Worst-case Analysis of Concurrent Systems with Duration Interval Petri Nets

Louchka Popova-Zeugmann
Humboldt University
Dep. of Computer Science
Axel-Springer-Straße 54 a
D - 10099 Berlin
Tel.: (030) 20 18 12 73
Fax: (030) 20 18 12 87
popova@informatik.hu-berlin.de

Monika Heiner
Brandenburg University of Technology
Dep. of Computer Science
Karl-Marx-Straße 17
D - 03013 Cottbus
Tel.: (0355) 69 38 85
Fax: (0355) 69 38 30
mh@informatik.tu-cottbus.de

Abstract: This paper deals with computing the minimal and maximal execution durations in a given concurrent control system in order to support dependability engineering by assuring the meeting of prescribed deadlines. For that purpose, a new type of time-dependent Petri nets - the Duration Interval Petri net - is introduced, and a dedicated reachability graph is defined in a discrete way. Using this reachability graph, shortest and largest time paths between two arbitrary states of the control system, and by this way minimal and maximal execution times, can be computed.

Keywords: system validation, qualitative and quantitative analysis, performance evaluation, worst-case analysis, time-dependent Petri nets, control systems.

1 Petri Net Based Methods to Improve System Dependability

Among those methods, which aim at the improvement of the dependability of any system, different kinds of Petri net based validation techniques to avoid faults during the development phase have attracted a lot of attention in the last decade. Within this general framework various Petri net based methodologies of dependability engineering have been outlined. At the beginning, only qualitative properties have been discussed. But because of the crucial impact of performance on parallel or distributed (shortly called concurrent) systems, special emphasis has been laid more and more on incorporation of performance criteria as part of the system development cycle [Balbo 92], [Heiner 94], [Donatelli 94]. Moreover, some situations are well-known in different kinds of hard real-time systems, making worst-case analysis unavoidable. E.g. if a process is caught in a livelock (i.e. blocked for longer than some critical time period), it may have the same consequences as if it were involved in a deadlock (i.e. blocked for ever).

Therefore, maybe the most important advantage of the Petri net approach to dependability engineering is its ability to combine qualitative analysis, monitoring and testing as well as quantitative analysis (in terms of performance/reliability prediction and worst-case analysis) on the basis of a common Petri net-based intermediate representation of the concurrent system under development.

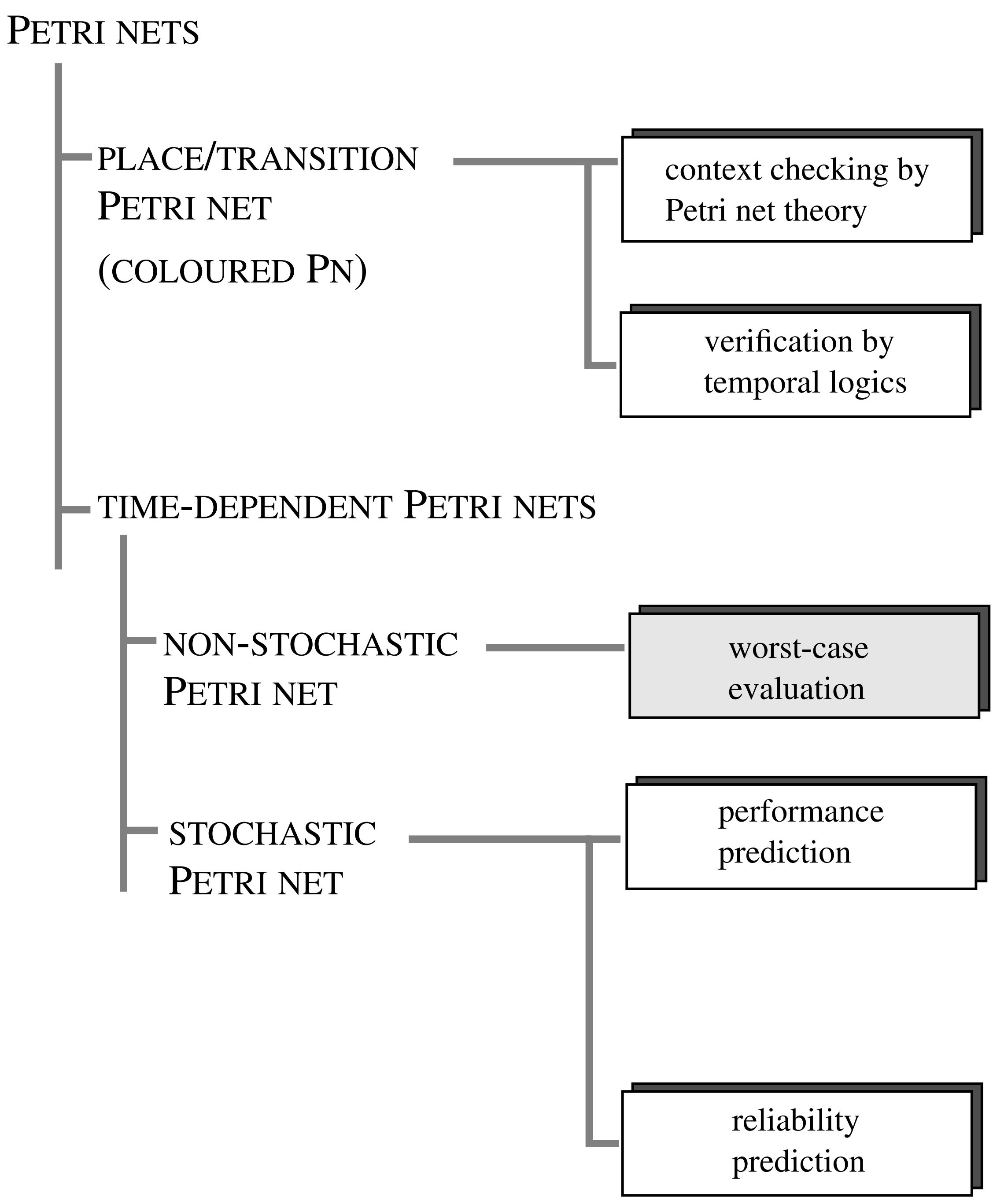
Different validation methods may require net models which vary partly in their level of abstraction. This variety comprises of course typical quantitative parameters as delay or branching information (which are obviously necessary in case of quantitative analysis), but also the granularity of considered control and/or data flow, i.e. the degree of details concerning structural information. Therefore, in order to integrate qualitative as well as quantitative analysis on a common intermediate system representation, an important feature of a related methodology is the ability of a controlled structural reduction, combined with compression of any quantitative parameters.

In [Heiner 95], a method is demonstrated how to develop at first qualitative models as place/transition nets suitable for analysis of general and special qualitative properties. Afterwards, the validated qualitative models are transformed step-by-step by quantitative expansion and property-preserving structural compression into quantitative models as Markovian object nets [Wikarski 95] for performance prediction.

Based on this experience, the approach to integrate different methods on a common representation is extended by a formal method to derive Petri net models suitable for a structure-oriented worst-case prediction of the system’s timing behaviour. For that purpose, we are going to introduce a dedicated kind of time-dependent Petri nets.

The choice of a net type, at best suitable for a given validation task, should be guided by the well-known engineer's basic principle to keep everything as simple as possible. So the answer to the question, which net type should be chosen, depends on the properties to be validated (compare Figure 1). As long as there are hard deadlines to

Figure 1: Relation of model classes & validation tasks.



meet definitely, as it should be the case for systems with predictably timing behaviour, the exact evaluation by non-stochastic nets is unavoidable. Only when average values or probability distributions of performance measures like load, throughput, utilization etc. are wanted, then the application of stochastic Petri nets becomes useful.

The presented approach to worst-case analysis by time-dependent nets is intended to be part of a general framework of Petri net based dependability engineering, starting with qualitative analysis using place/transition nets (or coloured nets as their short-hand notation), and ending up finally in reliability prediction using stochastic nets [Heiner 95].

2 A Petri Net Based Method for Worst-case Timing Prediction

We are now going to extend the approach to integrate different methods on a common representation by proposing a new kind of time-dependent Petri nets especially dedicated to worst-case prediction of a system's timing behaviour. According our analysis objective, we will restrict ourself in the following to discuss only non-stochastic Petri nets.

A search through the literature reveals a lot of different time-dependent Petri net classes, which differ essentially in the provided time concepts. For a concise summary see [Starke 95]. Due to our modelling procedure of control systems, which maps any atomic sequential parts to transitions, time consumption should be connected with transition firing. Among those, the most important time-dependent net classes are the following:

- **duration nets** (usually called timed nets) [Ramchandani 74]:
  constant delays with non-preemptive firing principle,
  firing consumes time,
  earliest firing rule realized by maximal step strategy;

- **interval nets** (usually called time nets) [Merlin 74]:
  interval delays with preemptive firing principle,
  firing itself happens timeless,
  latest firing rule realized by single step strategy.

The firing of a transition on model level corresponds to the execution of the actual atomic sequential system part mapped to it. Generally, such an execution cannot be interrupted again after being initiated (in case of modelling systems without timers, interrupts etc.). So, the non-preemptive firing principle is the natural way to express the system's execution progress. On the other side, execution times cannot generally be characterized adequately by constant delays because of data dependencies, operating system's influences or measurement deviations. Moreover, an adequate model for worst-case prediction of the timing behaviour should be able to determine exactly the minimal and maximal time consumption of critical system parts. So what we really need for our purposes is a suitable combination of the properties listed above - *interval delays*, as used in interval nets, and *non-preemptive firing rule*, as used in duration nets. That leads us to introduce a corresponding new time-dependent Petri nets - the Duration Interval Petri nets. For a formal definition of the corresponding net type see section 3.2.

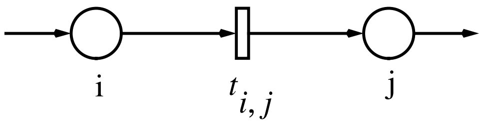
In the following, the quantitative parameters are time intervals of the minimal and maximal duration to execute a given (sequential) part without resource limitations (no processor sharing, no communication delays, etc.). Of course, the interval may collapse to a constant duration in case of purely linear, non-interrupted control statement sequences. These execution time intervals can be measured by monitoring tools of the development environment, or - in special cases - calculated from machine instruction sequences. In the worst case, they have just to be assessed.

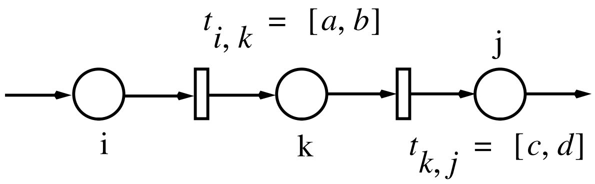
The model, which we need for the evaluation to be done, should not be built from the scratch, but instead of this, the timing model should be derived to a high degree automatically from that net models which we do have due to our qualitative analysis efforts done. Because we already know, how to model control systems by (place/transition) Petri nets [Heiner 95], [Heiner 96b], we have then altogether a formal method to derive systematically Petri net models suitable for worst-case prediction of the concurrent control system under development. For more details of the total framework, comprising different methods and tools, see [Heiner 96a].

The method proposed for worst-case analysis consists basically of two components:

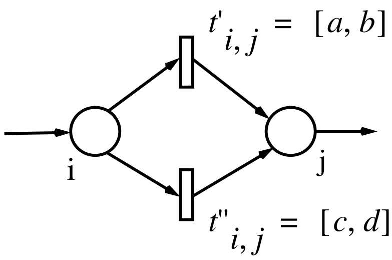
- a set of reduction rules describing the allowed structural reductions and the corresponding transformation rules of the quantitative parameters within any well-structured (sequential) substructures (see Figure 2), and

Figure 2: Structural compression within well-structured (sequential) parts.

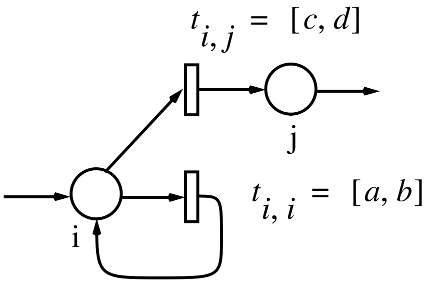




$t_{i,j} = [a + c, b + d]$

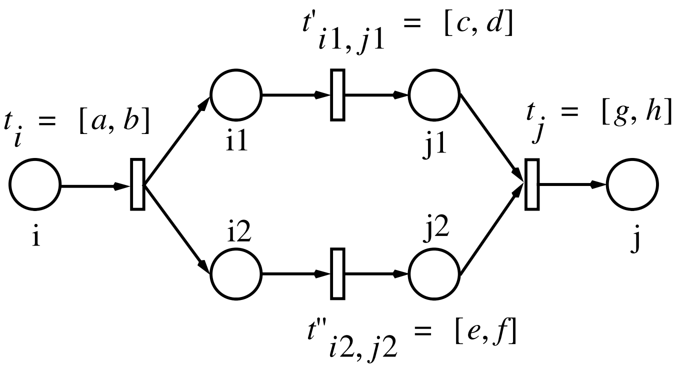


$t_{i,j} = [min(a, c), max(b, d)]$



$t_{i,j} = [ma + c, nb + d]$

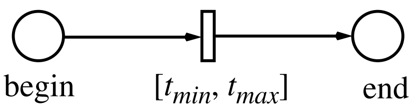
$\left.\begin{array}{ll} m & \text{- lower bound} \\ n & \text{- upper bound} \end{array}\right\}$ *of iterations*



$t_{i,j} = [\ a + max(c, e) + g,$
$\quad b + max(d, f) + h\ ]$

- a method to compute the execution interval of any (non-sequential) system part, including an arbitrary net structure, by length determination of the shortest and largest paths between the corresponding (begin and end) states of the reachability graph (see section 3.3).

If the structural reduction abstracts of cycles, we need the lower and upper bounds of cycle iterations in order to be able to calculate the new time interval. In case of control software worst-case analysis, we can extract the necessary iteration bounds from the corresponding loop statements - provided they are static ones. But this is a well-known unavoidable restriction for systems with high dependability demands which require predictably timing behaviour (see e.g. [Kopetz 95], [Vrchoticky 94]). The total execution time interval of given software can be immediately picked up from the reduced Petri net in case of well-structured (sequential) programs which have been completely reduced to[1)]



Generally, in case of concurrent control systems, the time interval has to be evaluated by a suitable Petri net evaluation tool (we use INA, lately updated according the method proposed here). But a structural reduction combined with compression of quantitative parameters, done before as strong as possible, may reduce the computational costs essentially.

3 Mathematical Background

3.1 Basic Notations and Definitions

We will use the following notations. $\boldsymbol{N}$ denotes the set of natural numbers, $\boldsymbol{N}^+$ is $\boldsymbol{N} \setminus \{0\}$. $\boldsymbol{Q_0}^+$ is the set of nonnegative rational numbers.

Definition 1:

The structure $PN = (P, T, F, m_0)$ is called **Petri net**, iff:

1) In case of “loop forever”-programs, begin and end places coincide.

(1) P, T are finite sets, and F is a mapping
F: $(P \times T) \cup (T \times P) \rightarrow \boldsymbol{N}$, indicating arc weights.
We define $X := P \cup T$, and we assume
$P \cap T = \varnothing$, $P \cup T \neq \varnothing$, and
$\forall x \in X : \exists y \in X : F(x, y) \neq 0 \vee F(y, x) \neq 0$

(2) m_0: $P \rightarrow \boldsymbol{N}$ (initial marking)

A marking of a *PN* is a function m: $P \rightarrow \boldsymbol{N}$, where $m(p)$ denotes the number of tokens in place p. The pre- and postsets of a transition t resp. of a place p are given by
$Ft := \{p | p \in P \wedge F(p, t) \neq 0\}$ and $tF := \{p | p \in P \wedge F(t, p) \neq 0\}$ resp.
$Fp := \{t | t \in T \wedge F(t, p) \neq 0\}$ and $pF := \{t | t \in T \wedge F(p, t) \neq 0\}$.
Each transition $t \in T$ induces the marking t^- and t^+, defined as $t^-(p) := F(p, t)$ and $t^+(p) := F(t, p)$. By Δt we denote $t^+ - t^-$. A transition $t \in T$ is enabled (may fire) at a marking m iff $t^- \leq m$ (i.e. $t^-(p) \leq m(p)$ for each place $p \in P$). When an enabled transition t at a marking m fires, a new marking m' given by $m'(p) := m(p) + \Delta t(p)$ is reached.

3.2 Duration Interval Petri Nets

In order to design and analyze such systems as given above, we define a new kind of time dependent Petri nets where the firing of each transition costs time. Generally, this time cannot be given as a fixed number but it ranges arbitrarily between a minimal and a maximal value. For that reason we will call these nets Duration Interval Petri nets (DIPN). The DIPN are classical Petri nets where to each transition t two nonnegative rational numbers a_t and b_t ($a_t \leq b_t$) are associated; a_t is the minimal possible value of the firing duration of t, and b_t is the maximal possible value. The times a_t and b_t are relatively to the moment at which t was enabled last. When the transition t becomes enabled it starts firing immediately, provided any dynamic conflict[1)] is resolved to its favour. The conflict resolution policy is as usual as in classical Petri nets, i.e. there is a free choice among the enabled transitions.

The firing rule can be basically considered as a 3-phase firing:

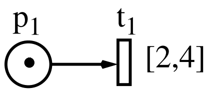
1) Two transitions are in a **dynamic conflict** at the marking m, if both transitions are enabled at m, but the firing of one transition disables the other one (comp. [Starke 90]).

- The tokens are removed from the input places when a transition starts working (firing).
- The transition holds the token(s) while working (time elapse).
- The tokens are put into the output places when a transition finishes working.

The token transfer itself does not consume any time.

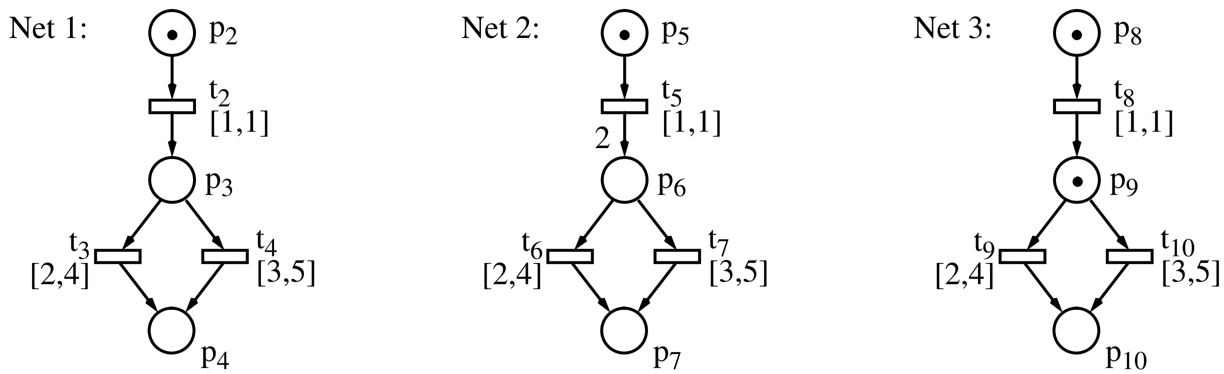
To illustrate the firing behaviour in more detail, let's consider two examples.

Example 1:



Assuming that t_1 becomes enabled at time c, then t_1 starts to fire at this moment and it finishes at the earliest at $c + 2$ and at the latest at $c + 4$, or it finishes at an arbitrary (not necessarily integer) moment between $c + 2$ and $c + 4$.

Example 2:



In Net 1, only one of the transitions t_3 and t_4 can and will fire (after firing of t_2). In Net 2, after the firing of the transition t_5, the place p_6 gets two tokens. Both transitions t_6 and t_7 fire immediately (t_6 and t_7 are in static conflict[1)], but not in a dynamic one). In Net 3, transition t_9 and t_{10} are in a dynamic conflict, but the situation is another one as in the Net 2. Assume that the conflict between t_9 and t_{10} is solved in favour of transition t_9, then after one time unit, the token, which is situated in place p_8 will move to place p_9. Since transition t_9 is still firing (but has not finished, yet), transition t_{10} starts to fire (one time unit later then t_9).

---

1) Two transitions t_1, t_2 are in a **static conflict**, if they share preplaces, i.e. $Ft_1 \cap Ft_2 \neq \varnothing$ (comp. [Starke 90]).

As demonstrated with Net 2 and Net 3, multiple firing of a transition (concurrently to itself) is not allowed, as usual in time-less Petri nets. Moreover, in the context of software-based system modelling this assumption reflects the fact that at any time only one process can run over a certain piece of hardware or software, resp.

Definition 2:

The structure $D = (P, T, F, m_0, DI)$ is called **Duration Interval Petri Net** (DIPN) iff:

(1) (P, T, F, m_0) is a Petri net (skeleton of D),

(2) $DI: T \to \boldsymbol{Q_0}^+ \times \boldsymbol{Q_0}^+$ and for each $t \in T$ holds $a_t \leq b_t$, where $DI(t) = (a_t, b_t)$.

The skeleton of D, the classical Petri net, we denote by $S(D)$. DI is the time function of D. a_t is called the shortest duration time of t ($sdt(t)$), and b_t is called the largest duration time of t ($ldt(t)$), resp.

Please note, immediate transitions are included in our time model, although in reality nothing happens timeless. Immediate transitions help to keep interval boundaries small. Often, system activities may be classified into activities with significant time consumption and those with non-significant (much less) time consumption. Without immediate transitions, this difference had to be modelled by an appropriate (possibly quite large) absolute difference of duration times. With immediate transitions, all duration times may be scaled down relatively to a suitable time axis.

The price for that advantage is the danger of time deadlocks, i.e. of system situations without any time progress because no state is reachable where the system clock is able to advance. Obviously, the appearance of time deadlocks corresponds to inconsistent time constraints. Therefore, their (at best static) analysis is unavoidable [Starke 96].

Obviously, it is easier to study DIPN whose *sdt*'s and *ldt*'s are natural numbers. The net behaviour of an arbitrary DIPN can be traced back, without restriction of generality, to a similar DIPN with natural numbers as lower and upper bounds for the time durations. Thus, in the following we always consider DIPN, whose time function DI is defined in $\boldsymbol{N} \times \boldsymbol{N}$, i.e. the *sdt* and the *ldt* of each transition are natural numbers.

An arbitrary situation in a given “classical” Petri net is completely described, if the number of tokens in each place in the net, i.e. the marking is known. This knowledge only is not enough for a DIPN. For a given marking we can determine which transitions are enabled and which transitions are disabled. But, for an enabled transition we cannot get any information about how many time is elapsed since the transition became enabled last. Therefore we need a carrier for the time information of each transition. Thus we define a time vector with as many components as transitions in the DIPN, we consider. The components of the time vector are (positive) rational numbers or a special sign - the sharp #. The sharp # means that the corresponding transition is not firing. The value zero shows that the corresponding transition is just about to finish its firing. For a firing transition, the corresponding component of the time vector in a given situation is a rational number, which shows how many time has still to elapse until this transition will stop its firing.

For example, we consider Net 3 again: The time vector tv := (#, 1.7, 0.5) shows that the transition t_8 is not firing, transition t_9 and t_{10} are firing - t_9 needs still 1.7 time units and t_{10} needs still 0.5 time units, resp.

The pair (marking, time vector) gives enough information about any situation in a given DIPN at each moment. Thus, this pair, here called state, is now the basic notion in our time-dependent Petri net.

Definition 3:

Let $D = (P, T, F, m_0, DI)$ be a DIPN, $tv\colon T \rightarrow \boldsymbol{Q}_0^+ \cup \{\#\}$, and m be a marking in $S(D)$. Then the pair $z := (m, tv)$ is called a **state** in D.

(Of course, not any state (m, tv) in D is an actually reachable one.)

The state $z_0 := (m_0, tv_0)$, with $tv_0(t) := \#$ for all transitions, is called the initial state of the DIPN D. For example, the initial state in Net 1 is

$$z_0 = (\underbrace{(1, 0, 0)}_{\text{initial marking}}, \underbrace{(\#, \#, \#)}_{\text{initial time vector}}) .$$

The situation in a DIPN changes, when a transition fires or by time elapse. As already mentioned, we demand that each enabled transition starts firing immediately, possibly by solving dynamic conflicts. This means, our firing strategy is the "maximal step", i.e. a maximal set of concurrent enabled transitions is firing or/and starts firing. A maximal step may be empty if the state is transient. In this case, time is elapsed only.

Definition 4:

A set U of transitions is said to be a **maximal step** in the state (m, tv), iff

(1) $U \subseteq U(z) := \{t | t^- \leq m \wedge tv(t) = \#\}$,

(2) when $U = \varnothing$, then $tv(t) \neq \#$ for at least one $t \in T$,

(3) $\left(U^- := \sum_{t \in U} t^-\right) \leq m$ and

(4) there does not exist a set U', $U \subset U'$ and U' satisfies (1), (2) and (3).

(Condition (2) means that the empty set is a maximal step, if there are transitions firing at that moment.)

In order to lay down formally the behaviour of DIPN's we have to answer the question: what is (are) the possible successor(s) of a given state in a given DIPN, when a transition or maximal step of transitions is firing or/and when a certain time is elapsed? Because the firing duration of each transition is not a fixed number, but ranges between a given interval, the successor of the given state will vary, i.e. we get more than one successor - in general a huge amount of successor states.

Definition 5: (Firing)

Let $z = (m, tv)$ be a state , i.e.

$m: \; P \rightarrow \boldsymbol{N}$

$tv: \; T \rightarrow \boldsymbol{Q_0}^+ \cup \{\#\}$, where

$tv(t) = \#$ **iff** t is not firing, and

$tv(t) = \tau$ **iff** t is firing, and t finishes the firing after τ time units.

Then $(m, tv) \xrightarrow{U} (m', tv')$ **iff**

(1) U is a maximal step in z.

(2) $m' := m - U^-$

(3) if $t \notin U$ then $tv'(t) := tv(t)$

if $t \in U$ then $sdt(t) \leq tv'(t) \leq ldt(t)$.

And $(m, tv) \xrightarrow{\tau} (m', tv')$ **iff**

(1) $\tau \leq \tau(z) := min\{tv(t) | tv(t) \neq \#\}$

(2) if $\tau < \tau(z)$ then $m' := m$ and

$$tv'(t) := \begin{cases} tv(t) - \tau & , t \text{ is firing} \\ \# & , \text{else} \end{cases} ,$$

if $\tau = \tau(z)$ then $m' := m + \sum_{tv(t) = \tau(z)} t^+$ and

$$tv'(t) := \begin{cases} \# & , tv(t) = \# \vee tv(t) = \tau(z) \\ tv(t) - \tau & , \text{else} \end{cases} .$$

How can we interpret this definition? We assign to each transition in the net an “egg timer”. The timer does not work ($vt(t) = \#$)at the marking *m* if *t* is disabled at *m*. If *t* has just become enabled at *m*, it starts firing immediately, i.e. the tokens are removed from all preplaces, and its timer is started to count down. At the beginning, the timer shows a possible duration (within the given time interval) of *t*. At each moment, while the timer is running, it shows how many time has still to elapse until the transition *t* will finish firing. If the timer reaches zero, *t* stops firing, and the tokens are put to all postplaces.

Using this rule of state changes, the state space could be generated for a given DIPN starting with the state $z_0 := (m_0, tv_0)$. But before implementing a new tool, we want to gather experience of the usefulness of the newly introduced net type. So (as short-term solution), we transform the DIPN into another well studied net type(s) with analysis tool(s) already available.

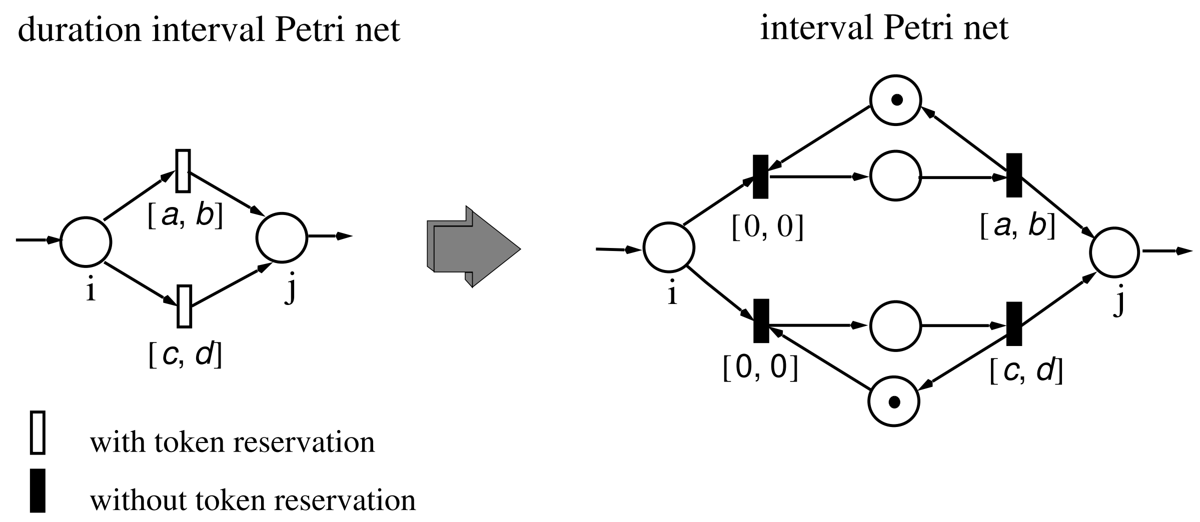
3.3 Transformation into Interval Petri Nets

In order to analyze a DIPN, we translate it into an interval Petri net[1)]. We do this because:

- the interval nets are already well studied (comp. [Popova 91]),
- there exists a tool for qualitative and quantitative analysis of interval nets, INA [Starke 92],
- the transformation is easy to manage.

The transformation adds two places and one immediate transition for each transition (involved in a structural conflict) of the original net, according the pattern given in Figure 3. Therefore, DIPN may be considered as an abbreviation of that structural subset of interval nets in which only immediate transitions are allowed to be in conflict, if any. But, this transformation blowing up the net structure does not enlarge the qualitative state space. Thus, we can compute a reachability graph for the given DIPN following the method to construct a reachability graph for interval nets presented in [Popova 91]. The nodes of this reachability graph are the „essential“, so-

Figure 3: Transformation pattern from DIPN to interval nets.



1) Similar considerations may be made for duration Petri nets.

called integer states in the net, where all components of the corresponding time vector are integers or #. Obviously, this reachability graph includes only a discrete part of all the situations, which are possible in a given DIPN. But the knowledge of the net behaviour in the integer states is sufficient for knowing the whole behaviour of the net, and the computation of the integer states is much easier to manage. For more details see [Popova 91].

Using this discrete reachability graph, we can analyze the DIPN in quality (boundedness, deadlock freedom, liveness, etc.) as well in quantity (including freedom of time deadlocks). Moreover, it has been shown that the time shortest and the time largest path between two reachable states resp. markings can be determined using the discrete reachability graph only. In order to be able to do it, it has been proven that the solution of the corresponding Linear Program, which is of the following type

$$\textit{min} / \textit{max}\ t_1 + \ldots + t_n$$

$$\text{(LP)} \quad \left\{ \begin{array}{l} b_1 \le a_{11}t_1 + \ldots + a_{1n}t_n \le c_1 \\ \ldots \\ b_m \le a_{m1}t_1 + \ldots + a_{mn}t_n \le c_m \\ a_{ij} \in \{0, 1\},\ b_i \in \boldsymbol{N},\ c_i \in \boldsymbol{N} \\ \forall i \forall s \forall k\, (1 \le i \le n \wedge 1 \le s \le k \le m \wedge \\ a_{is} = a_{ik} = 1 \rightarrow \forall j\, (s \le j \le k \rightarrow a_{ij} = 1)) \end{array} \right.$$

is always an integer. Please note, **this Linear Program has never to be solved** for a given net. It is only used within the proof to show the non-existence of non-integer solutions. Therefore, minimal and maximal durations of certain paths can be computed using the discrete reachability graph only. (In case of computing the time largest path we disregard possible cycles. But for our applications this is not a restriction, see section 2).

In opposite to that, the interval net analysis method introduced in [Berthomieu 91], [Aalst 92] is directly based on the solution of inequality systems. Obviously, their approach is quite laboriously in comparison with our discrete way of construction.

4 Final Remarks

A new type of time-dependent Petri net has been introduced to model and analyze in a straightforward manner concurrent control systems, which require predictably timing behaviour. The new time model allows more compact system descriptions as it would be the case using interval nets, and more precise one as under the restriction to duration nets. In Duration Interval Petri nets, transition firing consumes time, while the firing times are given by lower and upper bounds. This net type allows to compute execution durations (of the whole system or of interesting system parts) in the best and the worst case.

Up to now, this computation is done by transforming the net models of the newly introduced net type into interval Petri nets, which are well-investigated and computer-aided analyzable by already existing tools. This transformation adds for each transition two places and one immediate transition to the original user net, resulting possibly in states, which are transient from the user's point of view. Therefore, in order to increase user friendliness of the analysis tools provided it would be helpful to analyze Duration Interval Petri nets in a direct way, without blowing up the net structure.

These results have been applied to a medium-sized reactive system, a production cell of a metal processing plant [Heiner 96b]. Modelling the controlling system and the controlled process environment by Duration Interval Petri nets and assuming given time intervals for the elementary mechanical motion steps, we are now able to compute the worst case of the controlling system's execution time with the (latest update of the) software package INA [Starke 92].

On-going investigations to improve the presented approach to worst-case system analysis deal with the following two problems. (1) In order to take advantage from the results of qualitative analysis which should always preceed any kind of quantitative analysis, it is necessary to investigate carefully the influence of time on qualitative properties, e.g. to characterize time-independently live net structures. (2) Any kind of reachability graph based analysis is limited in practice due to the well-known state explosion problem. At least for 1-bounded DIPN the definition of an appropriate partial order behaviour description should be possible.

Acknowledgment

We would like to thank Peter H. Starke for his immediate compliance with our wishes by extending INA with algorithms for largest paths search and by provision of an algorithm to transform duration interval Petri nets to interval Petri nets. Without his support, the practical scrutiny of our approach would not have been possible in such a short time period.

References

[Aalst 92] Aalst, W.M.P van der: *Timed Coloured Petri Nets and their Application to Logistics;* Diss. Tech. Univ. Eindhoven, 1992.

[Balbo 92] Balbo, G.: *Performance Issues in Parallel Programming;* LNCS 616, 1992, pp. 1-23.

[Berthomieu 91] Berthomieu, B.; Diaz, M.: *Modeling and Verification of Time Dependent Systems Using Time Petri Nets;* IEEE Trans. on Software Egineering 17(91)3, 259-273.

[Donatelli 94] Donatelli, S. et al.: *Use of GSPNs for Concurrent Software Validation in EPOCA;* Information and Software Technology 36(94)7, 443-448.

[Heiner 94] Heiner, M.; Ventre, G.; Wikarski, D.: *A Petri Net Based Methodology to Integrate Qualitative and Quantitative Analysis;* Information and Software Technology 36(94)7, 435-441.

[Heiner 95] Heiner, M.: *Petri Net Based Software Dependability Engineering;* Tutorial Notes, Int. Symposium on Software Reliability Engineering, Toulouse, Oct. 1995, 101 p., available also via http://www.informatik.tu-cottbus.de.

[Heiner 96a] Heiner, M.; Popova-Zeugmann, L.: *Worst-case Analysis of Concurrent Systems with Duration Interval Petri Nets;* BTU Cottbus, Techn. Report I-02/1996, February 1996, available also via http://www.informatik.tu-cottbus.de.

[Heiner 96b] Heiner, M.; Deussen, P.: *Petri Net Based Design and Analysis of Reactive Systems;* Proc. Workshop on Discrete Event Systems (WODES ’96), Edingburgh, August 1996, pp. 308-313.

[Kopetz 95] Kopetz, H.: *The Time-Triggered Approach to Real-Time System Design;* in Randell, B.; Laprie, J.-C.; Kopetz, H.; Littlewood, B. (eds.): Predictably Dependable Computing Systems, Springer 1995, pp. 53-66.

[Merlin 74] Merlin, P.: *A Study of the Recoverability of Communication Protocols;* Univ. of California, Computer Science Dep., PhD Thesis, Irvine, 19974.

[Popova 91] Popova, L.: *On Time Petri Nets;* J. Information Processing and Cybernetics EIK 27(91)4, 227-244.

[Ramchandani 74] Ramchandani, C.: *Analysis of Asynchronous Concurrrent Systems Using Petri Nets;* PhD Thesis, MIT, TR 120, Cambridge (Mass.), 1974.

[Starke 90] Starke, P. H.: *Analysis of Petri Net Models (in German);* B. G. Teubner Stuttgart 1990.

[Starke 92] Starke, P. H.: *INA - Integrated Net Analyzer;* Manual; Berlin 1992.

[Starke 95] Starke, P. H.: *A Memo On Time Constraints in Petri Nets;* Humboldt-University zu Berlin, Informatik-Bericht Nr. 46, August 1995.

[Starke 96] Starke, P. H.: *On State-Invariants of Timed Petri Nets;* Humboldt-University zu Berlin, Informatik-Bericht Nr. 59, April 1996.

[Vrchoticky 94] Vrchoticky, A.: *The Basis for Static Execution Time Prediction;* Technical University of Vienna, PhD Thesis, 1994.

[Wikarski 95] Wikarski, D.; Heiner, M.: *On the Application of Markovian Object Nets to Integrated Qualitative and Quantitative Software Analysis;* Fraunhofer ISST, Berlin, ISST-Berichte 29/95, Oct. 1995.