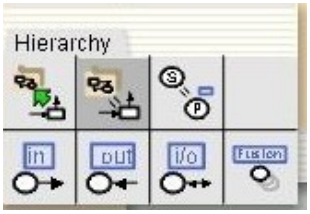

Figure 2: Move to subpage option in the Hierarchy menu of CPN tools

After you finished your model, initialize the top page to represent the fact that there are 5 customers waiting for the cash-register. Next fire the substitution transition representing the first cash register. Then examine the markings on each of the two instances of the subpage with the detailed model of the cash register. Explain what you see.

1.2 A model checking example: the dining philosophers problem

In this exercise we use the well known dining philosopher system as our main example. Five Chinese philosophers are sitting around a circular table. In the middle of the table there is a delicious dish of rice, and between each pair of philosophers there is a single chopstick. Each philosopher alternates between thinking and eating. To eat, the philosopher needs two chopsticks, and he is only allowed to use the two which are situated next to him (on his left and right side). The sharing of chopsticks prevents two neighbours from eating at the same time.

Now consider a variation of the Dining Philosophers example. A number of philosophers are initially outside a dining room. Each of them decides at some point to enter the room bringing along one chopstick (for the left hand) to be shared with the neighbor philosopher. Once in the room the philosopher sits down and starts thinking. If both left and right chopsticks are unused the philosopher can decide to start eating, thus occupying the two chopsticks. When finished eating the philosopher can decide to think again making available again the left and right chopsticks. As life is unpredictable, a philosopher can get poisoned and die while eating. In this case the chopsticks are forever lost. Obviously this model can end in a dead marking. You can find an example Petri net modelling this problem in the subdirectory "CPN Tools/samples/Diningphilosophers".

CPN tools supports a query language (ASK-CTL) with which you can query the reachability graph in order to establish the behavioural properties of the Petri net system that is associated with it. We will calculate the reachability graph of the Petri net modelling the dining philosophers problem and then run a number of queries to establish some behavioural properties of the Petri net.

2.1.1. Calculating the reachability graph Load the example net in CPN tools, and examine the model.

2.1.1a. Calculate the reachability graph manually Use the algorithm given in the lecture to calculate the reachability graph.

2.1.1b. Calculate the reachability graph with CPN tools Use the state space option in CPN tools to calculate the state space and the reachability graph. Compare the result with the result of the previous exercise.

2.1.2. Analyzing the reachability graph In this exercise you will use the query language of CPN tools to investigate the reachability graph. Below there are 4 different queries described. These queries are all available in the example, you just have to execute them and interpret the results. You should use the reachability graph you calculated manually in exercise 1.1a. to explain the results.

2.1.2.a) Is the initial marking a home marking? The following query can be used to check whether the initial marking is a home marking. A home marking is a marking to which it is always possible to return. Answer the following questions:
- what is the ASK-CTL formula given in the example that represents this query - give the CTL formula, that represents the same query - derive the answer to the query based on the reachability graph - calculate the answer to ASK-CTL formula with CPN tools (execute the formula), and give the result (is it the same as what you predicted?)

2.1.2.b) Is a given marking dead? The following query can be used to check whether a given marking is dead. We assume here that the marking in question has the number 8. - what is the ASK-CTL formula given in the example that represents this query - give the CTL formula, that represents the same query - derive the answer to the query based on the reachability graph - calculate the answer to ASK-CTL formula with CPN tools (execute the formula), and give the result (is it the same as what you predicted?)

2.1.2.c) Is a given transition live? Let us consider the "take" transition from the dining philosophers example. We would like to ask whether the binding element $(take, < p = ph(2) >)$ is live. - what is the ASK-CTL formula given in

the example that represents this query - give the CTL formula, that represents the same query - derive the answer to the query based on the reachability graph - calculate the answer to ASK-CTL formula with CPN tools (execute the formula), and give the result (is it the same as what you predicted?)

2.1.2.d) Can philosopher 2 become poisoned? - what is the ASK-CTL formula given in the example that represents this query - give the CTL formula, that represents the same query - derive the answer to the query based on the reachability graph - calculate the answer to ASK-CTL formula with CPN tools (execute the formula), and give the result (is it the same as what you predicted?)

1.3 Running case exercise

The running case exercises you have done until now involved making a Petri model of the current order process at LISTA and using that model to analyze the performance of the order process. This exercise is about using Petri net modelling as a tool to support the *improvement* of the process design.

3.1.1. Assess options for improvement The improvement of the order process at LISTA was identified as a critical element of the strategic reorientation of the Company. Possible improvements that you can think of is:

1. 1. Make a clear distinction in processing of orders for standard products and processing of orders for customized products ("Triage")
2. 2. Eliminate manual activities by the introduction of an automated order clearing system

Determine how these improvements influence the design of your process model and create a dataflow diagram or a sketch of the Petri net model of the improved process like you did in the running exercises described in *Handout I* and *Handout II*.

3.1.2. Model improved process Create a Hierarchical Timed Coloured Petri net model of the improved process like you did in the running case exercise described in *Handout III*.

3.1.2. Analyze improved process Analyze performance the Petri net model of the improved process like you did in the running case exercise described in *Handout III*. Establish the completion time of an order and compare it to the completion time you measured in the Petri net model of the current order process.

2 Analysis of Petri Nets

There are a large number of techniques available for analyzing Petri nets. This is in fact one of the reasons that the use of Petri nets for modelling real world problems appears to be a promising approach. In principle these techniques enable us to check whether the design of some process or system meets the specified requirements, before we put any effort in implementing the design. This could save a lot of effort and troubles. Ideally we would like to be able to **prove**that the design meets specific requirements in all circumstances. To do that we need we need **formal analysis methods**, which will be the subject of the next section. A second best approach would be that we determine that the design **probably** will meet the requirements. This can be accomplished by simulation, which was discussed in the 4th lecture.

2.1 Formal analysis

The best we can do in many cases to determine whether some system will perform satisfactorily is to test the implementation of the system or process. The drawback is that errors can be overlooked this way and if errors are found we have to change the implementation which can be a problem. If it were possible to *calculate* the properties of a system or process based on its *specification*, then both problems would be solved. The purpose of formal analysis is just to do that. A number of formal analysis techniques are available for Petri nets. These techniques can be conveniently categorized by making a distinction between *qualitative* and *quantitative* properties. Properties that can be specified by a logical formula, like "absence of deadlocks", are generally called qualitative properties. Properties concerning the performance of a system or process, like throughput, response time or efficiency, are generally called "quantitative" properties. Formal analysis of Petri nets is quite a subject in its own right, only a short introduction will be given below.

2.1.1 Qualitative analysis

Examples of some basic properties of Petri nets that have been studied extensively in the past are :

Boundedness A place in a Petri net is k-bounded if the number of tokens in that place cannot exceed an integer k. A Petri net is k-bounded if every place of the net is k-bounded.

Liveness A transition in a Petri net is live if for every reachable marking there is an occurrence sequence that contains that transition. So a live transition is always "potentially fireable".

Reachability The previous problems are in fact special cases of the "reachability problem". The reachability problem is concerned with the question whether, given a Petri net and a marking M, some marking $M^{'}$ is reachable from marking M.

In general, the purpose of analysis is to determine whether a given Petri net exhibits certain properties, like those given in the examples above. Analysis methods for Petri nets are categorized in [3] as either *enumeration-* or *net-driven* methods. Enumeration-driven methods start out with the computation of the occurrence graph. Net-driven methods focus on the structure of the Petri net and its initial marking. A quick view on these two approaches is given next.

Enumeration driven analysis A important development in the area of formal verification is the invention of model checking about a quarter of a century ago. In this section we give a short introduction on the subject. Model checking is an automatic technique for verifying finite state concurrent systems. Verification in this context means delivering irrefutable proof of the existence or absence of some property of a model of a system or process. Model checking of a system consists of the following three steps:

- create a model of the system
- specify the properties to be verified
- perform the verification

The occurrence graph of a Petri net, $G = (V, E, N)$, can be perceived of as a *model* of the process represented by the Petri net. The *properties* to be verified can be expressed in some kind of mathematical formula φ. Temporal logic has proved to be useful to specify desired properties. An example of a temporal logic that has been used in this context is *Computation tree logic* (CTL). A description of CTL is beyond the scope of this course and can be found for instance in [2]. There is an algorithm for determining whether the occurrence graph $G = (V, E, N)$ has a property expressed by a CTL formula φ that runs in time $O(|\varphi| * (|V| + |E|))$. The usefulness of this algorithm is however limited because of the so called *state space explosion problem*. This problem is caused by the fact that the term $|V| + |E|$, representing the size of the occurrence graph, can grow exponentially in relation to the size of the net. For instance, an EN-System, the most basic of all Petri net models, with n places, has an upper limit to its state space of 2^n. In general, the analysis of 1-safe PT-nets turns out to be at least a PSPACE-hard problem in terms of the size of the net. For general PT-nets the problem is at least EXPSPACE-hard.[5] Extensions with colour and hierarchy allow PT-nets to be represented more compactly, so the complexity problem for HCP-nets will grow even worse.

A number of different approaches have been explored to overcome this problem. One approach is to try to reduce the occurrence graph. One way to do that is to look for groups of states and transitions in the OG that are similar in some sense and can be represented as one state and one transition (see for instance p. 39 in [8]). Partial order reduction is another approach to reduce the OG for instance by exploiting the fact that several occurrence sequences might represent the same partial order. Net-driven techniques approach the problem from another angle and is described in the next section.

Net driven techniques There are two types of net-driven approaches:

1. Net transformations
2. Structure theory results, based on graph theory and/or linear algebra

Net transformations are transformations that preserve the relevant properties of the net but change it in a way that makes it easier to analyze. This could for example be the reduction of redundant transitions. *Structure theory* has proved the existence of certain properties for restricted classes of Petri nets. In [1] a technique is presented to decide soundness of a particular class of Petri nets suitable for business process modelling, so called BP-nets, in polynomial time. Linear algebraic techniques for analyzing Petri nets is for instance described in [4].

References

[1] W. van der Aalst; *A class of Petri nets for modelling and analyzing business processes*, Computing science report 95/26, Tue, 1995

[2] E.M. Clarke, O. Grumberg, D. Peled; *Model checking*, MIT Press, 2001

[3] J.M. Colom, M. Silva, E. Teruel; *Properties*, pp. 53-72 in [6]

[4] J.M. Colom, M. Silva, E. Teruel; *Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems*, pp. 309-373 in [10]

[5] J. Esparza; *Decidability and Complexity of Petri Net Problems- An Introduction*, in [10]

[6] C. Girault, R. Valk; *Petri Nets for Systems Engineering*, Springer,2003

[7] K. Jensen; *Coloured Petri Nets; Basic Concepts, Analysis Methods and Practical Use; Vol. 1*, Springer-Verlag, 1992

[8] K. Jensen; *Coloured Petri Nets; Basic Concepts, Analysis Methods and Practical Use; Vol. 2*, Springer-Verlag, 1995

[9] A.V. Ratzer, L. Wells, H.M. Lassen; *CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets*, Department of Computer Science, University of Aarhus

[10] W. Reisig, G. Rozenberg, (Eds.); *Lectures on Petri nets: Basic Models*, *Lecture Notes in Computer Science*, 1491, Springer-Verlag, 1998