

Timed Resource Allocation

Abstract

This is a small toy example which is well-suited as a first introduction to timed CP-nets. It shows how the CP-net from “Resource Allocation” can be turned into a timed CP-net. The time constructs are described in great detail, explaining the basic concepts of timed CP-nets.

The example is taken from Sect. 5.1 of Vol. 2 of the CPN book and Sect. 6.2 of Vol. 1 of the CPN book.

Developed and Maintained by:

Kurt Jensen, Aarhus University, Denmark (kjensen@daimi.aau.dk).

Graphical Quality

The figures in this document are inserted via PICT format. This is why some of the arcs and place borders look a bit ragged. A postscript printout from Design/CPN (and the screen image in Design/CPN) has much higher graphical quality.

CPN Model

To investigate the performance of systems, i.e., the speed at which they operate, it is convenient to extend CP-nets with a time concept. To do this, we introduce a **global clock**. The clock values represent the **model time**, and they may either be discrete (e.g., integers) or continuous (e.g., reals). In addition to the token colour, we allow each token to carry a **time value**, also called a **time stamp**. Intuitively, the time stamp describes the *earliest* model time at which the token can be used, i.e., removed by a binding element.

In a timed CP-net a binding element is said to be **colour enabled** when it satisfies the requirements of the usual enabling rule (for untimed CP-nets). However, to be **enabled**, the binding element must also be **ready**. This means all the time stamps of the removed tokens must be less than or equal to the current model time.

To model that an activity/operation takes r time units, we let the corresponding transition t create time stamps for its output tokens that are r time units larger than the clock value at which t occurs. This implies that the tokens produced by t are unavailable for r time units. It can be argued that it would be more natural to delay the creation of the output tokens, so that they did not come into existence until r time units after the occurrence of t had begun. However, such an approach would mean that a timed CP-net would get “intermediate” markings which do not correspond to markings in the corresponding untimed CP-net, because there would be markings in which input tokens have been removed but output tokens not yet generated. Hence we would get a more complex relationship between the behaviour of timed and untimed nets.

The execution of a timed CP-net is time driven, and it works in a similar way to that of the event queues found in many programming languages for discrete event simulation. The system remains at a given model time as long as there are colour enabled binding elements that are ready for execution. When no more binding elements can be executed, at the current model time, the system advances the clock to the next model time at which binding elements can be executed. Each marking exists in a closed interval of model time (which may be a point, i.e., a single moment). The occurrence of a binding element is instantaneous.

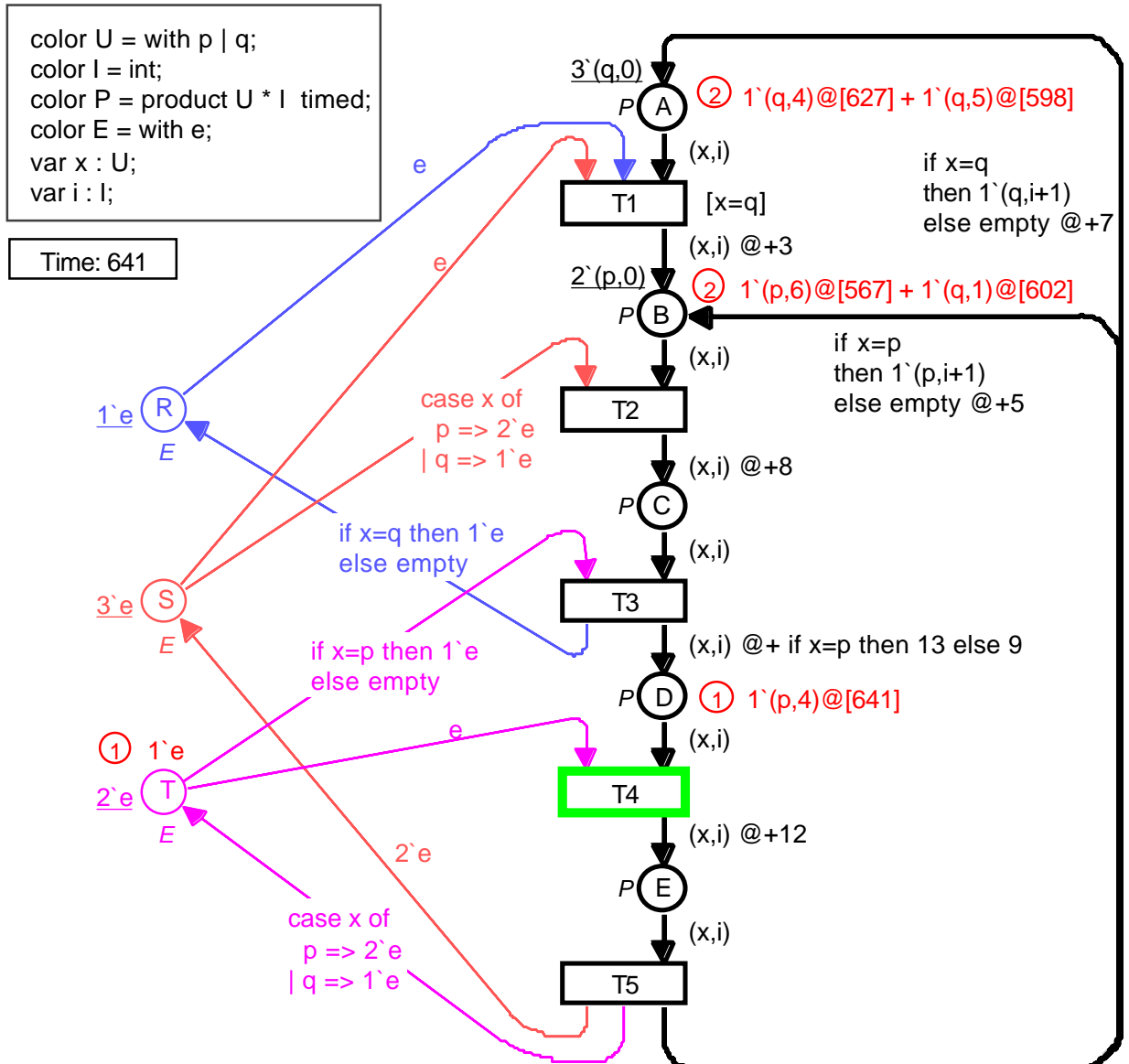
Now let us consider the timed CP-net for the resource allocation system (shown below). For this system, we use a discrete clock, starting at 0. From the third and fourth lines of the declarations, we see that P-tokens are timed (i.e., carry time stamps), while the E-tokens are not. This means E-tokens are always ready to be used. The small rectangle below the declarations indicates that the current model time is 641 (in the CPN simulator this information is displayed in the status bar).

The @ signs in the current markings should be read “at”. Each @ sign is followed by a list of time stamps. The marking of place A contains two tokens, one with colour (q,4) and time stamp 627, and one with colour (q,5) and time stamp 598. Analogously, place B has two tokens, one with colour (p,6) and time stamp 567, and one with colour (q,1) and time stamp 602. Place D has a single token with colour (p,4) and time stamp 641. Finally, place T has one token with colour e and no time stamp. In the shown marking all lists of time stamps have length 1,

because all the tokens have different colours. However, the initial marking of place A is displayed as $3^{\text{'}}(q,0)@[0,0,0]$, while the initial marking of B is $2^{\text{'}}(p,0)@[0,0]$.

Now let us consider the steps which may occur in the shown marking. The binding element $b_1 = (T4, \langle x=p, i=4 \rangle)$ is colour enabled, because the two input places have the necessary tokens. The binding element is also ready, because all the time stamps of the removed tokens are smaller than or equal to the current model time (in this case there is only one such time stamp and it is equal to the current model time). Hence b_1 is enabled and it may occur. The occurrence of b_1 removes the token from D and the token from T. The occurrence of b_1 will also add a token to E. The colour of the new token is calculated in the usual way, i.e., as specified by the occurrence rule for untimed CP-nets. The time stamp of the new token is calculated as the current model time plus a time delay, which is specified in the corresponding output arc expression – after the $@+$ operator. In this case the delay is 12 time units, and hence we get 653 as the new time stamp. Intuitively, this means the p-token must stay at least 12 time units at place E. We can interpret this to mean that

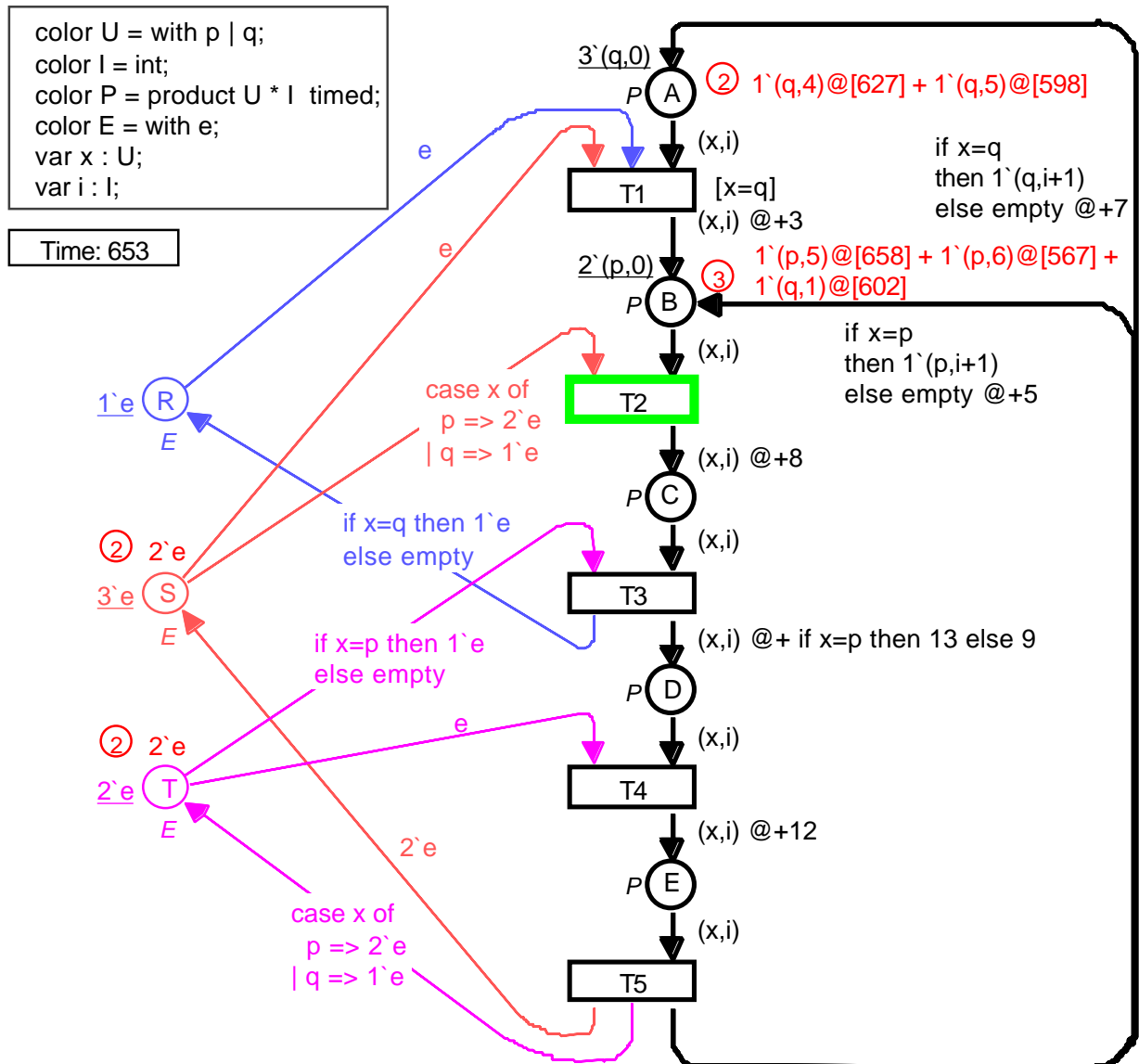
Marking at time 641



the state E has a minimal duration of 12 time units. We can also interpret it to mean that the activity T4 takes 12 time units.

When b_1 has occurred, we reach a marking in which $b_2 = (T5, \langle x=p, i=4 \rangle)$ is the only colour enabled binding element. However, b_2 is not ready at model time 641 because it involves a token with a too-high time stamp. Hence we increase the model time until b_2 becomes ready, i.e., to 653. If there had been several colour enabled binding elements we would have increased the model time until one of them became ready. When b_2 has occurred, at model time 653, we get the marking shown below. Now we have three colour enabled binding elements $b_3 = (T2, \langle x=p, i=5 \rangle)$, $b_4 = (T2, \langle x=p, i=6 \rangle)$, and $b_5 = (T2, \langle x=q, i=1 \rangle)$. The binding elements b_4 and b_5 are ready at 653, while b_3 uses a token with time stamp 658. Hence either b_4 or b_5 will occur. They are in conflict with each other, because they both need e-tokens from S. This means only one of them will be executed. However, had there been an additional e-token on S, b_4 and b_5 would have been concurrently enabled and both of them would have occurred at time 653 (either in the same step, or in two subsequent steps).

Marking at time 653



The time delays may depend upon the binding, i.e., upon the colours of the input and output tokens. This is illustrated by the output arc of T3, where the delay depends upon the value of the variable x . For p -processes the delay is 13, while for q -processes it is 9. The delays are specified by means of expressions, and this means, for example, they can use functions which implement complex statistical distributions.

For a timed CP-net we require that each step consists of binding elements which are both colour enabled and ready. Hence the possible occurrence sequences of a timed CP-net always form a subset of the possible occurrence sequences of the corresponding untimed CP-net. This means we have a well defined and easy-to-understand relationship between the behaviour of a timed CP-net and the behaviour of the corresponding untimed CP-net.

In the resource allocation system, we have only illustrated one of the simplest ways in which time stamps can be used. All removed tokens for a binding element (t,b) were required either to be without time stamps or to have time stamps which were less than or equal to the time value r^* at which (t,b) occurs. At a given place p , all added tokens for (t,b) either got no time stamps or got identical time stamps which were equal to r^* plus a delay r . In general, the situation can be considerably more complex.