

# Resource Allocation

## **Abstract**

This is a small toy example which is well-suited as a first introduction to occurrence graphs. The analysis of the occurrence graph is described in great detail, explaining the basic concepts of occurrence graphs. Hence, it can be read by people with no prior knowledge of occurrence graphs.

The CPN model describes how two different kinds of processes are sharing three different kinds of resources. The model is identical to the “Resource Allocation” system presented in “Introductory Examples” (which we recommend to study before this example).

The example is taken from Sect. 1.1 of Vol. 2 of the CPN book.

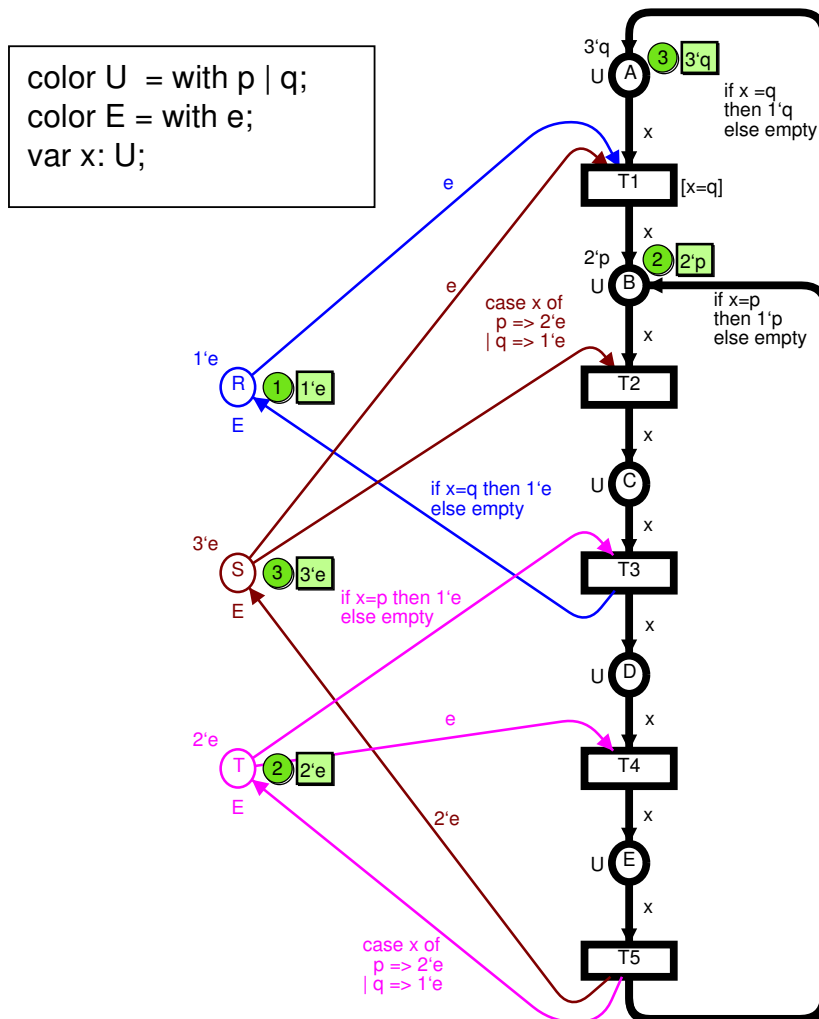
## **Developed and Maintained by:**

Kurt Jensen, Aarhus University, Denmark ([kjensen@daimi.au.dk](mailto:kjensen@daimi.au.dk)).

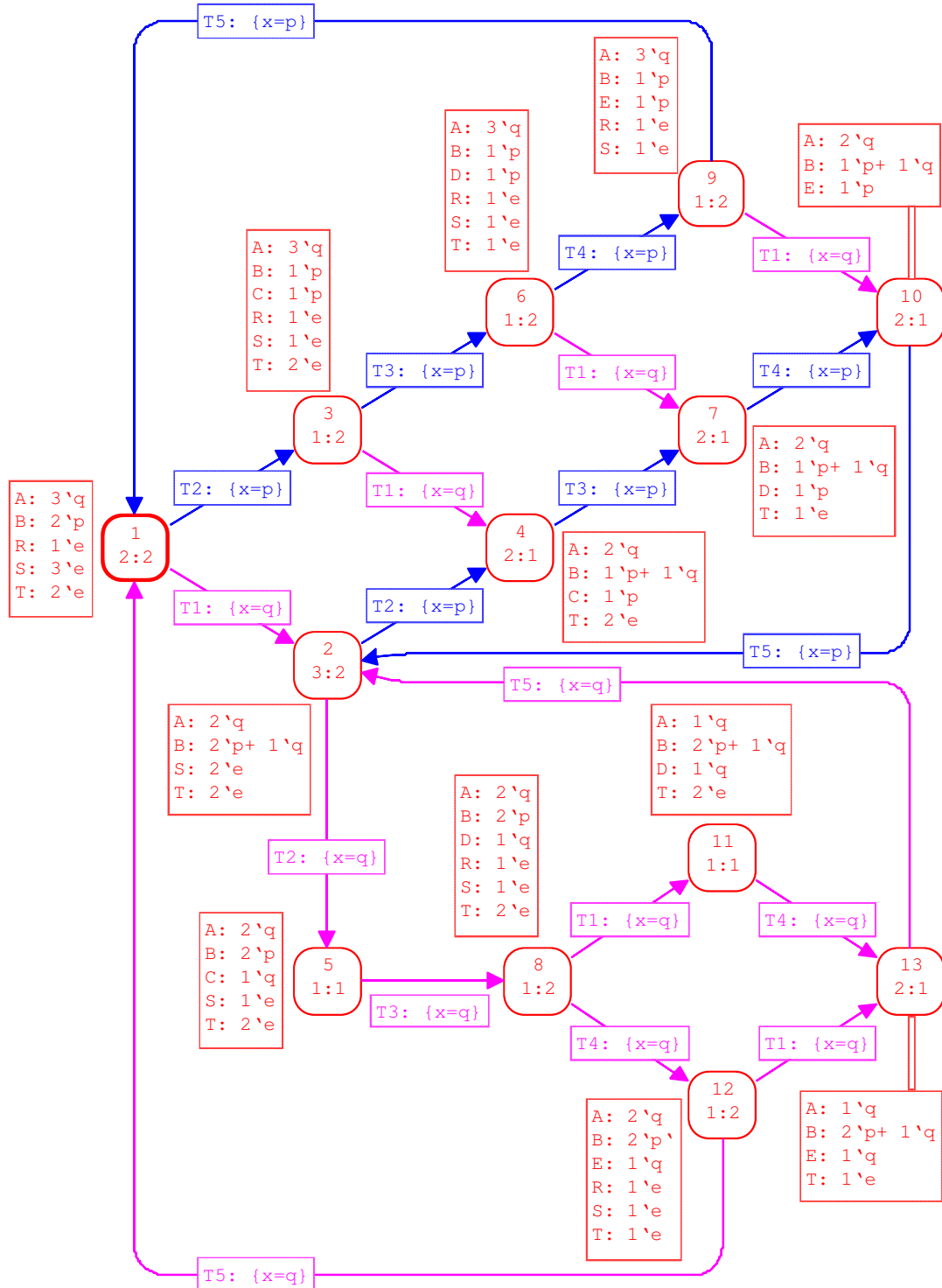
## CPN Model

The basic idea behind occurrence graphs is to construct a graph which has a node for each reachable marking and an arc for each occurring binding element. Obviously, such a graph may become very large, even for small CP-nets. As an example, let us consider again the “Resource Allocation” system from the “Introductory Examples”. Due to the cycle counters this net has an infinite number of reachable markings and thus an infinite occurrence graph. However, we can simplify the CP-net by omitting the cycle counters. Then we get the CP-net shown below.

It is easy to check that the cycle counters form an isolated part of the original CP-net – in the sense that they influence neither the enabling nor the effect of an occurrence (except that they determine the values of new cycle counters). This means the simplified net has a behaviour similar to that of the original net. For each occurrence sequence in one of the CP-nets there is a corresponding occurrence sequence in the other. Hence we can get information about the dynamic properties of the original net by constructing an occurrence graph for the simplified net.



Such a graph is shown below – it is called a **full occurrence graph** or an **O-graph**. The current version of CPN Tools does not include facilities for drawing O-graphs. The rounded boxes are nodes. Each of them represents a reachable marking, and the content of this marking is described in the dashed region next to the node – places with an empty marking are omitted. To the left



we have a node with a thicker borderline. This node represents the initial marking. The text inside the node tells us that this is node number 1 and that it has 2 predecessors and 2 successors (the latter information may be useful when we have drawn only a small part of a large occurrence graph). Analogously, we see that node #2 has 3 predecessors and 2 successors. By convention we use  $M_n$  to denote the marking of node number  $n$ .

Each arc represents the occurrence of the binding element in the dashed region on top of the arc. In  $M_1$  there are two enabled binding elements. If transition  $T_1$  occurs, with  $x$  bound to  $q$ , we reach  $M_2$ , and if transition  $T_2$  occurs, with  $x$  bound to  $p$ , we reach  $M_3$ .

Notice that we omit arcs that correspond to steps containing more than one binding element. Otherwise, we would have had, e.g., an arc from node #1 to node #4, representing the step  $1'(T_1, \langle x=q \rangle) + 1'(T_2, \langle x=p \rangle)$ . Such arcs would give us information about the concurrency between binding elements, but they are not necessary for the verification of boundedness, home, live and fairness properties.

When the occurrence graph and the strongly connected component (SCC) graph have been generated, we can ask the state space tool to generate a **standard report**, i.e., a text file with key information about the occurrence graph and the dynamic properties which can be deduced from it. The standard report has five parts. The first part looks as shown below. It contains statistical information about the size of the occurrence graph and the size of the SCC-graph (which has a node for each strongly connected component of the occurrence graph). We see that the occurrence graph has 13 nodes and 20 arcs. We have calculated the entire graph and this took less than 1 second. Finally, we see that there is only one strongly connected component.

```

Statistics
-----
Occurrence Graph
Nodes:  13
Arcs:   20
Secs:   0
Status: Full

Scc Graph
Nodes:  1
Arcs:   0
Secs:   0

```

The second part of the standard report contains information about the integer and multi-set bounds. First we get the upper and lower integer bounds, i.e., the maximal and minimal number of tokens on the individual places. As, an example, we see that place A always has 1-3 tokens. We also see that each of the places C, D and E has at most one token. Next we get the upper and lower multi-set bounds. From the upper multi-set bounds, we see that place A only can have  $q$ -tokens. We also see that each of the places B–E can have both  $p$ -tokens and  $q$ -tokens. From the lower multi-set bounds, we learn that place A always contains at least one  $q$ -token while place B always contains at least one  $p$ -token. We also see that we cannot omit any of the resources without changing the behaviour of the system.

Boundedness Properties		
-----		
Best Integers Bounds		
	Upper	Lower
A	3	1
B	3	1
C	1	0
D	1	0
E	1	0
R	1	0
S	3	0
T	2	0
Best Upper Multi-set Bounds		
A	3 `q	
B	2 `p+ 1 `q	
C	1 `p+ 1 `q	
D	1 `p+ 1 `q	
E	1 `p+ 1 `q	
R	1 `e	
S	3 `e	
T	2 `e	
Best Lower Multi-set Bounds		
A	1 `q	
B	1 `p	
C	empty	
D	empty	
E	empty	
R	empty	
S	empty	
T	empty	

The third part contains information about the home properties. Here we see that all reachable markings are home markings. This means that they all can be reached from each other.

```

Home Properties
-----
Home Markings:  All

```

The fourth part contains information about liveness properties. We see that there are no dead markings and that all transitions are live, which means that they always have the possibility of occurring once more.

```

Liveness Properties
-----
Dead Markings:  None
Dead Transitions Instances:None
Live Transitions Instances: All

```

The fifth and final part of the standard report contains information about the fairness properties. Here we see that each of the transitions T2-T5 is impartial, which means that each infinite occurrence sequence contains an infinite number of the transition. Transition T1 is neither impartial, fair, or just. From the drawing of the occurrence graph we can see why this is the case. By repeating the cycle through the nodes #1, #3, #6 and #9, we get an infinite occurrence sequence. T1 is enabled in every marking of this occurrence sequence, but T1 never occurs in the sequence.

```

Fairness Properties
-----
T1      No Fairness
T2      Impartial
T3      Impartial
T4      Impartial
T5      Impartial

```

The standard report is produced in a few seconds – totally automatic. The standard report contains a lot of highly useful information about the behaviour of CPN model. However, we may also want to verify some properties which are more particular for the model at hand.

As an example, we may want to see how many tokens the places C–E has together. This is done by formulating the simple 4-lines query shown in left-hand box below. The function *Mark.Top'C 1* allows us to determine the marking  $M(C)$  of the *first* instance of place *C* on page *Top*. Analogously, the

next lines determine the markings  $M(D)$  and  $M(E)$ , which are added to  $M(C)$ . The function *UpperInteger* calculates the maximal number of tokens in  $M(C)+M(D)+M(E)$  when  $M$  traverses the set of markings in the occurrence graph (i.e., all reachable markings). The result is shown in the rounded box to the right. It is 1, and this tells us that the places C, D and E form a critical region. There is never more than one process in this area – at a time.

#### Critical Region

```
UpperInteger( fn node =>
  (Mark.Top'C 1 node) ++
  (Mark.Top'D 1 node) ++
  (Mark.Top'E 1 node));
```

val it = 1 : int

From the standard report we know that all five transitions are live. However, we may also want to know whether they are strictly live, i.e., whether each individual binding element is live. To check this, for transitions T1 and T2, we make the following queries, which tell us that both transitions are strictly live. It should be noted that transition T1 only has one possible binding – due to the guard.

#### Strict Liveness

```
BESStrictlyLive
[Bind.Top'T1 (1, {x=q})];
```

val it = true : bool


```
BESStrictlyLive
[Bind.Top'T2 (1, {x=p}),
 Bind.Top'T2 (1, {x=q})];
```

val it = true : bool


From the standard report we know that transition T1 possesses no fairness property while the only four transitions are impartial. However, we may also want to investigate the fairness properties for the sets of those binding elements that correspond to q-processes and p-processes, respectively. To do this, we make the following queries, which tell us that the set of binding elements of p-processes is just while the set of binding elements of q-processes possesses no fairness property.

**Fairness**

```
BEsFairness
[Bind.TopT2 (1, {x=p}),
 Bind.TopT3 (1, {x=p}),
 Bind.TopT4 (1, {x=p}),
 Bind.TopT5 (1, {x=p})];
```



```
BEsFairness
[Bind.TopT1 (1, {x=p}),
 Bind.TopT2 (1, {x=p}),
 Bind.TopT3 (1, {x=p}),
 Bind.TopT4 (1, {x=p}),
 Bind.TopT5 (1, {x=p})];
```



Even for a small O-graph, like the one in this example, the construction and investigation are tedious and error-prone. In practice, it is not unusual to handle CP-nets that have O-graphs containing more than 100,000 nodes (and many CP-nets have millions of markings). Thus it is obvious that we could not work with occurrence graphs without tool support.