Computing Laboratory

Modelling Systems with Dynamic Priorities

M. Koutny

Technical Report Series
No 353 December, 1991
Modelling Systems with Dynamic Priorities

M. Koutny

Abstract

In the design of concurrent systems it is sometimes desirable to specify priority constraints to resolve conflicts between simultaneously enabled actions. e.g., operating systems employ priorities to control the execution of jobs waiting for processing, while programming languages provide primitives to specify the preference of the execution of one program's module over another one. Priority systems can be roughly divided into two classes: those employing static priority relation, and those employing dynamic priority relation. In the former all priority constraints are determined before the operation of the system or program begins, i.e., at the compile time, while in the latter the priority relation can be changed during the execution, i.e., priorities are determined at the run time. In this paper we will be concerned with the dynamic priority systems and investigate their formal semantics using Petri nets.
Bibliographical details

KOUTNY, Maciej

Modelling Systems with Dynamic Priorities.

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Laboratory, 1991.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 353)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.
Computing Laboratory. Technical Report Series. 353

Abstract

In the design of concurrent systems it is sometimes desirable to specify priority constraints to resolve conflicts between simultaneously enabled actions. e.g., operating systems employ priorities to control the execution of jobs waiting for processing, while programming languages provide primitives to specify the preference of the execution of one program's module over another one. Priority systems can be roughly divided into two classes: those employing static priority relation, and those employing dynamic priority relation. In the former all priority constraints are determined before the operation of the system or program begins, i.e., at the compile time, while in the latter the priority relation can be changed during the execution, i.e., priorities are determined at the run time. In this paper we will be concerned with the dynamic priority systems and investigate their formal semantics using Petri nets.

About the author

Dr. M. Koutny joined the Computing Laboratory in March 1985, where he is a Lecturer.

Suggested keywords

CONCURRENCY
PARTIAL ORDER SEMANTICS
PETRI NETS
PRIORITIES

Suggested classmarks (primary classmark underlined)

Dewey (18th): 001.6404 511.5 001.6424
U.D.C. 519.687 519.711 519.767
Modelling Systems with Dynamic Priorities

Maciej Koutny
Computing Laboratory
The University
Newcastle upon Tyne NE1 7RU, U.K.

Abstract
In this paper we discuss concurrent systems with dynamic priorities, i.e. we allow the priority relation to change as the system evolves. We identify two classes of such systems, state-controlled and event-controlled ones. We define their non-sequential semantics (in terms of step sequences) which reflects both the priority constraints and concurrency specifications. It is then shown that for a given prioritised system it is possible to construct an equivalent non-prioritised one. The systems dealt with in this paper are safe Petri nets augmented with a priority specification.

1 Introduction
In the design of concurrent systems it is often desirable to specify priority constraints to resolve conflicts between simultaneously enabled actions. E.g., operating systems employ priorities to control the execution of jobs waiting for processing, while programming languages provide primitives to specify the relative preferences for the execution of different parts of the program. Priority systems can be roughly divided into Static Priority Systems and Dynamic Priority Systems. In the former all priority constraints are determined before the operation of the system or program begins, i.e., at the compile time, while in the latter the priority relation can be
changed during its execution, i.e., priorities are determined at the run time.

The meaning of static priority systems is now rather well-understood. There have been a number of papers dealing with their semantics in the context of different models of concurrency, including Petri nets [Ha,BK], COSY [Ja,Ok], process algebras [BBK,CH,BG,CW] and programming languages [Ba,Ca,La].

In this paper we will define a priority system as a pair of the form

$$\Sigma_{PR} = (\Sigma, p)$$

where $\Sigma$ is a base non-prioritised system (Petri net) and $p$ priority specification. A system with static priorities is one for whom $p$ is simply a binary relation on the actions of $\Sigma$. We will use $a < b$ to denote that $b$ has higher priority than $a$, i.e. $a$ can occur only if $b$ is not enabled. For systems with dynamically changing priority relation, $p$ might be defined in different ways. In this paper we will look at two intuitively appealing mechanisms of modelling dynamic priorities: the event-controlled and state-controlled priority specifications.

In the event-based approach, for each priority constraint $a < b$ there will be actions whose occurrences can activate and suspend it. In this case $p = \pi_{EC} = (\rho_0, \text{On}, \text{Off})$, where $\rho_0$ is an initial priority relation, and $\text{On}$ and $\text{Off}$ are two mappings such that $\text{On}(a,b)$ and $\text{Off}(a,b)$ are respectively the sets of actions which can activate and suspend $a < b$. As a result, at any point in the evolution of the system, the current priority relation will depend on its past history. In particular, it may happen that two executions leading to the same global state in the base system $\Sigma$ have different continuations. Consider, rather informally, the system $\Sigma$ in Fig. 1.1(a) together with the priority specification $\rho_0 = \emptyset$ and $\text{On}(a,b) = \{c\}$. After executing $c$ the priority constraint $a < b$ becomes active and $a$ can no longer be executed, leading to a reachability graph (Fig.1.1(b)) which is not included in the reachability graph of the base system (Fig.1.1(c)).

In the state-based approach $p = \pi_{SC}$ is a relation which, for every potential priority constraint $a < b$, specifies those local states in which it holds. The
reachability graph of a state-controlled priority system is always included in the reachability graph generated by the base system. Consider again the system of Fig. 1.1(a) and a priority specification such that \( a < b \) holds if place \( p \) is marked. This priority system will generate a reachability graph shown in Fig. 1.1(d).

In this paper we will discuss non-sequential semantics of priority systems expressed in terms of step sequences [SM,RV]. For both event- and state-controlled priority systems we will formalise their semantics and provide a translation to equivalent non-prioritised systems. As the system model we use Petri nets, and the base system \( \Sigma \) will always be a safe Petri net.

![Diagram](image)

Figure 1.1: Reachability graphs (b,c,d) only show arcs labelled by single transitions.
2 Preliminaries

2.1 Priorities and Concurrency

To illustrate the problems posed by the interplay between the priority and concurrency specifications, we consider a static priority system

$$\Sigma_S = (\Sigma, n_S)$$

where $\Sigma$ is as in Fig. 2.1, and $n_S = \{(c,b)\}$. To obtain the step sequences generated by $\Sigma_S$, we take the step sequences of $\Sigma$ and delete those which are inconsistent with the priority specification. Since the step sequences of $\Sigma$ are

$$\text{steps}(\Sigma) = \{\lambda, \{a\}, \{c\}, \{a, c\}, \{a\} \{c\}, \{c\} \{a\}, \{a\} \{b\}\},$$

we obtain

$$\text{steps}(\Sigma_S) = \{\lambda, \{a\}, \{c\}, \{a, c\}, \{c\} \{a\}, \{a\} \{b\}\}.$$

Note that $\{a\} \{c\} \notin \text{steps}(\Sigma_S)$ since after executing $a$, $b$ becomes enabled and $c$ cannot be executed. It is not difficult to see that $\text{steps}(\Sigma_S)$ cannot be consistent with any semantical model based on causal partial orders. If it were, then the simultaneous occurrence of $a$ and $c$ in $\{a,c\}$ would imply that $\{a\} \{c\}$ could also be executed.

\begin{figure}[h]
    \centering
    \includegraphics[width=0.5\textwidth]{figure2_1.png}
    \caption{Figure 2.1}
\end{figure}
We now face the problem whether \( \{a,c\} \) should at all be allowed as a valid behaviour of \( \Sigma_S \). In [BK] it was argued that the answer is intrinsically related to whether or not one can regard \( a \) as an event taking some time: If \( a \) is instantaneous (i.e. takes no time) then \( \{a,c\} \) should not be allowed. (The resulting model was discussed in [BK].) If, however, \( a \) cannot be regarded as instantaneous (possibly because \( a \) is itself a compound event) then \( \{a,c\} \) should be allowed, and one should look for an invariant model more expressive than those based on causal partial orders to capture the behaviour of \( \Sigma_S \) (see [JK]). When dealing with dynamic priority systems, we also have to address the above problem (first discussed in [Ja]). We here will adopt the same position as [BK] and altogether exclude step sequences such as \( \{a,c\} \).

The discussion of a dynamic priority system \( \Sigma_{PR} = (\Sigma, \pi_{PR}) \) will proceed along the following lines: First, we derive \( \text{steps}(\Sigma_{PR}) \), the step sequences of the base system which do not violate \( \pi_{PR} \). Then we delete step sequences like \( \{a,c\} \) above and obtain \( \text{ker}(\Sigma_{PR}) \subseteq \text{steps}(\Sigma_{PR}) \) which we regard as the step sequence semantics of \( \Sigma_{PR} \). Finally, we construct an ordinary Petri net \( \Pi_{PR} \) generating \( \text{ker}(\Sigma_{PR}) \). Note that the existence of such a net is not a priori guaranteed. That is, we will adhere to the following line of development:

\[
\Sigma_{PR} \rightarrow \text{steps}(\Sigma_{PR}) \rightarrow \text{ker}(\Sigma_{PR}) \rightarrow \Pi_{PR}.
\]

### 2.2 Basic Definitions

The class of Petri nets we use are finite P/T-nets with weighted arcs [Pe]. Although the base systems will always be safe, the net \( \Pi_{PR} \) will in general be non-safe and have non-unitary arcs. (This cannot be avoided even for static priority systems [BK].) The definitions of the standard notions concerning Petri nets can be found in the Appendix. Others are introduced below.

Our generic system will be a net \( \Sigma = (S,T,W,M_0) \), where \( S \) are places, \( T \) are transitions, \( W \) is the arc weight function, and \( M_0 \) is the initial marking. As the semantical model we use step sequences [SM,RV], which are strings \( \sigma = A_1...A_n \) of sets of transitions (steps) fired simultaneously. \( \Sigma \) generates a set of step sequences, denoted by \( \text{steps}(\Sigma) \). To identify step sequences like
\{a,c\} in Section 2.1, we need the notion of a refinement (or decomposition) of a step sequence.

**Definition 2.1**
A refinement of a step sequence \( \sigma = A_1...A_n \) is any step sequence

\[
\omega = A_{i1}...A_{ik1}...A_{i1k}...A_{ikn}
\]

such that for every \( i \), the sets \( A_{i1},...,A_{ik_i} \) form a partition of \( A_i \). We will denote this by \( \omega \in \text{ref}(\sigma) \). □

The step sequences of a Petri net are closed w.r.t. refinement.

**Proposition 2.2**
\( \text{steps}(\Sigma) = \text{ref}(\text{steps}(\Sigma)) \). □

Going from \( \text{steps}(\Sigma_{PR}) \) to \( \ker(\Sigma_{PR}) \) amounts to deleting those step sequences which destroy the above closure property.

**Definition 2.3**
Let \( G \) be a set of step sequences. Its refinement kernel, \( \ker(G) \), is the maximal subset which is closed w.r.t. refinement operation, i.e.

\[
\ker(G) = \max\{H \subseteq G \mid \text{ref}(H) = H\} = \{\sigma \in G \mid \text{ref}(\sigma) \subseteq G\}. \quad □
\]

We will restrict the class of base systems to the safe ones. Note that safeness defined below does not constrain in any way the behaviour of the net, e.g., as it is the case for C/E-systems [Re].

**Definition 2.4**
A place \( s \in S \) is 1-bounded if

\[
\forall t \in T: W(t,s) \leq 1 \land W(s,t) \leq 1 \\
\forall M \in [M_0]: M(s) \leq 1.
\]

\( \Sigma \) is safe if every place \( s \in S \) is 1-bounded. □

In the modelling of priority systems in [BK] a key role was played by the notion of a generalised place complement. A generalised complement for a set \( P \) of 1-bounded places is a new place which 'counts' the number of places in \( P \) which currently are unmarked, without affecting the step sequence semantics.
For every integer $n$, let $\delta(n) = n$ if $n \geq 0$; and $\delta(n) = 0$ otherwise.

**Definition 2.5 [BK]**

Let $P$ be a non-empty set of 1-bounded places of $\Sigma$. A generalised complement of $P$ is a new place $\gamma = \gamma(P)$ such that for every $t \in T$,

$$
\hat{W}(t, \gamma) = \delta(\sum_{p \in P} W(p, t) - W(t, p))
$$

$$
\hat{W}(\gamma, t) = \delta(\sum_{p \in P} W(t, p) - W(p, t)).
$$

For every marking $M$ of $\Sigma$, let

$$
\hat{M} : S \cup \{\gamma\} \to N
$$

be a mapping defined by $\hat{M}(s) = M(s)$ for all $s \in S$, and

$$
\hat{M}(\gamma) = |P| \cdot \sum_{p \in P} M(p).
$$

After adding $\gamma$ to $\Sigma$ we obtain a new system

$$
\Sigma_\gamma = (S \cup \{\gamma\}, T, W \cup \hat{W}, \hat{M}_0).
$$

The counting property of $\gamma$ is expressed as follows.

**Proposition 2.6**

$$
K(\gamma) = |P| \cdot \sum_{p \in P} K(p) \text{ for all markings } K \in (\hat{M}_0) .
$$

The reachable markings of $\Sigma$ and $\Sigma_\gamma$ are in bijection through the operation with the corresponding markings enabling the same steps. Thus, as far as the step sequence semantics is concerned, $\Sigma$ and $\Sigma_\gamma$ are equivalent systems.

**Theorem 2.7 [BK]**

1. $\hat{M}_0 = \{\hat{M} | M \in (M_0)\}$.
2. $M \in (M_0)$ $\Rightarrow$ enabled$_\Sigma(M) = enabled_{\Sigma_\gamma}(\hat{M})$.
3. steps($\Sigma$) = steps($\Sigma_\gamma$).  

Fig. 2.2 shows an example of generalised place complement.
3 State-Controlled Priority Systems

Let $\Sigma=(S,T,W,M_o)$ be a safe system. A state-controlled priority system based on $\Sigma$,

$\Sigma_{SC}=(\Sigma,n_{SC})$

is characterised by a ternary relation

$n_{SC}\subseteq T\times T \times 2^S$.

Each $(a,b,Q)\in n_{SC}$ defines one local state $Q$ in which a priority constraint $a\prec b$ holds.

We first single out those step sequences of the base system $\Sigma$ which do not violate the priority specification.

Definition 3.1

Let $\text{steps}(\Sigma_{SC})$ be the maximal prefix-closed subset of $\text{steps}(\Sigma)$ such that if $oA \in \text{steps}(\Sigma_{SC})$ and $M_o(a)M$ then

$\forall (a,b,Q) \in n_{SC}: M(Q \cup \bullet b) = \{1\} \Rightarrow a \notin A$. □
We then define
\[ \text{ker}(\Sigma_{SC}) = \text{ker}(\text{steps}(\Sigma_{SC})). \]

It is possible to construct a priority-free system \( \Pi_{SC} \) such that
\[ \text{steps}(\Pi_{SC}) = \text{ker}(\Sigma_{SC}). \]

To illustrate its construction, we consider \( \Sigma_{SC} = (\Sigma, \pi_{SC}) \), where \( \Sigma \) is shown in Fig. 3.1(a) and \( \pi_{SC} = \{(a, b, \{p\})\} \). I.e. \( a < b \) is the only priority constraint which holds if \( p \) is marked. The construction of \( \Pi_{SC} \) is carried out in two steps:

Step 1: We construct a generalised complement place \( \gamma(P) \) for the set \( P = \{p\} \cup \bullet b \). This does not change the step sequence semantics of the system (Theorem 2.7(3)), yet enables us to find out whether both \( p \) and \( \bullet b \) are marked (Proposition 2.6). The resulting system is shown in Fig. 3.1(b).

Step 2: We add a loop (in fact, increment by one the arcs’ weights) between \( a \) and \( \gamma(P) \) to make \( a \) depend on the presence of a token in \( \gamma(P) \). As a result, \( a \) can only occur if \( b \) is not enabled or \( p \) is not

![Diagram](image-url)

**Figure 3.1**
marked (Proposition 2.6). The resulting system $\Pi_{SC}$ is shown in Fig. 3.1(c).

$\Pi_{SC}$ is defined in the following way.

**Definition 3.2**

Let $n_{SC} = \{(a_1, b_1, Q_1), \ldots, (a_m, b_m, Q_m)\}$, and $P_i = Q_i \cup b_i$ for $i = 1, \ldots, m$.

Let $\Sigma_1 = (S_1, T, W_1, M_1) = \Sigma_\gamma(P_1) \cdots \gamma(P_m)$.

Note: We distinguish $\gamma(P_i)$ from $\gamma(P_j)$ for $i \neq j$, even if $P_i = P_j$.

I.e., $\Sigma_1$ contains exactly $m$ new places $\gamma_i = \gamma(P_i)$.

Define $\Pi_{SC} = (S_1, T, W_2, M_1)$, where $W_2$ is the same as $W_1$ with one exception: If $i \leq m$ and $W_1(\gamma_i, a_i) = 0$, then

$W_2(\gamma_i, a_i) = 1$

$W_2(a_i, \gamma_i) = W_1(a_i, \gamma_i) + 1$. □

**Theorem 3.3**

$\text{steps}(\Pi_{SC}) = \ker(\Sigma_{SC})$.

**Proof**

We prove slightly stronger result (we need it in the proof of Theorem 5.4) by assuming that all places in $P_i = Q_i \cup b_i$ ($i = 1, \ldots, m$) are 1-bounded, and removing the assumption that $\Sigma$ be safe.

Let $\Psi_{SC} = (\Sigma_1, n_{SC})$.

From Theorem 2.7 it follows that $\ker(\Sigma_{SC}) = \ker(\Psi_{SC})$. Hence it suffices to prove that $\text{steps}(\Pi_{SC}) = \ker(\Psi_{SC})$.

We will first show $\text{steps}(\Pi_{SC}) \subseteq \ker(\Psi_{SC})$. In fact, we will prove $\text{steps}(\Pi_{SC}) \subseteq \text{steps}(\Psi_{SC})$ which suffices to show the required inclusion since $\text{ref}(\text{steps}(\Pi_{SC})) = \text{steps}(\Pi_{SC})$. The proof proceeds by induction on the length of the step sequence.

Clearly, $\lambda \in \text{steps}(\Psi_{SC})$. Suppose that $\sigma \in \text{steps}(\Pi_{SC}) \cap \text{steps}(\Psi_{SC})$ and $\sigma A \in \text{steps}(\Pi_{SC})$. Let $M_1(\sigma) M$ in $\Pi_{SC}$. By Definition 3.2 (the last two lines), we also have $M_1(\sigma) M$ in $\Sigma_1$.

From $A \in \text{enabled}_{\Pi_{SC}}(M)$ it follows that $A \in \text{enabled}_{\Sigma_1}(M)$ (the weights of arcs in $\Sigma_1$ do not exceed those in $\Pi_{SC}$). Thus, if $\sigma A \in \text{steps}(\Psi_{SC})$ then, by Definition 3.1, there is $i \leq m$ such that $a_i \in A$ and $M(P_i) = \{1\}$. From
$M(P_i) = \{1\}$ and Proposition 2.6 it follows that $M(\gamma_i) = 0$, which contradicts $a_i \in A \in \text{enabled}_{\Pi_{SC}}(M)$ and $W(\gamma_i, a_i) > 0$ (follows from Definition 3.2). Hence $\sigma A \in \text{steps}(\Psi_{SC})$.

Next we prove $\ker(\Psi_{SC}) \subseteq \text{steps}(\Pi_{SC})$, again by induction on the length of the step sequence.

Clearly, $A \in \text{steps}(\Pi_{SC})$. Suppose that $a \in \text{steps}(\Pi_{SC}) \cap \ker(\Psi_{SC})$ and $\sigma A \in \ker(\Psi_{SC})$. Let $M$ be a marking such that $M_1(\sigma)M$ in both $\Pi_{SC}$ and $\Sigma_1$.

If $A \in \text{enabled}_{\Pi_{SC}}(M)$ then there is $i \leq m$ such that the following hold.
(Below $\gamma = \gamma_i$, $a = a_i$ and $P = P_i$.)

1. $M(\gamma) < \sum_{t \in A} W_2(\gamma, t)$.
2. $M(\gamma) \geq \sum_{t \in A} W_1(\gamma, t)$.

Hence, by Definition 3.2,

3. $a \in A$ and $W_1(\gamma, a) = 0$ and $W_2(\gamma, a) = 1$.

Thus, by $\sigma A \in \ker(\Psi_{SC})$ and Definition 3.1,

4. $M(P) = \{1\}$.

Furthermore, from (3) and Definition 3.2 it follows that

5. $\sum_{t \in A} W_2(\gamma, t) = 1 = \sum_{t \in A} W_1(\gamma, t)$.

Let $C = \{t \in A \mid W_1(\gamma, t) > 0\}$ and $D = \{t \in A \mid W_1(\gamma, t) = 0\}$. From (3) it follows that $D \neq \emptyset$. Suppose that $C = \emptyset$. Then, by (1,2,5), $M(\gamma) = 0$. Hence, by Proposition 2.6, $M(P) = \{1\}$ which produces a contradiction with (4).

Thus $C \neq \emptyset$.

From $\sigma A \in \ker(\Psi_{SC})$ it follows that $\sigma CD \in \ker(\Psi_{SC})$. Let $M(C)K(D)L$ in $\Sigma_1$. From $a \in D$ it follows that $K(P) = \{1\}$. Hence, by Proposition 2.6, $K(\gamma) \geq 1$. Consequently, we have

$$1 \leq K(\gamma) = M(\gamma) + \sum_{t \in C} W_1(t, \gamma) - W_1(\gamma, t).$$

We also observe that (by Definition 2.5) for all $t \in A$,

$$W_1(t, \gamma) > 0 \Rightarrow W_1(\gamma, t) = 0 \Rightarrow t \notin C.$$  

Thus

$$M(\gamma) \geq 1 + \sum_{t \in C} W_1(\gamma, t) = 1 + \sum_{t \in A} W_1(\gamma, t) = (5) \sum_{t \in A} W_2(\gamma, t)$$
which produces a contradiction with (1). □

4 Event-Controlled Priority Systems

Let $\Sigma = (S,T,W,M_0)$ be a safe system. An event-controlled priority system based on $\Sigma$,

$$\Sigma_{EC} = (\Sigma, \pi_{EC})$$

has as its priority specification a triple

$$\pi_{EC} = (\rho_0, \text{On}, \text{Off})$$

such that $\rho_0 \subseteq T \times T$ is an initial priority specification and

$$\text{On}, \text{Off} : T \times T \to 2^T$$

are two mappings such that $\text{On}(a,b)$ and $\text{Off}(a,b)$ are disjoint, for all $a, b \in T$.

As before, we first single out those step sequences of $\Sigma$ which do not violate the priority specification.

**Definition 4.1**

Let $\rho : \text{steps}(\Sigma) \to 2^T \times T$ be defined by $\rho(\lambda) = \rho_0$ and

$$\rho(\sigma A) = \rho(\sigma) - \text{Off}^{-1}(A) \cup \text{On}^{-1}(A).$$

Let $\text{steps}(\Sigma_{EC})$ be the maximal prefix-closed subset of $\text{steps}(\Sigma)$ such that if $\sigma A \in \text{steps}(\Sigma_{EC})$ then

$$(a,b) \in \rho(\sigma) \land a \in A \Rightarrow \sigma \{b\} \not\in \text{steps}(\Sigma)$$

$$(a,b) \in \rho(\sigma) \Rightarrow \text{On}(a,b) \cap A = \emptyset$$

$$(a,b) \not\in \rho(\sigma) \Rightarrow \text{Off}(a,b) \cap A = \emptyset.$$ □

$\rho$ is a mapping which for every step sequence gives the current priority relation based on the initial priority relation and the transitions executed so far. The last two conditions mean that no transition in $\text{On}(a,b)$ can be executed if $a < b$ holds; similarly, no transition in $\text{Off}(a,b)$ can be executed if $a < b$ is suspended. We will discuss this restriction in the next section.

We define

$$\ker(\Sigma_{EC}) = \ker(\text{steps}(\Sigma_{EC})).$$
As in the case of $\Sigma_{SC}$, it is possible to find a priority-free system $\Pi_{EC}$ such that

$$\ker(\Sigma_{EC}) = \text{steps}(\Pi_{EC}).$$

The construction of $\Pi_{EC}$ is based on the following result which follows directly from the definition of $\ker(\Sigma_{EC})$ and Definition 4.1.

**Proposition 4.2**

Let $\sigma A \in \ker(\Sigma_{EC})$ and $a,b \in T$.

(1) $|\text{Off}(a,b) \cap A| \leq 1$.

(2) $|\text{On}(a,b) \cap A| \leq 1$. 

The problem of constructing $\Pi_{EC}$ can be reduced to that discussed in the previous section. Indeed, from Definition 4.1 and Proposition 4.2 it follows that for each priority constraint $a < b$ the occurrences of the transitions of $\text{On}(a,b)$ and $\text{Off}(a,b)$ in the step sequences of $\ker(\Sigma_{EC})$ are totally sequentialised and strictly alternating. Thus we can augment $\Sigma$ with two new places, $\text{on}_{ab}$ and $\text{off}_{ab}$, and connect them in the way shown in Fig. 4.1. If $(a,b) \in \rho_0$ then we mark $\text{on}_{ab}$ with one token and leave $\text{off}_{ab}$ unmarked; otherwise $\text{off}_{ab}$ is initially marked and $\text{on}_{ab}$ unmarked. As a result, the priority constraint $a < b$ holds if and only if $\text{on}_{ab}$ is marked. This, however, can be captured by setting a state-based priority specification

$$(a,b,\{\text{on}_{ab}\}) \in \Pi_{SC}.$$ 

![Diagram](image_url)

**Figure 4.1**

On($a,b$) = $\{t_1, \ldots, t_m\}$  \hspace{1cm} \text{off}_{ab} \hspace{1cm} \text{Off}(a,b) = \{u_1, \ldots, u_n\}$
Definition 4.3
Define $\Sigma_1 = (S_1, T, W_1, M_1)$ to be a safe net, where

1. $S_1 = S \cup \{\text{on}_{ab}, \text{off}_{ab} \mid a, b \in T\}$.

2. $M_1$ restricted to $S$ is the same as $M_o$. Moreover, for all $a, b \in T$,

   $(a, b) \notin \rho_o \Rightarrow M_1(\text{on}_{ab}) = 1 \wedge M_1(\text{off}_{ab}) = 0$

   $(a, b) \notin \rho_o \Rightarrow M_1(\text{on}_{ab}) = 0 \wedge M_1(\text{off}_{ab}) = 1$.

3. $W_1$ restricted to $S \times T \cup T \times S$ is the same as $W$. Moreover, for all $a, b \in T$,

   $\bullet \text{on}_{ab} = \text{off}_{ab} = \text{On}(a, b)$

   $\bullet \text{off}_{ab} = \text{on}_{ab} = \text{Off}(a, b)$.

Let $\Sigma_{SC} = (\Sigma_1, n_{SC})$ be a state-controlled priority system such that

$n_{SC} = \{(a, b, \{\text{on}_{ab}\}) \mid a, b \in T\}$.

Let $\Pi_{EC} = \Pi_{SC}$, where $\Pi_{SC}$ is defined for $\Sigma_{SC}$ according to Definition 3.2. \qed

Theorem 4.4
steps($\Pi_{EC}$) = ker($\Sigma_{EC}$).

Proof
We first show that

1. $\sigma \in \ker(\Sigma_{SC}) \cap \ker(\Sigma_{EC}) \wedge \sigma A \in \ker(\Sigma_{EC}) \Rightarrow \sigma A \in \text{steps}(\Sigma_{SC})$.

Let $M_o[\sigma]K$ in $\Sigma$ and $M_1[\sigma]L$ in $\Sigma_1$. By Definition 4.1 and 4.3, for all $a, b \in T$,

2. $L(\text{on}_{ab}) + L(\text{off}_{ab}) = 1$

3. $(a, b) \in \rho(\sigma) \Rightarrow L(\text{on}_{ab}) = 1$.

From Definition 4.1 and Proposition 4.2 it follows that

$(a, b) \in \rho(\sigma) \Rightarrow \text{On}(a, b) \cap A = \emptyset \wedge |\text{Off}(a, b) \cap A| \leq 1$

$(a, b) \notin \rho(\sigma) \Rightarrow \text{Off}(a, b) \cap A = \emptyset \wedge |\text{On}(a, b) \cap A| \leq 1$.

This and (2, 3) means that $\sigma A \in \text{steps}(\Sigma_1)$. If $\sigma A \notin \text{steps}(\Sigma_{SC})$ then there must be $a, b \in T$ such that $\sigma A$ and $L(\bullet b \cup \{\text{on}_{ab}\}) = \{1\}$. Thus, by (3),

$(a, b) \in \rho(\sigma)$. This, $\sigma A$ and $K(\bullet b) = L(\bullet b) = \{1\}$ produces a contradiction with Definition 4.1. Hence (1) holds.
We further observe that $\text{enabled}_{\Sigma_1}(L) \subseteq \text{enabled}_{\Sigma}(K)$ and (2,3) imply the following:

$$\sigma \in \ker(\Sigma_{SC}) \cap \ker(\Sigma_{EC}) \land \sigma A \in \ker(\Sigma_{SC}) \Rightarrow \sigma A \in \text{steps}(\Sigma_{EC}).$$

By (1,4) and $\lambda \in \ker(\Sigma_{SC}) \cap \ker(\Sigma_{EC})$, we obtain $\ker(\Sigma_{SC}) = \ker(\Sigma_{EC})$. Furthermore, by Theorem 3.3, $\text{steps}(\Pi_{EC}) = \text{steps}(\Pi_{SC}) = \ker(\Sigma_{SC})$ which completes the proof. □

5 Extended Event-Controlled Priority Systems

In the event-controlled priority system $\Sigma_{EC}$ no transition in $\text{On}(a,b)$ could be executed when $a < b$ was active. We now want to relax this restriction. More precisely, in addition to $\text{On}(a,b)$ and $\text{Off}(a,b)$ we want to introduce two new sets of transitions, $\text{ON}(a,b)$ and $\text{OFF}(a,b)$, such that the enabling of $\sigma \in \text{ON}(a,b) \cup \text{OFF}(a,b)$ does not depend on whether $a < b$ is currently active or not. The resulting mechanism of controlling priorities cannot be modelled within the framework of unlabelled nets. To show this we consider a priority system $\Sigma_{PR}$ based on the net of Fig. 5.1 with $\rho_o = \emptyset$ and $\text{ON}(a,b) = \{c\}$.

Let $\sigma = \{a\}$ and $\sigma_i = \{c\}^i$ for $i = 1,2,\ldots$. We would expect $\Sigma_{PR}$ to generate step sequences

$$\sigma, \sigma_1, \sigma_2, \sigma_3, \ldots$$

while $\omega = \{c\}{\{a\}}$ should not be allowed.

![Figure 5.1](image)
Suppose that $\Sigma = (S,T,W,M_0)$ is a system such that

$$\sigma_0, \sigma_1, \sigma_2, \sigma_3, \ldots \in \text{steps}(\Sigma) \text{ and } \omega \notin \text{steps}(\Sigma).$$

From $\sigma_1, \sigma_2, \sigma_3, \ldots \in \text{steps}(\Sigma)$ it follows that for all $s \in S$, $W(c,s) - W(s,c) \geq 0$. This, however, contradicts $\sigma \in \text{steps}(\Sigma)$ and $\omega \notin \text{steps}(\Sigma)$.

As before, let $\Sigma = (S,T,W,M_0)$ be a safe system. An extended event-controlled priority system based on $\Sigma$,

$$\Sigma_{\text{EEC}} = (\Sigma, n_{\text{EEC}})$$

is characterised by

$$n_{\text{EEC}} = (\rho_0, \text{On}, \text{Off}, \text{ON}, \text{OFF})$$

generalising $n_{\text{EC}}$ in the following way:

$$\text{ON}, \text{OFF} : T \times T \rightarrow 2T$$

$\text{On}(a,b)$, $\text{Off}(a,b)$, $\text{ON}(a,b)$, $\text{OFF}(a,b)$ are disjoint sets

for all $a, b \in T$. We will denote by $\text{ON}$ and $\text{OFF}$ two mappings defined by

$$\text{ON}(a,b) = \text{ON}(a,b) \cup \text{On}(a,b)$$

$$\text{OFF}(a,b) = \text{OFF}(a,b) \cup \text{Off}(a,b).$$

We first define $\text{steps}(\Sigma_{\text{EEC}})$.

**Definition 5.1**

Let $\rho : \text{steps}(\Sigma) \rightarrow 2T \times T$ be defined by $\rho(\lambda) = \rho_0$ and

$$\rho(\sigma A) = \rho(\sigma) \cdot \text{OFF}^{-1}(A) \cup \text{ON}^{-1}(A).$$

Let $\text{steps}(\Sigma_{\text{EEC}})$ be the maximal prefix-closed subset of $\text{steps}(\Sigma)$ such that if $\sigma A \in \text{steps}(\Sigma_{\text{EEC}})$ then

$$(a,b) \notin \rho(\sigma) \land a \notin A \Rightarrow \sigma(b) \notin \text{steps}(\Sigma)$$

$$(a,b) \in \rho(\sigma) \Rightarrow \text{On}(a,b) \cap A = \emptyset \land |\text{OFF}(a,b) \cap A| \leq 1$$

$$(a,b) \notin \rho(\sigma) \Rightarrow \text{Off}(a,b) \cap A = \emptyset \land |\text{ON}(a,b) \cap A| \leq 1$$

$$\text{ON}(a,b) \cap A = \emptyset \text{ or } \text{OFF}(a,b) \cap A = \emptyset.$$

$\rho$ is again a mapping which gives the current priority relation. We assumed that if a step $A$ changes the status of a priority constraint then there is a unique transition in $A$ which effects that. The last condition has
been added to exclude a simultaneous occurrence of two transitions, one activating, the other suspending, the same priority constraint.

We then define

$$\ker(\Sigma_{\text{EEC}}) = \ker(\text{steps}(\Sigma_{\text{EEC}})).$$

As before, our aim is to construct a system which would generate the step sequences in $\ker(\Sigma_{\text{EEC}})$. As we have already seen, this cannot in general be achieved using unlabelled nets. We will have to set ourselves slightly less ambitious goal. Namely, we will construct a system $\Pi_{\text{EEC}}$ and define a relabelling function on transitions, $f$, such that

$$f(\text{steps}(\Pi_{\text{EEC}})) = \ker(\Sigma_{\text{EEC}}).$$

The basic idea behind the construction of $\Pi_{\text{EEC}}$ is to replace each $c \in \text{ON}(a,b)$ by two transitions labelled $c$, and to introduce four new places, $\text{on}_{ab}$, $\text{off}_{ab}$, $\text{ON}_{ab}$ and $\text{OFF}_{ab}$, as shown in Fig. 5.2. The rest of the construction is essentially the same as in Section 4, however some care needs to be taken to make the application of Theorem 3.3 possible.

There are two points which should be mentioned. We need places $\text{ON}_{ab}$ and $\text{OFF}_{ab}$, holding either 0 or $|T|$ tokens, to avoid unnecessary sequentialisation of transitions in $\text{ON}(a,b)$ and $\text{OFF}(a,b)$ in cases when they do not change the status of the priority constraint $a < b$. Also, we may need more than two copies of $c$ since it may belong to several different ON and OFF sets. The definition of $\Pi_{\text{EEC}}$ comes in two parts. We first formalise the construction illustrated in Fig. 5.2.

![Figure 5.2]

Figure 5.2
Definition 5.2

Let $\Sigma_1 = (S_1, T_1, W_1, M_1)$ be a system such that

1. $S_1 = S \cup \{ \text{on}_x, \text{off}_x, \text{ON}_x, \text{OFF}_x \mid x \in T \times T \}$.

2. $T_1 = \{ t_F \mid t \in T \land F \subseteq \{ x \mid t \in \text{ON}(x) \cup \text{OFF}(x) \} \}$.  
   (Note: $(a,b) \in F$ means that $t_F$ always changes the status of the priority constraint $a < b$, as the right $c$ in Fig. 5.2.)

3. For all $t_F$ and $s \in S$, $W_1(t_F,s) = W(t,s)$ and $W_1(s,t_F) = W(s,t)$.

4. For all $t_F$ and $s \in S_1 \setminus S$, $W_1(t_F,s) = W_1(s,t_F) = 0$ unless one of the following holds:
   (a) $t \in \text{ON}(x)$ or $t \notin \text{ON}(x) \land x \notin F$. Then
       \[ W_1(t_F, \text{on}_x) = W_1(\text{off}_x, t_F) = 1 \]
       \[ W_1(t_F, \text{ON}_x) = W_1(\text{OFF}_x, t_F) = |T|. \]
   (b) $t \notin \text{ON}(x) \land x \notin F$. Then
       \[ W_1(t_F, \text{ON}_x) = W_1(\text{ON}_x, t_F) = 1. \]
   (c,d) Similar to (a) and (b) with the roles of $\text{on}$'s and $\text{off}$'s interchanged.

5. For $s \in S$, $M_1(s) = M_0(s)$, while for $s \in S_1 \setminus S$, if $x \in \rho_0$ then
   \[ M_1(\text{on}_x) = 1 \land M_1(\text{ON}_x) = |T| \]
   \[ M_1(\text{off}_x) = M_1(\text{OFF}_x) = 0; \]
   otherwise
   \[ M_1(\text{on}_x) = M_1(\text{ON}_x) = 0 \]
   \[ M_1(\text{off}_x) = 1 \land M_1(\text{OFF}_x) = |T|. \]

We now transform $\Sigma_1$ with the priority specification $\pi_{\text{EEC}}$ into a state-controlled priority system $\Sigma_{\text{SC}}$. To be able to apply Theorem 3.3 we need to ensure that all places involved in the complement construction described in Definition 3.2 are 1-bounded. To achieve this we resort to a little trick.
Definition 5.3

Let $\Sigma_2$ be the union of $\Sigma_1$ of Definition 5.2 and the following net:

\[
\begin{array}{c}
t_0 \\
\end{array}
\begin{array}{c}
\Rightarrow\\\nonumber
\end{array}
\begin{array}{c}
s_0 \\
\end{array}
\]

Let $\Sigma_{SC} = (\Sigma_2, \pi_{SC})$ be a state-controlled priority system such that

\[
\pi_{SC} = \{(t_0, t_0, \emptyset)\} \cup \{(t_F, t_0, w) \cup \{on_{tw}\} | t_F \in T_1 \land w \in T\}.
\]

Let $\Pi_{EEC} = \Pi_{SC}$, where $\Pi_{SC}$ is defined for $\Sigma_{SC}$ according to Definition 3.2. $\square$

t_0$ is a dummy transition which is never executed in $\Sigma_{SC}$ although it is always enabled in $\Sigma_2$. The original priority constraints are transformed into static priority constraints involving $t_0$. $(t_F, t_0, w) \cup \{on_{tw}\} \in \pi_{SC}$ means that $t < w$ will be properly reflected in $\Sigma_{SC}$; since $t_0$ is always enabled in $\Sigma_2$, whenever $w \cup \{on_{tw}\}$ are all marked, $t_F$ will be disabled.

Theorem 5.4

\[f(\text{steps}(\Pi_{EEC})) = \ker(\Sigma_{EEC}), \text{ where } f(t_F) = t, \text{ for all } t_F, \text{ and } f(t_0) = t_0.\]

Note: $f(A_1 \ldots A_k) = f(A_1) \ldots f(A_k)$ for every step sequence $A_1 \ldots A_k$.

Proof

(Below we use notation from Definition 5.1, 5.2 and 5.3)

From Theorem 3.3 (in its slightly strengthened version) it follows that

\[\text{steps}(\Pi_{EEC}) = \ker(\Sigma_{SC}).\]

Thus to show the result it suffices to prove that

\[f(\ker(\Sigma_{SC})) = \ker(\Sigma_{EEC}).\]

This in turn follows directly from $\lambda \in f(\ker(\Sigma_{SC})) \cap \ker(\Sigma_{EEC})$ and the following two Facts.

**FACT 1:**

If $\sigma A \in \text{steps}(\Sigma_2)$ then for all $t, w \in A$

\[f(t) = f(w) \Rightarrow t = w.\]

proof of the fact: Follows from the safeness $\Sigma$ of and Definition 5.2(3).
FACT 2:

Let $\sigma \in \ker(\Sigma_{SC})$ and $f(\sigma) \in \ker(\Sigma_{EEC})$.

1. $f(\sigma)A \in \ker(\Sigma_{EEC}) \Rightarrow \exists C \in \text{steps}(\Sigma_{SC}). f(C) = A.$
2. $\sigma C \in \ker(\Sigma_{SC}) \Rightarrow f(\sigma C) \in \text{steps}(\Sigma_{EEC}).$

proof of the fact:

Let $M_0[f(\sigma)]K$ in $\Sigma$ and $M_1[\sigma]L$ in $\Sigma_2$. From Definition 5.2 and 5.3 it follows that for all $x \in T \times T$,

(F1) $L(\text{on}_x) = 1 \land L(\text{ON}_x) = |T| \land L(\text{off}_x) = L(\text{OFF}_x) = 0$ or
$L(\text{off}_x) = 1 \land L(\text{OFF}_x) = |T| \land L(\text{on}_x) = L(\text{ON}_x) = 0.$

(F2) $x \in \rho(f(\sigma)) \Leftrightarrow L(\text{on}_x) = 1.$

To show (1) we observe that from Definitions 5.2 and 5.3 and (F1) it follows that if $f(\sigma)\{t\} \in \text{steps}(\Sigma)$ then there is $t_F$ such that $\sigma\{t_F\} \in \text{steps}(\Sigma_2).$ Thus, from $f(\sigma)A \in \ker(\Sigma_{EEC}) \subseteq \text{steps}(\Sigma_{EEC})$, Definition 5.1, (F1) and the safeness of $\Sigma$, it follows that there is $C$ such that $\sigma C \in \text{steps}(\Sigma_2)$ and $f(C) = A$. Suppose that $\sigma C \notin \text{steps}(\Sigma_{SC}).$ Then is $t_F \in C$ and $w \in T$ such that $L(\bullet w \cup \{\text{on}_t\}) = \{1\}.$ Thus, by (F2), we have: $t \in A$, $(t, w) \in \rho(f(\sigma))$ and $K(\bullet w) = \{1\},$ producing a contradiction with Definition 5.1. Hence (1) holds.

To show (2) we first observe that, by safeness of $\Sigma$ and $(t_0, t_0, \emptyset) \in \Pi_{SC},$

$\sigma C \in \ker(\Sigma_{SC}) \Rightarrow \sigma C \in \text{steps}(\Sigma_1) \Rightarrow f(\sigma C) \in \text{steps}(\Sigma).$

Suppose there is $(t, w) \in \rho(f(\sigma))$ such that $t \in f(C)$ and $f(\sigma)\{w\} \in \text{steps}(\Sigma).$ Then there is $t_F \in C$ and, by (F.2), $L(\text{on}_t) = 1.$ Moreover, by $f(\sigma)\{w\} \in \text{steps}(\Sigma), L(\bullet w) = K(\bullet w) = \{1\}.$ Hence $L(\bullet w \cup \{\text{on}_t\}) = \{1\},$ contradicting

$\sigma C \in \text{steps}(\Sigma_{SC}) \land L(\bullet t) = \{1\} \land (t_F, t_0, \bullet w \cup \{\text{on}_t\}) \in \Pi_{SC}.$

Hence

$(t, w) \in \rho(f(\sigma)) \land t \in f(C) \Rightarrow f(\sigma)\{w\} \notin \text{steps}(\Sigma).$

The remaining three conditions in Definition 5.1 follow directly from (F1,F2). Hence $f(\sigma C) \in \text{steps}(\Sigma_{EEC})$ and (2) holds. □
Final Remarks

In [JL] dynamic priorities are simulated by using 'ghost' events and static priority COSY path programs. The ghost events might be rendered in our framework in the following way. Suppose that $\Sigma_{SC}=(\Sigma, \pi_{SC})$ is a state-controlled priority system such that $\pi_{SC} = [(a,b,Q)]$. One then constructs a ghost transition $gh$ such that $\bullet gh = \bullet \delta U Q$ and models $\Sigma_{SC}$ as a static priority system $\Sigma_S = (\Sigma', \rho)$, where $\rho = \{(gh,b), (a,gh)\}$ and $\Sigma'$ is $\Sigma$ after adding $gh$. Note that $gh$ is never enabled in $\Sigma_S$ yet it prevents $a$ from occurring if all places in $\bullet a U \bullet b U Q$ are marked. The way in which ghost events are used resembles quite closely our modelling of state-controlled priorities. As far as the event-controlled priority systems are concerned, we are not aware of any published paper on these. However, it seems that within a suitably chosen process algebra it should be possible to model event-controlled priorities, extending the treatment for statically defined priorities [Gr].

Acknowledgment

This work was supported by ESPRIT BRA 3148 Project DEMON. The author would like to thank the referees for their very helpful comments and suggestions for improvement.

References


Appendix

A net is a triple $N=(S,T,W)$ with $S \cap T = \emptyset$ and

$$W : S \times T \cup T \times S \rightarrow \mathbb{N}.$$ 

$S$ is the set of places, $T$ is the set of transitions, and $W$ is the arc weight function. We assume that both $S$ and $T$ are finite sets. For $x \in S \cup T$, the pre-set of $x$ is defined as

$$\bullet x = \{ y \in S \cup T \mid W(y,x) > 0 \}$$

and the post-set of $x$ is defined as

$$x^\bullet = \{ y \in S \cup T \mid W(x,y) > 0 \}.$$ 

We require that for all $t \in T$, $\bullet t = \emptyset$ and $t^\bullet = \emptyset$.

A marking of $N$ is defined as a function $M : S \rightarrow \mathbb{N}$. For $s \in S$, $M(s)$ denotes the number of tokens in $s$. A system

$$\Sigma = (S,T,W,M_o)$$ 

is a net $N$ with the initial marking $M_o$.

Let $M$ be a marking of $\Sigma$ and $A = \emptyset$ be a set of transitions of $T$. $A$ is concurrently enabled at $M$ if for all $s \in S$,

$$M(s) = \sum_{t \in A} W(s,t).$$ 

We denote this by

$$A \in \text{enabled}_\Sigma(M).$$ 

The marking $K$ produced from $M$ by the occurrence of $A$ is defined as

$$K(s) = M(s) + \sum_{t \in A} W(t,s) - W(s,t).$$
for all $s \in S$. We denote this by $M(A)K$.

A step sequence of $\Sigma$ is a sequence of sets of transitions $\sigma = A_1A_2...A_n$ ($n \geq 0$) such that there are markings $M_1, M_2,..., M_n$ satisfying $M_{i-1}(A_i)M_i$, for $i = 1,...,n$. We will denote this by $M_0(\sigma)M_n$.

The empty step sequence will be denoted by $\lambda$. The set of the step sequences of $\Sigma$ will be denoted by $\text{steps}(\Sigma)$. 

24