Polynomial Throughput Bounds for Equal Conflict Petri Nets with Multi-Guarded Transitions

Jorge Júlvez

Department of Software, Universitat Politècnica de Catalunya
Jordi Girona 1-3, 08034 Barcelona, Spain
julvez@lsi.upc.edu

Abstract

Early evaluation is a strategy that aims at enhancing the system performance by executing operations as soon as enough information is available. A common unit that allows early evaluation is the multiplexer: its output can be produced as soon as data is available in the selected channel, without waiting for data in the other channels. Petri nets can model early evaluation of operations by associating several guards with each transition. A multi-guarded transition can fire as soon as the guard selected for the next firing is satisfied. This paper proposes a linear programming problem to compute throughput bounds for equal conflict Petri nets with multi-guarded transitions.

1 Introduction

Some common operations of computer systems can be evaluated even if some input data are not available. Early evaluation takes advantage of this fact by executing operations as soon as enough information is available. The early evaluation of a given operation can trigger the evaluation of other operations what might result in an overall enhancement of the system performance.

Consider the multiplier device in Figure 1(a) that reads two numbers from input channels a and b, and produces the result in channel c. A token in a given channel means availability of data in the channel, e.g., a token in a means that a number in channel a is available. A multiplier without early evaluation always waits for both input numbers to be available, then it would consume both numbers and produce the result in c. Assume that no number is available in b (it has not been produced yet), but a number equal to 0 is available in a. Then, a multiplier with early evaluation can produce the result 0 without waiting for b.

The multiplexer is one of the most common units allowing early evaluation. The multiplexer in Figure 1(b) has two input channels x and y, one control channel s, and one output channel z with the following behavior:

$$\texttt{if } s \texttt{ then } z = x \texttt{ else } z = y \qquad (1)$$

A multiplexer without early evaluation waits for x, y, and s to be available, consume their values, and yields in z the result of evaluating (1). A multiplexer with early evaluation can yield the result without waiting for y if s is true (without waiting for x if s is false).

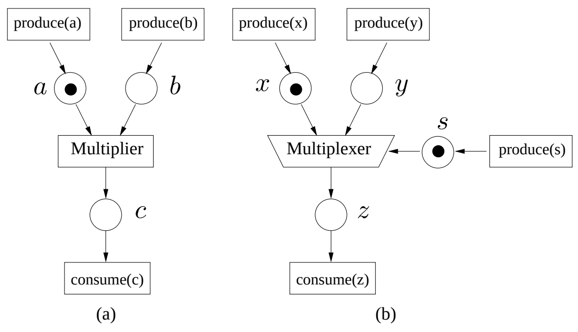

Figure 1. Both a multiplier (a) and a multiplexer (b) can operate even if some inputs are not available.

Apart from some circuit devices (multiplier, multiplexer, AND-gate, OR-gate, ...), early evaluation can be used in a number of software and communication systems.

Example 1 *Consider the parallel program in Figure 2(top) which is executed by two processors with shared memory. Variables i and j contain the current iteration number which is reset when it reaches cap. Subindices are used to access the value of the variables produced at a given iteration, e.g., b_k is the value yielded by function fun_b in the k^{th} iteration (it is assumed that cap is high enough to ensure that every variable x_k is used before its value is overwritten by iteration $k + cap$).*

The Petri net in Figure 2(bottom) is a graphical representation of the parallel program. Transitions t_a, t_b, t_c, t_d, t_f are associated with tasks $read(a_i)$, $fun\_b(a_i)$, $fun\_c(b_i, e_i)$, $read(d_j)$, $fun\_f(e_j)$, and t_e is associated with the conditional statement in processor 2. Places p_{ae} and p_{ec} act as communication channels (or queues) between the processors: when t_a (t_e) fires, a token joins queue p_{ae} (p_{ec}) and waits to be served by t_e (t_c) following a first-come-first-served discipline.

Let us assume that $bool\_cond$ is a boolean expression that depends on the environment and not on the particular values of the program variables. Then, if a_j is available and $bool\_cond$ is known to be true*, the conditional statement can be evaluated without waiting for a token in d_j to be available. Similarly, if $bool\_cond$ is* false*, the conditional statement can be evaluated without waiting for a_j.*

This work was supported by the Spanish Ministry of Education and Science (Juan de la Cierva fellowship) and project TIN2007-66523.

```
Processor 1
$i = 0$
loop
 $read(a_i)$
 $b_i = fun\_b(a_i)$
 $c_i = fun\_c(b_i, e_i)$
 $i = (i+1)\ mod\ cap$
end loop
```

```
Processor 2
$j = 0$
loop
 $read(d_j)$
 if $bool\_cond$
   then $e_j = fun\_e(a_j)$
   else $e_j = fun\_e(d_j)$
 end if
 $f_j = fun\_f(e_j)$
 $j = (j+1)\ mod\ cap$
end loop
```

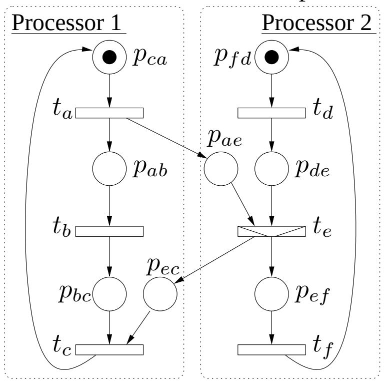

Figure 2. A parallel program and its graphical representation as a Petri net.

Early evaluation has already been used in asynchronous design [1]. The primary reason to use early evaluation of operations is to enhance the system performance. Petri nets [7] represent a well-known formalism for the modeling and analysis of a wide variety of discrete event systems. Conventional Petri nets rely on the AND-causality paradigm (often associated with *rendez-vous* synchronizations). According to AND-causality, every input of a given operation must be available to perform the operation.

In contrast to conventional Petri nets, multi-guarded Petri nets [4] associate a set of guards with each transition. If the guard selected for the next firing of a transition is satisfied, it can fire even if some input places are not marked. This way, multi-guarded transitions can be used to model early evaluation of a number of operations. For instance, early evaluation of the conditional statement in Example 1 can be modeled by associating two guards with t_e, $g_1 = \{p_{ae}\}$ and $g_2 = \{p_{de}\}$: if $bool\_cond$ is *true* the guard for t_e is g_1 and it indicates that t_e can fire as soon as a token in p_{ae} is available; if $bool\_cond$ is *false* the guard is g_2 and then t_e just requires a token in p_{de} to fire.

In order to define the time evolution of a multi-guarded Petri net system, a delay is associated with each transition, and a real number representing the probability of being selected is associated with each guard, e.g., the probability associated with g_1 is the (estimated) probability that $bool\_cond$ is *true*, and the probability associated with g_2 is the probability that $bool\_cond$ is *false*. Thus, the time evolution of a multi-guarded Petri net system becomes a semi-Markov process. Several techniques exist to compute the steady state behavior of semi-Markov processes. Unfortunately, most of them suffer from the state explosion problem and are non suitable for large systems. An alternative approach is to intensively simulate the system, but this might not be appropriate if quick estimations are required.

The main goal of this paper is to obtain an efficient method to compute steady state throughput bounds of multi-guarded Petri net systems. The method is based on a linear programming problem that relates the throughput of each transition to the average marking of its input places. We will focus on Equal Conflict systems [9] working under infinite server semantics. Preliminary results for marked graphs under single server semantics can be found in [4].

The remainder of the paper is organized as follows: Section 2 introduces multi-guarded Petri nets. The evolution of timed multi-guarded Equal Conflict nets is defined in Section 3. A linear programming problem to compute upper throughput bounds is designed in Section 4. Section 5 shows the bounds obtained for three multi-guarded systems. The main conclusions are drawn in Section 6.

2 Untimed Multi-guarded Petri Nets

2.1 Basic Definitions

It is assumed that the reader is familiar with Petri nets (PNs) (see [7] for instance).

Definition 1 (MPN) *A* Multi-guarded Petri Net *(MPN) is a tuple $\mathcal{N} = \langle P, T, Pre, Post, G\rangle$ where:*

- *P is a set of $|P|$ places, and T is a set of $|T|$ transitions.*
- *$Pre : P \times T \rightarrow \mathbb{N} \cup \{0\}$ and $Post : P \times T \rightarrow \mathbb{N} \cup \{0\}$ are the* pre- *and* post- *incidence functions that specify the arc weights. The incidence matrix of the net is $\mathbb{C} = Post - Pre$. The* preset *and* postset *of a node $x \in P \cup T$ are denoted as ${}^{\bullet}x$ and $x^{\bullet}$.*
- *$G : T \rightarrow 2^{2^P}$ assigns a set of guards $G(t)$ to every transition t. The following two conditions must be fulfilled: a) $\forall\ g \in G(t)$ it holds $g \subseteq {}^{\bullet}t$; b) $\bigcup\limits_{g \in G(t)} g = {}^{\bullet}t$.*

In a MPN, a transition t can also satisfy the condition $G(t) = \{\{{}^{\bullet}t\}\}$. Such transitions are called *simple* transitions; the rest of transitions are called *multi-guarded* transitions. Simple transitions are represented graphically as empty rectangles; multi-guarded transitions are represented as rectangles with oblique lines inside (see t_e in Figure 2).

Definition 2 (MPN system) *A* Multi-guarded Petri Net system *is a tuple $\langle \mathcal{N}, M_0\rangle$ where $\mathcal{N}$ is a MPN, and $M_0 : P \rightarrow \mathbb{N} \cup \{0\}$ assigns an initial marking to each place p. The initial marking of place p is denoted as $M_0(p)$.*

2.2 Firing Rule

Each guard of a transition t is a set of places from which tokens are required for a given firing instance of t. In the untimed framework, the guard for a given firing instance is selected in a non-deterministic way. When the selected guard is satisfied, the transition is enabled and can fire.

Definition 3 (Enabling condition) *Let $g \in G(t)$ be the guard selected for the next firing of t, then t is enabled if $M(p) \geq Pre(p, t)$ for every $p \in g$.*

Thus, a multi-guarded transition t is enabled even if $p \in {}^\bullet t$ exists such that $M(p) < Pre(p,t)$, $p \notin g$. Hence, the enabling condition of multi-guarded transitions can be seen as a relaxation with respect to the enabling condition of simple transitions.

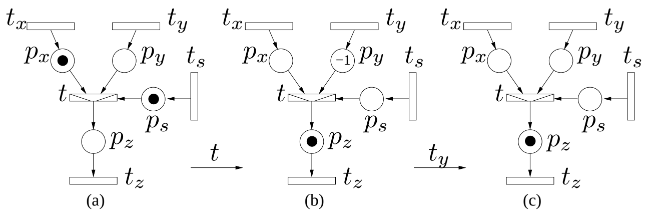

Figure 3. (a) A MPN modeling the multiplexer in Figure 1(b); (b) The firing of t produces a negative token in p_y; (c) The firing of t_y produces a token that is cancelled out by the negative token.

Consider the MPN system in Figure 3(a) modeling the multiplexer in Figure 1(b). If transition t is simple, it would wait for a token in p_x, p_y and p_s to be enabled. Its firing removes a token from every input place (in order to evaluate (1)) and produces a token in p_z. Notice that if some input tokens were not consumed, subsequent firings of t would use data produced for a previous firing instance.

If t is multi-guarded with $G(t) = \{\{p_x, p_s\}, \{p_y, p_s\}\}$, and the guard selected for the next firing is $\{p_x, p_s\}$ then t is enabled in Figure 3(a). In order to preserve the data flow of the original simple transition, its firing should consume a token from every input place and produce a token in p_z. Given that p_y is not marked in Figure 3(a) the firing of t cannot consume a token in p_y. However, it is possible to consume the first token that will come into p_y to obtain an equivalent behavior. A simple way to achieve this is by producing a negative token in p_y, see Figure 3(b). This way, when a token is produced by t_y, it is automatically cancelled out by the existing negative token, see Figure 3(c). Negative tokens or markings are a direct result of removing tokens from every input place even if they are not marked.

Definition 4 (Firing rule) *Let t be enabled at M, the firing of t yields a new marking M' such that $M' = M + \mathbb{C}(P,t)$, where $\mathbb{C}(P,t)$ is the column of $\mathbb{C}$ corresponding to t.*

Thus, although the firing of a multi-guarded transition t can occur as soon as the selected guard is satisfied, it changes the marking in the same way simple transitions do. This way, the state equation $M = M_0 + \mathbb{C} \cdot \sigma$ provides a necessary condition for the reachability of M, where σ is the firing count vector of the transitions and M can contain negative elements. Then, as in usual PNs, vectors $Y \geq 0$, $Y \cdot \mathbb{C} = 0$ ($X \geq 0$, $\mathbb{C} \cdot X = 0$) represent P-semiflows or conservative components (T-semiflows or consistent components). A semiflow V is said to be *minimal* when its support, $\|V\|$, is not a proper superset of the support of any other, and the greatest common divisor of its elements is one. A MPN $\mathcal{N}$ is conservative (consistent) if there exists $Y > 0$ such that $Y \cdot \mathbb{C} = 0$ ($X > 0$ such that $\mathbb{C} \cdot X = 0$).

2.3 Equal Conflict Nets

Transitions t and t' are said to be in Equal Conflict Relation [9] (denoted by $(t,t') \in$ ECR) iff $t = t'$ or $Pre(P,t) = Pre(P,t') \neq 0$. This is an equivalence relation on the set of transitions. Each equivalence class is called Equal Conflict Set (ECS).

Definition 5 (MEC) *A MPN is a* Multi-guarded Equal Conflict net *(MEC) if for every $t, t' \in T$ such that ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$ it holds $(t,t') \in$ ECR.*

From a structural point of view equal conflict nets [9] subsume several known classes of Petri nets: Weighted T-Graphs ($\forall\ p\ |{}^\bullet p| = |p^\bullet| = 1$) which is the weighted version of Marked Graphs, Choice Free nets ($\forall\ p\ |p^\bullet| = 1$), and ordinary Free Choice nets ($\forall\ t, t'$, if ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$ then ${}^\bullet t = {}^\bullet t'$). The rest of the paper focuses on conservative and consistent MEC systems.

3 Timed Multi-guarded Equal Conflict Nets

3.1 Definition and Time Evolution

The concept of time is introduced in MECs by means of a time delay associated with each transition [3]. In order to study the behavior of the timed system, a non-null probability is associated with each guard.

Definition 6 (TMEC) *A* Timed Multi-guarded Equal Conflict net *(TMEC) is a tuple $\mathcal{N}^\tau = \langle P, T, Pre, Post, G, \alpha, \delta, \mathcal{R}\rangle$ where:*

- *$\langle P, T, Pre, Post, G\rangle$ is a MEC.*
- *$\alpha : G \rightarrow (0,1]$ assigns a probability to each guard such that for every transition t: $\sum\limits_{g \in G(t)} \alpha(g) = 1$.*
- *$\delta : T \rightarrow \mathbb{R}^+ \cup \{0\}$ assigns a delay to every transition.*
- *$\mathcal{R}$ is the routing matrix for transitions in ECR.*

A transition t is *timed* if $\delta(t) > 0$; a transition t is *immediate* if $\delta(t) = 0$. Immediate simple transitions are represented graphically as lines (see t_{im1} in Figure 9).

We assume that the transitions to fire are selected at net level, i.e., a preselection execution policy is adopted. The preselection policy is introduced by means of immediate simple transitions. This way, in the timed framework, only immediate simple transitions can be in conflict with other transitions, i.e., if $(t,t') \in$ ECR and $t \neq t'$ then t and t' are simple and $\delta(t) = \delta(t') = 0$.

Let $\{t_1, \ldots, t_k\}$ be an ECS and $r_i \in \mathbb{R}^+$ be the routing rate associated with t_i, i.e., the probability of firing t_i when the ECS is enabled is $r_i / \Sigma_{j=1}^k r_j$. The matrix $\mathcal{R}$ is obtained in the following way: For each ECS $\{t_1, \ldots, t_k\}$, $k-1$ rows are added to $\mathcal{R}$, each row corresponding to pairs $\{t_1, t_2\}$, …, $\{t_{k-1}, t_k\}$. If h is the row number for pair $\{t_i, t_j\}$, then $\mathcal{R}(h, t_i) = r_j$, $\mathcal{R}(h, t_j) = -r_i$, and the rest of elements in row h are zero.

Definition 7 (TMEC system) *A* Timed Multi-guarded Equal Conflict system *is a tuple $\langle \mathcal{N}^\tau, M_0\rangle$ where $\mathcal{N}^\tau$ is a TMEC, and $M_0 : P \rightarrow \mathbb{N} \cup \{0\}$ is the initial marking.*

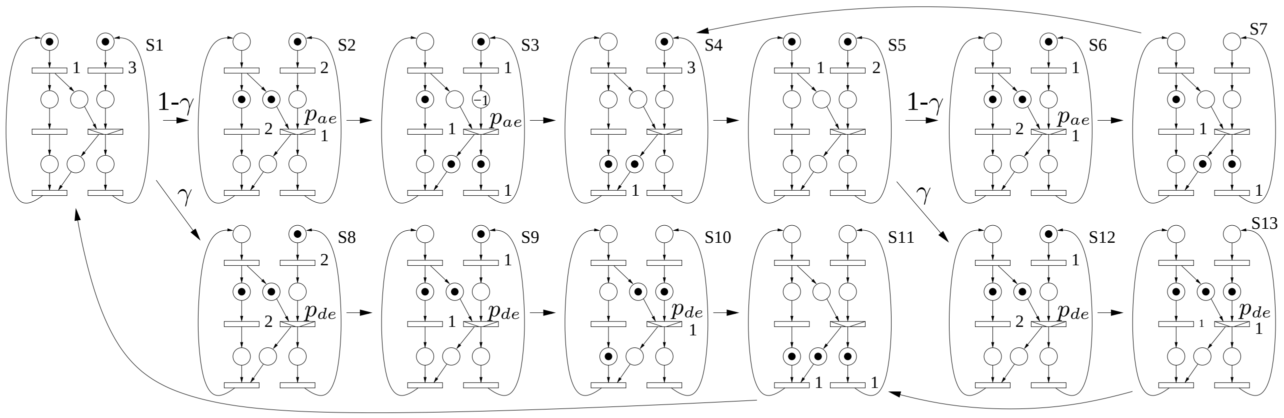

Figure 4. Semi-Markov process of the TMEC system in Figure 2.

For the firing of transitions, infinite server semantics is adopted. According to this semantics, several firing instances of the same transition can proceed in parallel. As in simple PNs, finite server semantics can be simulated by adding extra places: An easy way to make a multi-guarded transition t work under k-server semantics is by adding a self-loop place p, i.e., $^\bullet p = p^\bullet = t$, with initial marking $M_0(p) = k$, that is contained in every guard of t.

It will be assumed that all the delays are deterministic. Thus, the evolution of a TMEC system along time is subject to the following rules:

- For each firing instance of a transition t a guard $g \in G(t)$ is selected. The probability of selecting g is $\alpha(g)$.
- If a timed transition t becomes enabled for a given firing instance at time τ it will fire at $\tau + \delta(t)$. Several firing instances of t can progress simultaneously.
- Immediate transitions fire as soon as they become enabled. If an ECS is enabled, $\mathcal{R}$ determines the probability of firing each transition.

3.2 Semi-Markov Process

The evolution of a TMEC system along time depends on the probability of the guards selected for each firing, and the routing selected for transitions in conflict. Thus, the marking evolution of a TMEC system can be described as a semi-Markov process.

Example 2 *Consider again the MEC system in Figure 2. Assume that the delays associated with the transitions are $\delta(t_a) = 1$, $\delta(t_b) = 2$, $\delta(t_c) = 1$, $\delta(t_d) = 3$, $\delta(t_e) = 1$, $\delta(t_f) = 1$; and that the probability of each guard is $\alpha(\{p_{de}\}) = \gamma$ and $\alpha(\{p_{ae}\}) = 1 - \gamma$. The states of the associated semi-Markov process are depicted in Figure 4. The sojourn time of every state is 1. The time to fire of enabled transitions is shown on the right of the transitions. The guard selected for the firing of t_e is shown on top of t_e (it is not shown if the marking of both input places is zero).*

At state s_1(s_5) transition t_a is enabled, its firing produces a token in p_{ab} and p_{ae}. If the guard selected for t_e is p_{ae}, the new state is s_2(s_6) at which t_e is enabled and fires. If the selected guard is p_{de}, the new state is s_8(s_{12}), and t_e waits for a token in p_{de} to fire.

If the embedded Markov chain of the semi-Markov process is irreducible and recurrent then the fraction of time, $\psi(s)$, that the process spends in s can be computed as [10]:

$$\psi(s) = \frac{\rho(s) \cdot \Pi(s)}{\sum\limits_{s_i \in \mathcal{S}} \rho(s_i) \cdot \Pi(s_i)} \tag{2}$$

where $\mathcal{S}$ is the set of states, $\rho(s_i)$ is the average sojourn time of state s_i, and Π is the stationary measure (or distribution) of the embedded Markov chain [10].

The value of ψ for the semi-Markov process in Figure 4 is: $\psi(s_1, \ldots, s_{13}) = \frac{\gamma}{4+\gamma^2} \cdot (1,\ 1-\gamma,\ 1-\gamma,\ \frac{1-\gamma}{\gamma},\ \frac{1-\gamma}{\gamma},\ \frac{(1-\gamma)^2}{\gamma},\ \frac{(1-\gamma)^2}{\gamma},\ \gamma,\ \gamma,\ \gamma,\ 1,\ 1-\gamma,\ 1-\gamma)$. From ψ, several steady state measures of interest can be obtained. Let us compute, for instance, the steady state throughput of t_a, i.e., the average number of times t_a fires per time unit in the steady state. The steady state throughput of a transition t, $Th(t)$, can be computed as its average enabling degree, $\overline{enab}(t)$, divided by its delay, $\delta(t)$:

$$\delta(t) \cdot Th(t) = \overline{enab}(t) \tag{3}$$

Since t_a is enabled in s_1 and s_5, the throughput of t_a is:

$$Th(t_a) = \frac{\overline{enab}(t_a)}{\delta(t_a)} = \psi(s_1) + \psi(s_5) = \frac{1}{4+\gamma^2} \tag{4}$$

3.3 Singleton Guards

The behavior of a multi-guarded transition t can be mimicked by a multi-guarded transition t' with singleton guards, i.e., $\forall\ g \in G(t')\ \ |g| = 1$, and immediate transitions. This property makes the developments in Section 4 easier. A simple way to obtain t' is to add as many immediate transition as $|G(t)|$, and duplicate each input place p of t as many times as the number of guards that contain p.

The net in Figure 5(b) shows the result of transforming transition t in Figure 5(a) which has two guards $g_1 = \{p_1, p_2\}$ and $g_2 = \{p_2, p_3\}$, into a transition t' with singleton guards. Transitions t_{aux1} and t_{aux_2} are immediate transitions. The probabilities of the singleton guards are: $\alpha(p_{g1}) = \alpha(g_1)$, $\alpha(p_{g2}) = \alpha(g_2)$. In the sequel, it is assumed that multi-guarded transitions have singleton guards.

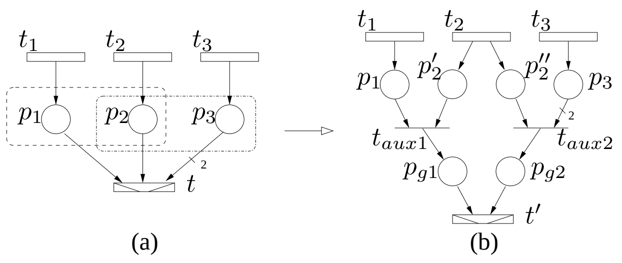

Figure 5. (a) A multi-guarded transition t with two guards; (b) Its transformation to a multi-guarded transition with singleton guards.

4 Polynomial Throughput Bounds

This section presents a linear formulation to compute an upper bound for the steady state throughput of a TMEC.

4.1 Throughput and Visit Ratios

The throughput of a transition t is the average number of times t fires per time unit in the steady state. For a given TMEC system $\langle \mathcal{N}^\tau, M_0\rangle$, the throughput of all transitions, $Th(\mathcal{N}^\tau, M_0)$, is given by the following (Cesàro) limit: $Th(\mathcal{N}^\tau, M_0) = \lim\limits_{\tau\to\infty} \frac{\sigma(\tau)}{\tau}$ where τ represents time and $\sigma(\tau)$ is the firing count vector at time τ. The average marking, $\overline{M}(\mathcal{N}^\tau, M_0)$, of a TMEC system $\langle \mathcal{N}^\tau, M_0\rangle$ is: $\overline{M}(\mathcal{N}^\tau, M_0) = \lim\limits_{\tau\to\infty} \frac{1}{\tau}\int_0^\tau M(\xi)d\xi$ where $M(\xi)$ is the marking at time ξ.

Both the throughput and the average marking can be obtained from the semi-Markov process associated with the TMEC system. For clarity, given a TMEC system, the steady state throughput of transition t is simply denoted as $Th(t)$ and the average marking of place p as $\overline{M}(p)$.

The relative throughputs of transitions are often called *visit ratios*. The vector of visit ratios normalized, for instance, for transition t_1 is defined as $v^{(1)}(j) = Th(t_j)/Th(t_1)$. For a live and bounded simple Equal Conflict net, $v^{(1)}$ is the only solution of [3]:

$$\left(\begin{smallmatrix}\mathbb{C}\\ \mathcal{R}\end{smallmatrix}\right)\cdot v^{(1)} = 0;\;\; v^{(1)}(1) = 1 \qquad (5)$$

The vector $v^{(1)}$ satisfying (5) is also the vector of visit ratios of any TMEC system with incidence matrix $\mathbb{C}$ and routing matrix $\mathcal{R}$.

Proposition 1 *Let $\langle \mathcal{N}^\tau, M_0\rangle$ be a TMEC system. The only solution of* (5)*, $v^{(1)}$, is the vector of visit ratios.*

Proof: Assume that the vector of visit ratios, $v^{(1)}$, is different from the only solution of (5). Necessarily $\mathcal{R}\cdot v^{(1)} = 0$, then $v^{(1)}$ is not a T-semiflow. Since the system is conservative, the marking of at least one place p will tend to minus infinity. This implies that the guards that contain p are never selected for the firing of its output transition t. Contradiction, by definition $\bigcup\limits_{g\in G(t)} g = {}^\bullet t$ and $\alpha(g) > 0$ for every guard $g \in G(t)$, i.e., p is contained at least in one guard with strictly positive probability. □

4.2 Linear Programming Problem

Let us partition the set of transitions in two sets T_m and T_s: $t \in T_m$ iff t is a multi-guarded transition; $t \in T_s$ iff t is a simple transition. We will design a linear programming problem that includes an inequality for every input place of a simple transition based on Little's formula, and an inequality for every $t \in T_m$ based on the associated semi-Markov process. Each pair $\{p, t\}$ where $p^\bullet = t$ and $t \in T_s$ can be seen as a simple queuing system for which Little's formula [6] can be directly applied [3]:

$$\overline{M}(p) = (Pre(p,t)\cdot Th(t))\cdot \overline{R}(p) \qquad (6)$$

where $\overline{R}(p)$ is the average residence time at place p, i.e., the average time spent by a token in p. The average residence time is the sum of the average waiting time due to a possible synchronization and the average service time which in our case is $\delta(t)$. Therefore the service time $\delta(t)$ is a lower bound for the average residence time. On the other hand, the average marking of the input places of simple immediate transitions is clearly non-negative even if they are in ECR. This leads to the inequality:

$$\delta(t)\cdot Th(t) \le \frac{\overline{M}(p)}{Pre(p,t)} \;\; \forall\, \{p,t\} \text{ such that } p\in{}^\bullet t,\;\; t\in T_s \qquad (7)$$

For each $t \in T_m$ a linear inequality exists that relates the throughput of t to the average marking of its input places.

Theorem 2 *Let $\langle \mathcal{N}^\tau, M_0\rangle$ be a TMEC. Let t be a multi-guarded transition of the TMEC, then:*

$$\delta(t)\cdot Th(t) \le \sum_{p\in{}^\bullet t} \alpha(\{p\})\cdot \frac{\overline{M}(p)}{Pre(p,t)} \qquad (8)$$

Proof: See Appendix. □

One can combine constraints (7) and (8) on the throughput and on the average marking, to build a Linear Programming Problem (LP) that maximizes a variable, ϕ_1, corresponding to the throughput of transition t_1, $\phi_1 = Th(t_1)$. One scalar variable suffices since the throughput of all transitions is related by the vector of visit ratios $v^{(1)}$ (Proposition 1). The resulting LP is:

$$\begin{aligned}
&max \;\; \phi_1 :\\
&\delta(t_j)\cdot\phi_1\cdot v^{(1)}(j) \le \sum_{p\in{}^\bullet t_j} \alpha(\{p\})\cdot\frac{\widehat{M}(p)}{Pre(p,t_j)} \;\forall\, t_j\in T_m\\
&\delta(t_j)\cdot\phi_1\cdot v^{(1)}(j) \le \frac{\widehat{M}(p)}{Pre(p,t_j)} \;\forall\{p,t_j\},\; p\in{}^\bullet t_j, t_j\in T_s\\
&\widehat{M} = M_0 + \mathbb{C}\cdot\sigma,\;\; \sigma\ge 0
\end{aligned} \qquad (9)$$

where σ represents the firing count vector that drives the system from the initial marking, M_0, to the estimated average marking, $\widehat{M}$. The LP (9) always has solution since all its constraints must hold in the steady state. Given that the throughput variable, ϕ_1, is maximized, the vector $\phi_1\cdot v^{(1)}$ is an upper bound for the throughput of transitions.

4.3 Reduction Rule

This subsection presents a rule to merge the input places of a multi-guarded transition in order to simplify (9).

Let ${}^\bullet t_j = \{p_1, \ldots, p_n\}$ for a given $t_j \in T_m$. The term $\sum_{p \in {}^\bullet t_j} \alpha(\{p\}) \cdot \frac{\widehat{M}(p)}{Pre(p, t_j)}$ in (9) can be substituted by a single variable $\widehat{M}_{tj}$ such that $\widehat{M}_{tj} = \sum_{p \in {}^\bullet t_j} \alpha(\{p\}) \cdot \frac{\widehat{M}(p)}{Pre(p, t_j)}$. This is equivalent to substituting the rows of M_0, Pre and $Post$ corresponding to $\{p_1, \ldots, p_n\}$ by a single row that is a linear combination of them (in such a linear combination the weight of the row corresponding to p_i is $\frac{\alpha(\{p_i\})}{Pre(p_i, t_j)}$). Let M_{0r}, Pre_r, $Post_r$ and $\mathbb{C}_r = Post_r - Pre_r$ be the arrays obtained after performing such a substitution on the set of input places of every multi-guarded transition. Figure 6(a) shows the graphical interpretation of the reduction.

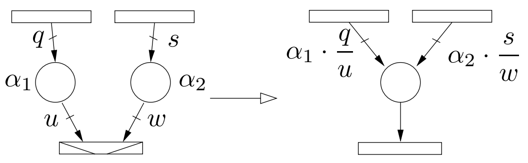

Figure 6. Reduction rule: the multi-guarded transition has two guards with probabilities α_1 and α_2.

Now the LP (9) can be expressed as:

$$max\{\, \phi_1 \mid \phi_1 \cdot D^{(1)} \leq M_{0r} + \mathbb{C}_r \cdot \sigma,\ \sigma \geq 0 \,\} \qquad (10)$$

where $D^{(1)}(p) = \delta(p^\bullet) \cdot v^{(1)}(p^\bullet) \cdot Pre_r(p, p^\bullet)$ if $|p^\bullet| = 1$, $D^{(1)}(p) = 0$ if $|p^\bullet| > 1$ (recall that if $|p^\bullet| > 1$ then p is an input place of a simple immediate transition in ECR). Let us define $\rho = \frac{1}{\phi_1}$, and $\sigma' = \frac{\sigma}{\phi_1}$, then (10) becomes:

$$min\{\, \rho \mid D^{(1)} \leq \rho \cdot M_{0r} + \mathbb{C}_r \cdot \sigma',\ \sigma' \geq 0 \,\} \qquad (11)$$

The dual of (11) is:

$$max\{\, Y \cdot D^{(1)} \mid Y \cdot \mathbb{C}_r \leq 0,\ Y \cdot M_{0r} \leq 1,\ Y \geq 0 \,\} \quad (12)$$

One theorem of the alternatives [8] states that $\exists\ X > 0$ such that $\mathbb{C}_r \cdot X \geq 0$ iff $\forall\, Y \geq 0$ such that $Y \cdot \mathbb{C}_r \leq 0$ then $Y \cdot \mathbb{C}_r = 0$. Since we are dealing with consistent systems, $Y \cdot \mathbb{C}_r \leq 0$ can be replaced by $Y \cdot \mathbb{C}_r = 0$. Furthermore, since the objective function $Y \cdot D^{(1)}$ is maximized, the solution must satisfy $Y \cdot M_{0_r} = 1$ (otherwise a "more optimal" solution is possible with $\beta \cdot Y$, $\beta = 1/(Y \cdot M_{0r})$).

Theorem 3 *Let $\langle \mathcal{N}^\tau, M_0 \rangle$ be a TMEC. Let $\mathbb{C}_r$, $D^{(1)}$ and M_{0r} be the matrices obtained after performing the above reduction rule. Let Γ be the solution of:*

$$\Gamma = max\{\, Y \cdot D^{(1)} \mid Y \cdot \mathbb{C}_r = 0,\ Y \cdot M_{0r} = 1,\ Y \geq 0 \,\} \qquad (13)$$

Then, $Th(\mathcal{N}^\tau, M_0) \leq \frac{1}{\Gamma} \cdot v^{(1)}$.

This way, the computation of an upper throughput bound for a TMEC can be achieved in polynomial time by solving the LP (13), which actually represents a search for the bottleneck P-semiflow of the reduced net.

5 Experimental Results

This section shows the throughput bounds obtained for three different TMEC systems.

5.1 Parallel Program

Consider again the system in Figure 2 to illustrate the computation of throughput bounds. Let us rename $t_a, \ldots, t_f$ by $t_1, \ldots, t_6$. Let the delays be $\delta(t_1) = \delta(t_3) = \delta(t_5) = \delta(t_6) = 1$, $\delta(t_2) = 2$, $\delta(t_4) = 3$, and the probability of the guards be $\alpha(\{p_{de}\}) = \gamma$, $\alpha(\{p_{ae}\}) = 1 - \gamma$.

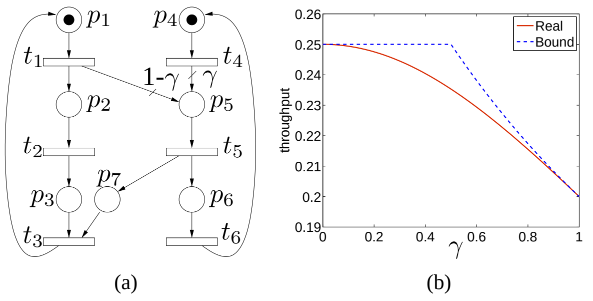

Figure 7. (a) Net obtained after applying the reduction rule to the net in Figure 2. (b) Real throughput and throughput bound for $\gamma \in (0,\ 1)$.

Figure 7(a) shows the net obtained after applying the reduction rule to the net in Figure 2. For the obtained system: $M_{0r} = (1,\ 0,\ 0,\ 1,\ 0,\ 0)$, $v^{(1)} = (1,\ 1,\ 1,\ 1,\ 1,\ 1)$, $D^{(1)} = (1,\ 2,\ 1,\ 3,\ 1,\ 1,\ 1)$, and two minimal P-semiflows exist which, normalized to satisfy $Y \cdot M_{0r} = 1$, are $Y_1 = (1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0)$ and $Y_2 = (1 - \gamma,\ 0,\ 0,\ \gamma,\ 1,\ \gamma,\ 1 - \gamma)$. Thus, $Y_1 \cdot D^{(1)} = 4$ and $Y_2 \cdot D^{(1)} = 3 + 2 \cdot \gamma$. Then, according to Theorem 3, the throughput of the system is bounded by $Th(\mathcal{N}^\tau, M_0) \leq min\{1/4, 1/(3 + 2 \cdot \gamma)\}$. Figure 7(b) shows the system throughput (see Equation (4)) and the obtained throughput bound for $\gamma \in (0,\ 1)$.

5.2 Multipliers with Early Evaluation

A digital multiplier can yield a zero result as soon as one of its inputs is known to be zero. The system in Figure 8(a) models three processes, $P1$, $P2$, $P3$, that interchange data. Each process ($P1$, $P2$, $P3$) reads a natural number from an input file ($FI1$, $FI2$, $FI3$) at the beginning of each iteration. Then, each pair of numbers read from $FI1$ and $FI2$(from $FI2$ and $FI3$) is multiplied by process $P1$($P3$). Afterwards, each process applies a given function, f_1, f_2, f_3, to the multiplication results, and writes the resulting number in the output files $FO1$, $FO2$, and $FO3$. In order to model early evaluation of the multipliers with multi-guarded transitions, one of the inputs is assumed to contain always strictly positive numbers. It will be assumed that every natural number in $FI2$ is strictly positive, and that a number in $FI1$($FI3$) is zero with probability $\beta(\gamma)$.

The early evaluation of the multiplier in $P1$ can be modeled as follows: 1) Add a transition t_3 with $t_3^\bullet = p_b$ and

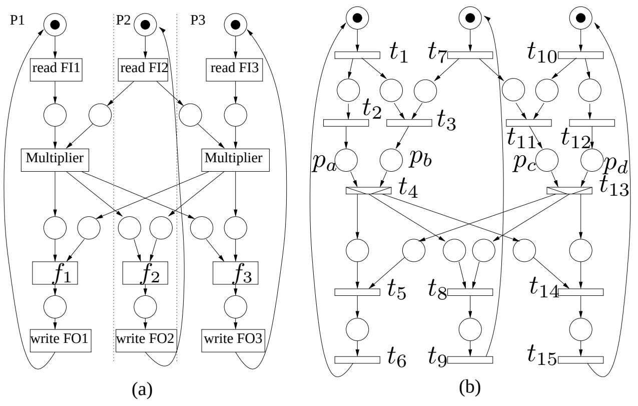

Figure 8. (a) A system with early evaluated multipliers. (b) Its associated TMEC system.

delay equal to the time required to perform the multiplication if both inputs are positive; 2) Add a transition t_2 with $t_2^\bullet = p_a$ and delay equal to the time required to test whether the input from $F1$ is zero; 3) Add an immediate transition t_4 with two guards, $G(t_4) = \{\{p_a\}, \{p_b\}\}$. A token in p_a means that the result of the 0-*test* is available. Given that the result of the test is true with probability β, transition t_4 can fire without waiting for a token p_a with probability β, hence, $\alpha(\{p_a\}) = \beta$ and $\alpha(\{p_b\}) = 1 - \beta$. Figure 8(b) shows the TMEC system obtained after modeling the early evaluated multipliers in $P1$ and $P3$ with the described steps.

The visit ratio of every transition of the obtained TMEC is equal to 1, i.e., every transition has the same steady state throughput. Let the delays associated with the transitions be $\delta = (1, 0.2, 4.5, 0, 1, 1, 1.5, 1.5, 1.5, 1.2, 5, 0.3, 0, 1.2, 1.2)$.

$\beta = 0.2$			
γ	**Real**	**Bound**	**error**
0.2	0.1088	0.1193	9.69%
0.5	0.1145	0.1276	11.36%
0.8	0.1210	0.1276	5.41%

$\beta = 0.8$			
γ	**Real**	**Bound**	**error**
0.2	0.1165	0.1193	2.41%
0.5	0.1388	0.1493	7.55%
0.8	0.1717	0.1992	16.02%

Table 1. Bounds for the multipliers in Figure 8(b).

The bounds yielded by the LP (13) for several values of β and γ of the TMEC system are reported in the column *Bound* of Table 1. Column *Real* is the real throughput of the TMEC system, *error* is the relative error of the bound with respect to the real throughput.

The column *Real* was obtained by simulating the TMEC system. The simulations were carried out using the independent replication method [5]. The precision of the computed throughput was set to 1% with a confidence level of 95%. All the experiments were performed in the Matlab 7.3 environment running on Linux in a 2.0 GHz processor. The CPU time required to obtain each bound was less than 0.01 seconds, the average time for each simulation (including all the necessary replications) was 75 seconds.

5.3 System of Communicating Modules

The TMEC system in Figure 9 represents a "System of Asynchronously communicating Modules" (SAM) adapted from [2]. It consists of three modules, A, C, and D, that communicate through buffers b_1, b_2, b_3, b_4, b_5, and b_6. We will assume that each firing of $t_a(t_c,\ t_d)$ requires data either from b_1 or from $b_2(b_3$ or b_4, b_5 or $b_6)$. Hence, t_a, t_c, and t_d are multi-guarded transitions with $G(t_a) = \{\{a_1, b_1\}, \{a_1, b_2\}\}$, $G(t_c) = \{\{c_1, b_3\}, \{c_1, b_4\}\}$, and $G(t_d) = \{\{d_1, b_5\}, \{d_1, b_6\}\}$.

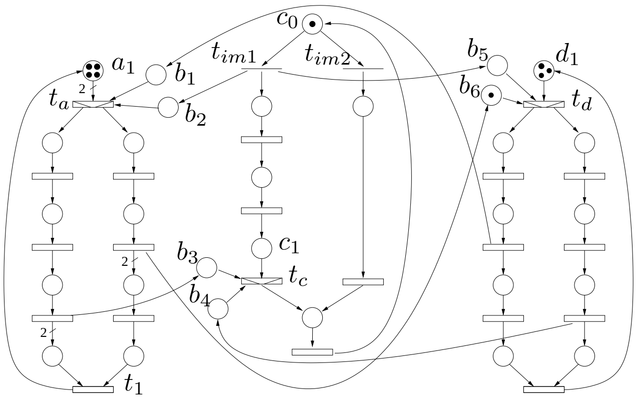

Figure 9. Asynchronously communicating modules.

Let us assume that $t_a(t_c,\ t_d)$ requires data from $b_1(b_3,\ b_5)$ with probability $0.3(0.3, 0.3)$. Let the routing for immediate transitions be $r_{im1} = r_{im2} = 0.5$, and the delays of non-immediate transitions be equal to 1. Table 2 reports the throughput bounds obtained for different initial markings of c_0. Column $M_0(c_0)$ is the initial marking of c_0, column *Real* is the real throughput of t_1, column *Bound* is the obtained bound for t_1, and column *error* is the relative error of the bound with respect to the real throughput.

$M_0(c_0)$	**Real**	**Bound**	**error**
1	0.1290	0.1417	9.84%
2	0.2200	0.2250	2.25%
3	0.2833	0.3083	8.82%
4	0.3180	0.3409	7.21%

Table 2. Bounds for the TMEC system in Figure 9.

As in the previous example, the column *Real* is obtained by using the independent replication method with precision of 1% and confidence level of 95%. The average CPU time for the computation of bounds was less that 0.01 seconds, and 97 seconds for the simulations.

6 Conclusions

Early evaluation aims at enhancing the system performance by executing operations as soon as possible. In Petri nets, multi-guarded transitions can be used to model early evaluation of some operations, e.g., multiplexers. In contrast to conventional transitions, a multi-guarded transition becomes enabled when a selected subset of input places have enough tokens. Thus, multi-guarded transitions can fire even if some input places are not marked.

This paper has focused on the class of Equal Conflict Petri nets. For such a class, a linear expression relating the throughput of a multi-guarded transition to the average marking of its input places has been obtained. The use of such an expression allows one to compute throughput bounds in polynomial time by means of a linear programming problem.

Acknowledgments

The author would like to thank Jordi Cortadella for his valuable suggestions and comments.

References

[1] C.F. Brej and J.D. Garside. Early output logic using anti-tokens. In *Int. Workshop on Logic Synthesis*, pages 302–309, May 2003.

[2] J. Campos, S. Donatelli, and M. Silva. Structured solution of asynchronously communicating stochastic modules. *Software Engineering*, 25(2):147–165, 1999.

[3] J. Campos and M. Silva. Structural Techniques and Performance Bounds of Stochastic Petri Net Models. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 609 of *Lecture Notes in Computer Science*, pages 352–391. Springer, 1992.

[4] J. Júlvez, J. Cortadella, and M. Kishinevsky. Performance analysis of concurrent systems with early evaluation. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, November 2006.

[5] Averill M. Law. *Simulation Modeling and Analysis*. McGraw-Hill, 2007.

[6] J. D. C. Little. A proof of the queueing formula *L*= λ *W*. *Operations Research*, 9:383–387, 1961.

[7] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[8] K. G. Murty. *Linear Programming*. Wiley and Sons, 1983.

[9] E. Teruel and M. Silva. Liveness and Home States in Equal Conflict Systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, Lecture Notes in Computer Science, pages 415–432.

[10] Ronald W. Wolff. *Stochastic modeling and the theory of queues*. Prentice Hall, 1989.

7 Appendix: Proof of Theorem 2

This section presents a proof of Theorem 2. First, some auxiliary definitions are introduced. Then, a technical lemma is presented. Finally, the linear relationship is obtained.

7.1 Auxiliary Definitions

Each input place p of a multi-guarded transition t can be seen as a queue where tokens are arranged in cells, each cell having capacity $Pre(p, t)$. For instance the capacity of the cells in the queue corresponding to p_1 in Figure 10(a) is 3, see Figure 10(b).

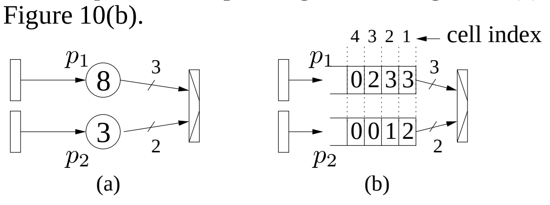

Figure 10. Each place is seen as a queue where tokens are arranged in cells.

Cells are indexed according to their proximity to the multi-guarded transition, see Figure 10(b). When a cell i is full, the new incoming tokens are stored in cell $i + 1$.

Consider the net in queue-like form in Figure 11 with $G(t) = \{\{p_1\}, \{p_2\}\}$ to show how the marking of the different cells evolves. Assume that the guard for the first firing of t is $\{p_1\}$ (see above each cell in Figure 11). Thus, the first firing requires 2 tokens in p_1. Hence, t is not enabled in Figure 11(a), but becomes enabled when t_1 fires, Figure 11(b). The firing of t removes 2 tokens from p_1 and 1 token from p_2. After the firing of t, the first cell of each place becomes empty, and hence every cell is shifted to the right, Figure 11(c). Assume that t_2 fires from the marking in Figure 11(c), and the guards for the next two firings are $\{p_2\}$ and $\{p_2\}$, Figure 11(d). Then, the enabling degree of t becomes 2, i.e., 2 firing instances are enabled in Figure 11(d). The firing of the 2 firing instances of t drives the system to the marking in Figure 11(e). If $t_1(t_2)$ fires from Figure 11(e) one token will go to cell 1 of p_1, one token will go to cell 2, and every cell will be shifted to the right(one token will go to cell 3 of p_2 and one token will go to cell 4).

The marking of the l^{th} cell of p_i is denoted as $M(p_i, l)$, and the guard for the l^{th} cell is denoted as $g(t, l)$. For each place p_i and each cell l, we define three boolean functions, $p^+_{i,l}$, $p^0_{i,l}$, and $p^-_{i,l}$, that take a marking M as input argument:

- $p^+_{i,l}(M) \leftrightarrow\ M(p_i, l) = Pre(p_i, t)$
- $p^0_{i,l}(M) \leftrightarrow\ 0 \leq M(p_i, l) < Pre(p_i, t)$
- $p^-_{i,l}(M) \leftrightarrow\ M(p_i, l) < 0$

If the argument M of the above functions is evident from the context, it will be dropped for clarity. We use the shorthand α_i to denote $\alpha(\{p_i\})$, and the boolean variable $g_{i,l}$ that is true iff $g(t, l) = \{p_i\}$. The fact that $g(t, l) \neq p_i$ is denoted as $not(g_{i,l})$. The next statement illustrates the use of the introduced notation: the firing instance corresponding

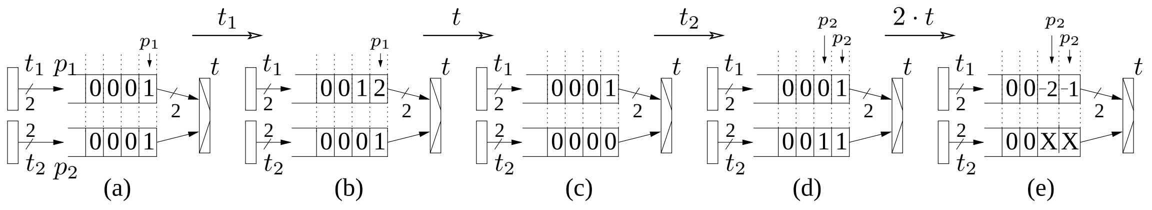

Figure 11. Each pair of cells with same index corresponds to a firing instance.

to the l^{th} cell is enabled at M' iff there exists $p_i \in {}^\bullet t$ such that $(p^+_{i,l}(M') \wedge g_{i,l})$ (or simply $(p^+_{i,l} \wedge g_{i,l})$ given that the argument of $p^+_{i,l}$ is clearly M').

By $prob(statement)$ we denote the fraction of time *statement* holds in the steady state. This way, as an example, $prob(p^+_{j,l} \wedge p^0_{k,l} \wedge g_{k,l})$ is the fraction of time at which the system marking satisfies that $M(p_j, l) = Pre(p_j, t)$, $0 \le M(p_k, l) < Pre(p_k, t)$, and $g(t, l) = p_k$.

7.2 Preliminary Lemma

This section focuses on cells with the same index, then for clarity $p^+_{i,l}$, $p^0_{i,l}$, $p^-_{i,l}$, $g_{i,l}$ are shortened to p^+_i, p^0_i, p^-_i, g_i.

Lemma 4 *Let $\langle \mathcal{N}^\tau, M_0\rangle$ be a TMEC system. Let t be a multi-guarded transition of the TMEC and $p_j, p_k \in {}^\bullet t$, then for each cell l it holds that:*

$$\frac{\alpha_k}{\alpha_j} \le \frac{prob(p^+_j \wedge p^0_k \wedge g_k)}{prob(((p^+_j \wedge p^0_k) \vee p^-_k) \wedge g_j)} \qquad (14)$$

Proof: The proof is based on the semi-Markov process associated with the TMEC system. Without loss of generality, we assume that the guard for the l^{th} cell is selected when there exists $p \in {}^\bullet t$ such that $M(p, l) = Pre(p, t)$. For instance, in Figure 4 a guard for cell 1 is selected when leaving s_1 and s_5. Let $H(t, l)$ denote the set of states of the semi-Markov process from which a guard for the l^{th} cell is selected, e.g., $H(t_e, 1) = \{s_1, s_5\}$ in Figure 4. Then, among the successors of $h \in H(t, l)$, e.g., state s_1, there are $|{}^\bullet t| = n$ states, $h_1, \ldots, h_n$ that only differ on the guard selected for the l^{th} cell, e.g., states s_2 and s_8.

If the statement $(p^+_j \wedge p^0_k)$ does not hold for any successor of $H(t, l)$, then in any successor $p^+_j \rightarrow p^+_k$, consequently $prob(p^-_k \wedge g_j) = 0$ and the lemma trivially holds. Assume there exists $h \in H(t, l)$, such that a successor h_j of h exists at which $(p^+_j \wedge p^0_k \wedge g_j)$ holds, e.g., $(p^+_{ae} \wedge p^0_{de} \wedge g_{ae})$ holds in s_2. Then, a successor h_k, e.g., s_8, of h that only differs from h_j in the guard must exists, namely at h_k the statement $(p^+_j \wedge p^0_k \wedge g_k)$ holds. Given that the guard selected for cell l does not affect the system evolution neither from h to h_j nor from h to h_k, $\dfrac{\alpha_k}{\alpha_j}$ is the ratio of visits between states h_k and h_j. The lemma is true if the time fraction during which $(p^+_j \wedge p^0_k \wedge g_k)$ holds from h_k is not less than the time fraction during which $(((p^+_j \wedge p^0_k) \vee p^-_k) \wedge g_j)$ holds from h_j. The statement $(p^+_j \wedge p^0_k)$ will hold from h_k until $Pre(p_k, t) - M_{hk}(p_k, l)$ tokens are put in cell l of p_k, where $M_{hk}(p_k, l)$ is the number of tokens in cell l of p_k at state h_k. The statement $(((p^+_j \wedge p^0_k) \vee p^-_k) \wedge g_j)$ will hold from h_j until $Pre(p_k, t) - M_{hj}(p_k, l)$ tokens are put in in cell l of p_k, where $M_{hj}(p_k, l)$ is the number of tokens in cell l of p_k at state h_j. Given that the marking is the same at h_j and h_k, the same amount of tokens makes false both statements.

States h_j, e.g., s_2, and h_k, e.g., s_8, only differ in the guards, and therefore, in the fact that t is enabled in h_j, and will fire after $\delta(t)$ time units, but it is not enabled in h_k. Since all conflicts are equal, the routing of tokens does not depend on the mentioned firing of t, i.e., the same routings happen from h_j and from h_k. In other words, such firing of t just makes the system progress faster. Hence, the time fraction during which $(p^+_j \wedge p^0_k \wedge g_k)$ holds, e.g., $\psi(s_8) + \psi(s_9)$, from h_k cannot be less than the time fraction during which $(((p^+_j \wedge p^0_k) \vee p^-_k) \wedge g_j)$ holds, e.g., $\psi(s_2) + \psi(s_3)$, from h_j. $\square$

7.3 Proof of Theorem

Theorem 2: Let $\langle \mathcal{N}^\tau, M_0\rangle$ be a TMEC system. Let t be a multi-guarded transition of the TMEC system, then:

$$\delta(t) \cdot Th(t) \le \sum_{p \in {}^\bullet t} \alpha(\{p\}) \cdot \frac{\overline{M}(p)}{Pre(p, t)} \qquad (15)$$

Proof: Since infinite server semantics is adopted the average enabling degree of t is: $\overline{enab}(t) = \sum_{l=1}^{\infty} \overline{enab}(t, l)$ where $\overline{enab}(t, l)$ is the average enabling degree of the l^{th} cell. Assume without loss of generality that ${}^\bullet t = \{p_1, \ldots, p_n\}$. The value of $\overline{enab}(t, l)$ can be expressed as:

$$\begin{aligned}
\overline{enab}(t, l) &= \sum_{p_j \in {}^\bullet t} prob(p^+_{j,l} \wedge\ g_{j,l}) \\
&= \sum_{p_j \in {}^\bullet t} \Big(prob(p^+_{j,l}) - prob(p^+_{j,l} \wedge\ not(g_{j,l}))\Big) \\
&= \sum_{p_j \in {}^\bullet t} \Big(prob(p^+_{j,l}) - \big((1 - \alpha_j) \cdot prob(p^+_{j,l} \wedge\ not(g_{j,l})) \\
&\qquad + \alpha_j \cdot prob(p^+_{j,l} \wedge\ not(g_{j,l}))\big)\Big)
\end{aligned} \qquad (16)$$

The subindex l is omitted in the following developments for clarity. Notice that if p^+_j holds, the transition has not fired yet, then $p_k^0 \vee p_k^+$ holds for every $k \in \{1,\ldots,n\}$. Hence:

$$prob(p_j^+) = prob(p_j^+ \wedge p_k^0) + prob(p_j^+ \wedge p_k^+) \qquad (17)$$

for every $k \in \{1,\ldots,n\}$, and:

$$prob(p_j^+ \wedge g_k) = prob(p_j^+ \wedge p_k^0 \wedge g_k) + prob(p_j^+ \wedge p_k^+ \wedge g_k) \qquad (18)$$

for every $k \in \{1,\ldots,n\}$. Then, the term $\alpha_j \cdot prob(p_j^+ \wedge\ not(g_j))$ in (16) can be expanded as:

$$\alpha_j \cdot prob(p_j^+ \wedge\ not(g_j)) = \alpha_j \cdot \sum_{k=1,k\neq j}^{n} prob(p_j^+ \wedge g_k)$$

$$= \alpha_j \cdot \sum_{k=1,k\neq j}^{n} \big(prob(p_j^+ \wedge p_k^0 \wedge g_k) + prob(p_j^+ \wedge p_k^+ \wedge g_k)\big)$$

$$= \sum_{k=1,k\neq j}^{n} \alpha_j \cdot prob(p_j^+ \wedge p_k^+ \wedge g_k) + \sum_{k=1,k\neq j}^{n} \alpha_j \cdot prob(p_j^+ \wedge p_k^0 \wedge g_k) \qquad (19)$$

The inequality (14) provided by Lemma 4 can be written as:

$$\alpha_j \cdot prob(p_j^+ \wedge p_k^0 \wedge g_k) \geq\ \alpha_k \cdot prob(p_j^+ \wedge p_k^0 \wedge g_j) + \alpha_k \cdot prob(p_k^- \wedge g_j) \qquad (20)$$

The substitution of $\alpha_j \cdot prob(p_j^+ \wedge p_k^0 \wedge g_k)$ in (19) by the expression on the right of (20) yields:

$$\alpha_j \cdot prob(p_j^+ \wedge\ not(g_j)) \geq \sum_{k=1,k\neq j}^{n} \alpha_j \cdot prob(p_j^+ \wedge p_k^+ \wedge g_k) + \sum_{k=1,k\neq j}^{n} \alpha_k \cdot prob(p_j^+ \wedge p_k^0 \wedge g_j) + \sum_{k=1,k\neq j}^{n} \alpha_k \cdot prob(p_k^- \wedge g_j) \qquad (21)$$

Recall that (21) is an expression resulting from the development of $\alpha_j \cdot prob(p_j^+ \wedge\ not(g_j))$ in (16). For each pair of expressions (21) resulting from the development of $\alpha_j \cdot prob(p_j^+ \wedge\ not(g_j))$ and $\alpha_k \cdot prob(p_k^+ \wedge\ not(g_k))$ such that $k \neq j$, we swap $\alpha_j \cdot prob(p_j^+ \wedge p_k^+ \wedge g_k)$ with $\alpha_k \cdot prob(p_j^+ \wedge p_k^+ \wedge g_j)$, and we swap $\alpha_k \cdot prob(p_k^- \wedge g_j)$ with $\alpha_j \cdot prob(p_j^- \wedge g_k)$. Thus, the expression associated with $\alpha_j \cdot prob(p_j^+ \wedge\ not(g_j))$ becomes:

$$\sum_{k=1,k\neq j}^{n} \alpha_k \cdot prob(p_j^+ \wedge p_k^0 \wedge g_j) + \sum_{k=1,k\neq j}^{n} \alpha_k \cdot prob(p_j^+ \wedge p_k^+ \wedge g_j) + \sum_{k=1,k\neq j}^{n} \alpha_j \cdot prob(p_j^- \wedge g_k) \qquad (22)$$

Notice that the reasonings used to obtain (18) can also be used to deduce that:

$$prob(p_j^+ \wedge g_j) = prob(p_j^+ \wedge p_k^0 \wedge g_j) + prob(p_j^+ \wedge p_k^+ \wedge g_j)$$

for every $k \in \{1,\ldots,n\}$. Notice also that if p_j is selected as guard it cannot become negatively marked after the firing, i.e., $prob(p_j^- \wedge g_j) = 0$, therefore:

$$\sum_{k=1,k\neq j}^{n} prob(p_j^- \wedge g_k) = prob(p_j^-)$$

Hence, (22) can be rewritten as:

$$= \sum_{k=1,k\neq j}^{n} \alpha_k \cdot prob(p_j^+ \wedge g_j) + \alpha_j \cdot prob(p_j^-)$$

$$= (1-\alpha_j) \cdot prob(p_j^+ \wedge g_j) + \alpha_j \cdot prob(p_j^-) \qquad (23)$$

After substituting $\alpha_j \cdot prob(p_j^+ \wedge not(g_j))$ by (23) in (16), $\overline{enab}(t,l)$ becomes:

$$\overline{enab}(t,l) \leq \sum_{p_j \in {}^\bullet t} \Big(prob(p_j^+) - \big((1-\alpha_j) \cdot prob(p_j^+ \wedge\ not(g_j)) + (1-\alpha_j) \cdot prob(p_j^+ \wedge g_j) + \alpha_j \cdot prob(p_j^-)\big)\Big)$$

$$= \sum_{p_j \in {}^\bullet t} \alpha_j \cdot \big(prob(p_j^+) - prob(p_j^-)\big)$$

We are ready to write an expression for the overall average enabling degree under infinite server semantics:

$$\overline{enab}(t) = \sum_{l=1}^{\infty} \overline{enab}(t,l)$$

$$\leq \sum_{l=1}^{\infty} \sum_{p_j \in {}^\bullet t} \alpha_j \cdot \big(prob(p_{j,l}^+) - prob(p_{j,l}^-)\big)$$

$$= \sum_{p_j \in {}^\bullet t} \alpha_j \cdot \sum_{l=1}^{\infty} \big(prob(p_{j,l}^+) - prob(p_{j,l}^-)\big)$$

$$= \sum_{p_j \in {}^\bullet t} \alpha_j \cdot \Big(\sum_{l=1}^{\infty} prob\Big(\frac{M(p_j,l)}{Pre(p_j,t)} = 1\Big) - \sum_{l=1}^{\infty} prob\Big(\frac{M(p_j,l)}{Pre(p_j,t)} < 0\Big)\Big)$$

$$\leq \sum_{p_j \in {}^\bullet t} \alpha_j \cdot \Big(\sum_{i=1}^{\infty} \frac{i}{Pre(p_j,t)} \cdot prob\big(M(p_j) = i\big) - \sum_{i=-1}^{-\infty} \frac{i}{Pre(p_j,t)} \cdot prob\big(M(p_j) = i\big)\Big)$$

$$= \sum_{p_j \in {}^\bullet t} \alpha_j \cdot \frac{\overline{M}(p_j)}{Pre(p_j,t)}$$

From (3) the desired expression is obtained:

$$\delta(t) \cdot Th(t) = \overline{enab}(t) \leq \sum_{p_j \in {}^\bullet t} \alpha_j \cdot \frac{\overline{M}(p_j)}{Pre(p_j,t)}$$

$\square$