Handout I - ICTiB/Process modelling

Pieter Kwantes

January 9, 2010

Contents

1 Exercises **1**
- 1.1 Safe EN systems . . . 1
- 1.2 Modelling and analysis of EN-systems with Woped . . . 2
- 1.3 Calculating the size of the state space of an EN system . . . 4
- 1.4 Transition systems versus Petri nets . . . 4
- 1.5 Representation of Petri nets . . . 5
- 1.6 Running case . . . 6

2 Case description : Order processing at the LISTA Company **7**
- 2.1 Introduction . . . 7
- 2.2 The order process at LISTA . . . 7
- 2.3 The order process at the Sales Subsidiary . . . 8

3 Elementary Net systems **9**
- 3.1 Introduction . . . 9
- 3.2 Formal definition of EN-systems . . . 9

1 Exercises

1.1 Safe EN systems

In this Exercise class we will be using *Woped* as the tool for building Petri net models. Unfortunately Woped does not really acknowledge the only member of the Petri net family that we have learned about sofar: i.e. the *Elementary Net system* (EN-system). Woped does however enable us to model *Place transition systems*, another member of the Petri net family we will learn about in the next lecture. In this exercise we will learn how we can transform an EN system into a contact-free or *safe* EN system which is in fact the same as a *safe* Place transition system (i.e. a Place transition system where each place can have no more then one token).

1.1.1. *Understanding safe EN systems*

An EN-system is contact-free if in all reachable states of the reachability graph, for each transition in the EN system the following condition holds: if all the inputplaces of the transition hold (contain a token), then the outputplaces do not hold (are empty). If we look at the EN-system shown on page 10 of this handout then we can see that this is not a contact-free EN-system.

question a) Explain why.

Now look at the formal definition of the **enabling** of a transition in an EN-system on page 10. You see that there are in fact **two** conditions that must be fullfilled for a transition to be enabled.

question b) Would **both** conditions, just mentioned, still be necessary in a **contact free** or **safe** EN system? If not, then which conditions might be redundant?

1.1.2. *Making EN systems contact-free*

We can make EN systems contact free by adding so called **complement places**. We can accomplish this by doing the following :

1. For each original place in the net we create a complement place. That means we add a place with a token if the original place has no token and we add a place without a token if the original place has a token.
2. For each transition to which the original place is an inputplace, we create an arc that will make the complement place an outputplace of that transition.

3. For each transition to which the original place is an outputplace, we create an arc that will make the complement place an inputplace of that transition.

Then we will have a contact-free, i.e. safe, EN system and in fact also a safe PT system.

Because Woped **only** checks the inputplaces to determine whether a transition is enabled, and **not** its outputplaces we can only model EN systems in Woped if they are safe EN systems.

question a) explain in your own words why this would be so.

Using the concept of safe EN systems, we are now able to create EN-systems in Woped. This will be done in the next exercise.

1.2 Modelling and analysis of EN-systems with Woped

1.2.1. Creating a Petri net model in Woped In this exercise you will create the net as shown in figure 5 on page 10 in Woped. Woped is an easy to use Petri net editor. You can find a lot of useful information about Woped under the help menu. For a quick start, just follow the steps summarized below:

Step 1: Open a new net. After you started Woped, you select the *file submenu* from the menu bar and then you select the menu option *new*. Woped will then present to you an empty workspace in the editor window that allows you to create a new Petri net.

Step 2: Adding places and transitions. In the toolbar, just below the menu bar (see figure 1), you will see a number of icons representing *net elements* that you can use to create a Petri net model.

Figure 1: Net elements available in Woped

From the icons shown in figure 1 you should only use the two at the top, to create a *transition* and to create a *place*. The other ones (with the exception of the one at the bottom) are only for "syntactic sugaring" and can be created by using just the two elements just mentioned. Now add the places and transitions you need to create the Petri net shown in figure 5.

Step 3: Connect the places and transitions , that you created in step 2, as shown in figure 5. You can create connections or *arcs* as follows:

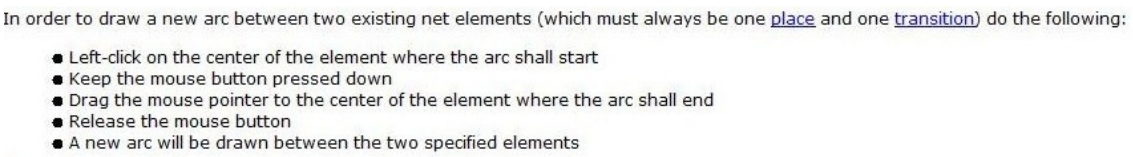

Figure 2: Drawing an arc in Woped

Step 4: Add the initial marking as shown in figure 5. You can add a marking by creating tokens as follows:

Figure 3: Creating tokens in Woped

1.2.2. Executing a model After you created the net, in exercise 1.2.1., you can execute it using the menu option in Woped *start tokengame*. Use the option in the toolbar to fire the transitions manually one by one (see the help menu for a detailed explanation of the *tokengame* option). From exercise 1.1.1. you already know that the net is not safe. Now you can see what happens if you execute an unsafe Petri net. Try to explain what you see and why you might have a problem analyzing the behaviour of unsafe Petri nets (Hint: try to answer the question how many different states the reachability graph of this Petri net will have).

1.2.3. Making the net safe. Add a place to the net in figure 5 that is the complement of place $p3$ and build the resulting contact-free EN-system, i.c. safe PT net, in Woped.

1.2.4. Generating the reachability graph Generate the reachability graph of the safe net you created in exercise 1.2.3.. Explain how you can use the reachability graph to verify whether the net is safe or not.

1.3 Calculating the size of the state space of an EN system

1.3.1. Create the Petri net model of exercise 2.4. ("Binary counter") in the book "Workflow Management, Models, Methods and Systems" with Woped.

1.3.2. Extend the model of the binary counter from exercise 5. so that it can count to 15.

1.3.3. State the relationship between the number of places in the Petri net model of the binary counter and how far the counter can count.

1.4 Transition systems versus Petri nets

In computer science so-called *transition systems* are often used to describe the behavior of systems. A transition system basically is a directed graph where the nodes represent *states* and the edges model *state changes*. The figure below shows a transition system that models the behaviour of a dog, which is either quiet or is barking.

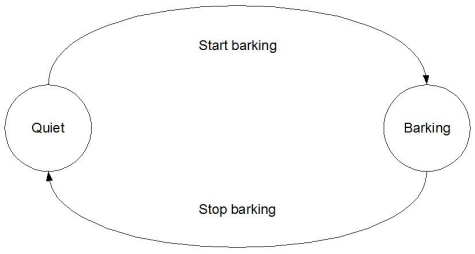

Figure 4: Transition system modelling the behaviour of a dog

In fact we have already seen, in the first lecture, a transition system of a traffic light: this was the reachability graph of the Petri net model of the traffic light.

1.4.1. Create a transition system that models the behaviour of the dog standing beside the traffic light. We must thereby assume that the behaviour of the dog is completely independent of the traffic light.

1.4.2. Create two Elementary net systems, one that models the behaviour of the dog, and the other of the traffic light (which you already have seen in the lecture sheets).

1.4.3. Compare the models in exercise 1.4.1 and 1.4.2. Which model do you think is the best and most convenient representation of the real world? Explain why.

1.4.4. Create the model in exercise 1.4.2. (containing two EN-systems) in Woped. Generate the reachability graph of the model. Compare the resulting reachability graph with the model you created in exercise 1.4.1. and explain what you see.

1.5 Representation of Petri nets

Petrinets can be represented by a graphical as well as a mathematical notation. The graphical representation of the net you created in exercise 1.2.1. is shown in figure 5 on page 10 and the mathematical representation of that same net is

shown on page 9. For this exercise you are asked to write down the mathematical representation of the Petri net you created in exercise 1.2.3.

1.6 Running case

In this exercise you will set the first steps to make a formal model of the order process at Lista as described in the case documentation. A top-down modelling approach seems appropriate. For this exercise you will execute the following steps:

1. **Read Case description** in section 2.

2. **Identify relevant internal departments and external parties** This step involves that you identify all the departments and external parties that participate in the order process at LISTA given in the case description.

3. **Identify dependencies between departments** This step involves finding the dependencies between the activities that are executed by the parties. A dependency can for example consist of some information send by one department to another. You can start by finding the information flows between the departments and external parties. A dependency can also be the *order* in which activities are executed.

4. **Draw dataflow diagram** draw a dataflow diagram representing the communication between the departments and external parties involved in the execution of the Order process at the Sales Subsidiaries of LISTA, in a top-down fashion. So, for example, the highest level of you dataflow diagram could show an entity "LISTA" and an entity "Customer"'that exchange information.

2 Case description : Order processing at the LISTA Company

2.1 Introduction

The case study about the LISTA company is selected as a running case for this course about *business process modelling*. The LISTA company already featured in the course "ICT enabled process innovation", given earlier this academic year, so you should already be familiar with this case. The documentation on the LISTA case, will be published again on blackboard. The running case exercises given during this course will focus specifically on the order process at LISTA. In the available case documentation you can find that the order process was identified as one of the processes eligible for re-engineering to meet the targets that were defined in the strategic reorientation that was decided upon by the management team of LISTA.

The running case in this course will involve making a Petri net model of the order process at LISTA. Based on that model you will then do a performance analysis of the order process as it is currently organized. This will then serve as a benchmark for proposing improvements to the process. This proposal can be the basis for a Petri net model and performance analysis of the improved process. But before you can start modelling you must try to understand what the order process at LISTA actually entails.

2.2 The order process at LISTA

The starting point for analysis of the current order process is the process chart that is available which describes the processing of an order at the LISTA company. It describes the *individual tasks* within the order process, *the ordering* of the tasks with the process and the *routing* of an order through the order process. Further more it describes the *actors*, i.e. departments or external parties, that are involved in this order process. You should try to understand the order process at LISTA by investigating the process chart and read the explanation about the order process that is given below.

If you look a the process chart of the order process at LISTA, you will see that the order process can be considered as consisting of *three* different "versions" of the order process. There is an order process executed by the internal department of LISTA called *Sales Subsidiary* , there is an order process executed by the external party *Independent Dealer (ID)* and there is an order process executed by the internal department of LISTA called *Key Account*. We will focus our attention on the order process at the Sales Subsidiary.

2.3 The order process at the Sales Subsidiary

LISTA has twelve Sales Subsidiaries (*SS*), distributed over different locations. Each *SS* can be considered as consisting of two subsections:

- an *Internal Service* (*SS internal*), responsible for operating a showroom for product demonstrations and reception of customers and for a number of administrative tasks associated with the order process.
- a *Field Service* (*SSFS*), which is highly mobile, not restricted to a fixed location and takes care of most of the communication between the customer and the other LISTA departments involved in the order process.

The order process can be initiated by the Customer, either by calling directly to the *SS Internal* or by visiting one of the showrooms maintained by the *SS Internal*. Alternatively, the order process can be initiated by the *SSFS*, which actively monitors public announcements for development of new office space for potential market demand and calls potential new Customers by phone. Each of these three options for initiating a Customer order, might result in the preparation of basic sales documents for an offering to the Customer. The potential order can subsequently follow a number of alternative routes through order process and the LISTA organisation, as described in the process chart, before it ends up as an order, accepted by the Customer, and entered into the production system of LISTA to be handled further by the manufacturing department of LISTA.

3 Elementary Net systems

3.1 Introduction

Elementary Net systems (EN-systems) are the most basic of all Petri Net models discussed in this course. Like finite automata, EN-systems can be viewed as an abstract model of computation. A crucial difference however is that the state of the EN-system is distributed over a set of *local* states and each action in the EN-system is only influenced by, and influences, the local state in its direct neighbourhood. This allows the modelling of concurrent systems in a natural way. The state of each concurrent system is represented by local states. Synchronization of subsystems can be modelled by having these systems shared transitions that change the local states of each system simultanuously. While the finite automaton is a model of sequential systems, EN-systems are a model of concurrent systems. In the following section we will give a formal description of Elementary net systems.

3.2 Formal definition of EN-systems

In the following paragraphs we will give a formal description of respectively the structure, the dynamics and the state space of an EN-system.

Structure of EN-Systems The structure of an EN-system is defined by a quadruple :

$ENS = (P, T, A, I)$

where

1. P is a finite set of places
2. T is a finite set of transitions,
3. $A \subset (P \times T) \cup (T \times P)$ is a set of arcs between places and transitions.
4. $I : P \rightarrow \{true, false\}$ is an initialization function.

A marking M of an EN-system is a function : $M : P \rightarrow \{true, false\}$. Markings describe global states consisting of places that hold. In ENS the initialization function I defines the initial marking of the system.

A simple example of a formal definition of an elementary net system might be :

$P = \{p1, p2, p3, c1, c2\}$
$T = \{t1, t2, t4, t5\}$
$A = \{(p1, t1), (p2, t2), (p3, t4), (c1, t4), (c2, t5)\} \cup$
$\{(t1, p2), (t2, p1), (t2, p3), (t4, c2), (t5, c1))\}$

$I = \{(p1, false), (p2, true), (p3, false), (c1, true), (c2, false)))\}$

This EN-system can also be given a nice graphical representation, as can be seen in figure 5 below.

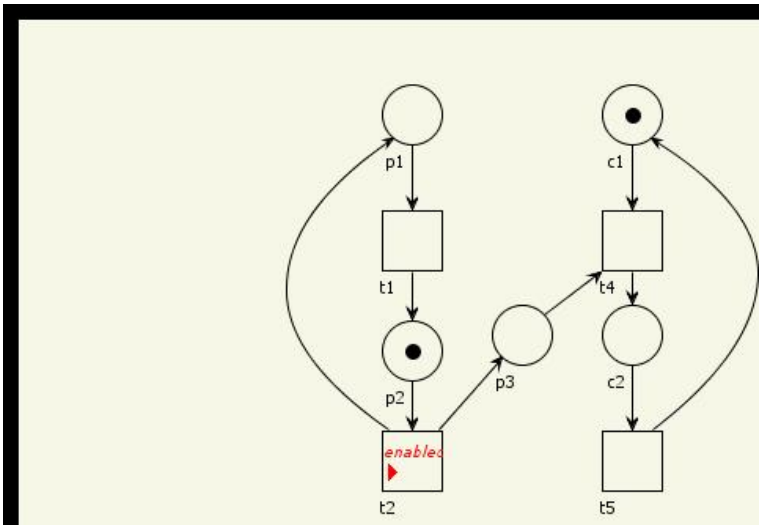

Figure 5: EN-system of a producer-consumer process

The circles in figure 5 represent the places of the EN-system, the squares represent the transitions and the arrows represent the arcs. Markings are pictured by using dots, often called "tokens", which indicate the places that hold.

An EN-system ENS=(P,T,A,I) as given above specifies the static part of the ENS. The underlying net (P,T,A) constitutes a bipartite directed graph, where the places and transitions are the nodes of the graph connected by arcs. The transitions represent activities while the input places of a transition $t \in T$, i.e. the places $p \in P$ where $(p, t) \in A$, can be seen as the preconditions for this activity to occur. Similarly, the output places can be seen as the postconditions for this activity. A dot or token in a place indicates that the pre- or postcondition it represents is true. If a postcondition of a transition is true, then this prevents the transition or activity from occurring. The formal description of the dynamics is given below.

Dynamics of EN-systems The occurrence of a transition in an EN-system only depends on and only influences the local states in its neighbourhood.

Let $In(t) = \{p \in P | (p, t) \in A\}$ be the set of input places of transition t

and $Out(t) = \{p \in P | (t,p) \in A\}$ the set of output places of t.
Then,

1. A transition t is **enabled** in a marking M iff the following property is satisfied :

$$\begin{aligned} &\forall p \in In(t) && M(p) = true \quad \wedge \\ &\forall p \in Out(t) && M(p) = false. \end{aligned}$$

2. When a transition t is enabled in a marking M it may **occur**, changing the marking from M to $M^{'}$, defined by:

$$\begin{aligned} &\forall p \in In(t) && M^{'}(p) = false \wedge \\ &\forall p \in Out(t) && M^{'}(p) = true \wedge \\ &\forall p \in P - (In(t) \cup Out(t)) && M(p) = M^{'}(p) \end{aligned}$$

Thus the occurrence of t leads to the non-holding of its input places while its output places begin to hold. This is illustrated by the figures 5 and 6 respectively.

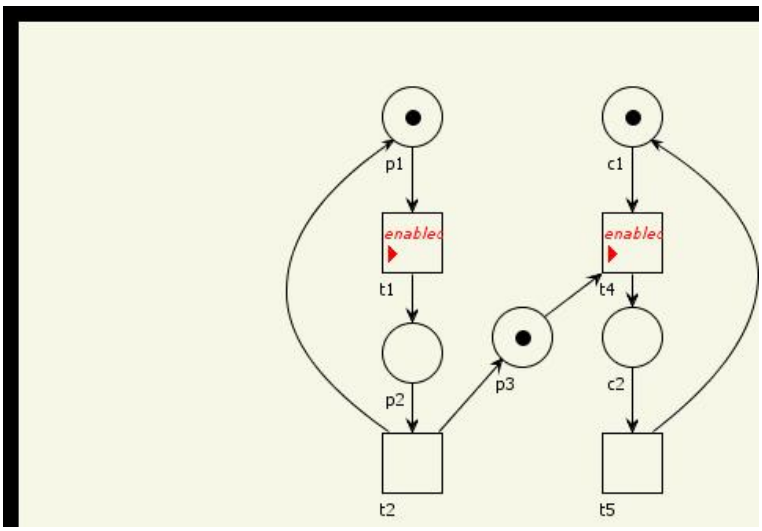

Figure 6: producer-consumer process after firing transition t2

The marking $M^{'}$ is said to be **directly reachable** from marking M.

Two transitions in an EN-system can occur concurrently if their neighbourhoods are disjoint. This is defined using the concept of a **step**:

1. A step is a subset of T.

2. A step Y is enabled in a marking M iff the following properties are satisfied:

   (a) all $t \in Y$ are *disjoint*, i.e. :

   $$\begin{aligned}&\forall t_1 \neq t_2 \in Y : \\ &[In(t_1) \cap In(t_2) = \emptyset \ \wedge Out(t_1) \cap Out(t_2) = \emptyset \ \wedge \\ &\ In(t_1) \cap Out(t_2) = \emptyset \ \wedge In(t_2) \cap Out(t_1) = \emptyset]\end{aligned}$$

   (b) all $t \in Y$ are *enabled*

3. When a step Y is enabled in a marking M it may **occur**, changing the marking from M to $M^{'}$, defined by :

$$\begin{aligned}\forall t \in Y \quad (\forall p \in In(t) \quad M^{'}(p) = false \wedge \forall p \in Out(t) \quad M^{'}(p) = true \quad \wedge \\ \forall p \in P - (\bigcup_{t \in Y} In(t) \cup \bigcup_{t \in Y} Out(t)) \quad M(p) = M^{'}(p))\end{aligned}$$

Reachability graph of EN-Systems The reachability graph of an EN-system is a representation of the state space of the system by a graph and can be considered a representation of its behaviour. Each node represents a reachable marking and each arc a step that can occur in that marking. If a state is directly reachable from another state there will be an edge connecting the nodes that represent these two states. The reachability graph can be used to analyze the behaviour of the net. A formal definition of the reachability graph is given below.
A **directed graph** is a tuple $DG = (V, E, N)$ where :

1. V is a set of nodes (or vertices).

2. E is a set of arcs (or edges), where $V \cap E = \emptyset$

3. N is a node function. It is defined from E into $V \times V$

DG is finite iff V and E are finite.
The reachability graph of an EN-system $ENS = (P, T, A, I)$ is defined as a directed graph $OG = (V, E, N)$ where :

1. $V = [M_1\rangle$ where $\forall p \in P \quad M_1(p) = I(p)$.

2. $E = \{(M_1, t, M_2) \in (V \times T \times V) | M_1[t\rangle M_2\}$

3. $\forall e = (M_1, t, M_2) \in E : N(e) = (M_1, M_2)$

Since the set of places P and the set of transitions T are both required to be finite, the set of all possible markings and steps of ENS must also be finite. Therefore the occurrence graph is also finite. The reachability graph for an EN-system can be constructed by the following algorithm :

1. $V := \{M_1\}$; $E := \emptyset$; M_1 is untagged;
2. **while** there are untagged nodes in V **do** :
   (a) Select an untagged node $M \in V$ and tag it
   (b) **for** each enabled transition, t, at M, **do**
      i. Compute $M^{'}$ such that $M[t\rangle M^{'}$;
      ii. $V := V \cup \{M^{'}\}$;
      iii. $E := E \cup \{(M, t, M^{'})\}$