ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol. 35, No. 2

REFINEMENT OF PETRI-NET-BASED SYSTEM SPECIFICATION

King Sing Cheung

SCE, Hong Kong Baptist University
Kowloon Tong, Hong Kong

Abstract. For a discrete-event system specified as a labelled Petri net, locations of conditions are denoted as places with condition labels while locations of events are denoted as transitions with event labels. Very often, the same condition label or event label may appear in multiple locations in the system specification. Since every condition is finally implemented as a unique state and every event as a unique operation, in order for the system specification to become useful for implementation, all duplicate condition labels and event labels must be eliminated. In this paper, we propose an algorithm for this refinement through the fusion of common subnets. Our algorithm has three distinctive features. First, the units of fusing are subnets instead of individual places and transitions. Second, the groups of common subnets identified for fusing are maximal and disjoint so that the fusion needs to be done once. Third, the fusion preserves firing sequences so that the system behaviours will not be altered.

Keywords: Petri nets, Fusion, Firing Sequences, Refinement, System Specification.

1. Introduction

Petri nets and labelled Petri nets (or labelled nets) are often used for specifying discrete-event systems, because of their semantically rich syntactic constructs for representing sequential, concurrent, synchronous or asynchronous processes [1, 2, 3, 4]. There are also many well developed techniques for analysing the system properties such as liveness, boundedness and reversibility.

Typically, in early system design, a system is specified as a labelled net whose places and transitions are labelled for representing the locations of conditions and events, respectively. These places and transitions are not necessarily unique in the sense that duplicate condition labels or event labels may exist. This reflects the fact that the locations and conditions for executing the same operation may be different at different moments. Yet, every condition is eventually implemented as a unique system sub-state and every event as a unique operation. For implementation purposes, every condition or event needs to be uniquely represented in a system specification. Therefore, in order for the labelled net to be effectively used for implementation, it should be refined with all duplicate condition labels and event labels eliminated.

A straight-forward strategy for this elimination is to fuse each set of places having the same condition label into a single place, and each set of transitions having the same event label into a single transition. However, this does not work because the resulting net may exhibit firing sequences different from the original ones. In other words, the system behaviours may be distorted. Therefore, it is very important that the original firing sequences can be preserved after the fusion. In the literature, it has been studied on the composition of Petri nets by fusion of places and transitions [5, 6, 7]. Yet, among these composition methods, the purpose is not for eliminating duplicate labels, and the emphasis is placed on preservation of properties such as liveness and boundedness, but not firing sequences.

In this paper, we propose a method for eliminating duplicate labels from a labelled net while preserving the original firing sequences (event sequences). The elimination is made through the fusion of common subnets. In the rest of this paper, Section 2 describes labelled nets and their uses in system specification. Section 3 shows the elimination of duplicate labels by fusing common subnets. Then, in Section 4, a real-life example is presented for illustration. Section 5 concludes our results and highlights the distinctive features of our algorithm.

2. Labelled nets for system specification

In this section, we briefly introduce labelled Petri nets and show how they can be used for system specification.

Definition 2.1. A *labelled Petri net* (or *labelled net*) is a 7-tuple $N = \langle P, T, F, C, E, Lp, Lt \rangle$, where $\langle P, T, F \rangle$ is an ordinary PT-net, C is a set of *condition labels*, E is a set of *event labels*, $Lp : P \rightarrow C$ is a function for assigning a condition label to every place, and $Lt : T \rightarrow E$ is a function for assigning an event label to every transition.

Definition 2.2. Let $N = \langle P, T, F, C, E, Lp, Lt \rangle$ be a labelled net. A place p is said to be *uniquely labelled* in N if and only if $\forall p' \in P : (Lp(p') = Lp(p)) \Rightarrow (p' = p)$. A transition t is said to be *uniquely labelled* in N if and only if $\forall t' \in T : (Lt(t') = Lt(t)) \Rightarrow (t' = t)$. N is said to be *uniquely labelled* if and only if all places and transitions are uniquely labelled.

Figure 1 shows a labelled net. Places p_3, p_4, p_5, p_6, p_9 and p_{10} are uniquely labelled whereas places p_1, p_2, p_7 and p_8 are not. Condition label c_1 appears in p_1 and p_7, and c_2 in p_2 and p_8. Besides, transitions t_3, t_4 and t_5 are uniquely labelled whereas transitions t_1, t_2, t_6 and t_7 are not. Event label e_1 appears in t_1 and t_6, and e_2 in t_2 and t_7. The net is not uniquely labelled.

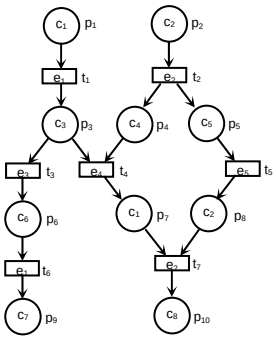

Figure 1. A labelled Petri net

A labelled net can be used for system specification. Basically, for a discrete event system, there are two essential elements, namely, events and conditions. An event may denote a signal, a trigger, a message or a user-triggered action. A condition may denote a system sub-state (state qualifying condition), a pre-condition or post-condition for an action or activity, such as the availability of resources.

For a system specified as a labelled net, the location where an event occurs is represented by a transition and the location of a condition by a place.

The semantic meaning of a condition or event is denoted by its label. For an event to occur, some conditions must be fulfilled in advance (pre-conditions) and some afterwards (post-conditions). These pre-conditions and post-conditions are represented as the pre-set and post-set of the transition. In the system specification, the same condition or event may appear at more than one location, reflecting the fact that the locations and conditions for executing the same event may be different at different moments. Hence, the labelled net is not necessarily unique.

3. Fusing common subnets of a labelled net

In this section, we propose an algorithm for fusing common subnets of a labelled net while preserving firing sequences (in terms of event sequences). The complete algorithm is shown in the Appendix. We first introduce the notions of patterns and common subnets for a labelled net, and then elaborate the fusion algorithm with examples.

3.1. Patterns and common subnets

Patterns and common subnets for a labelled net are defined as follows.

Definition 3.1 Let S be a uniquely labelled subnet of a labelled net N. The *pattern* of S in N, denoted as *Patt(N, S)*, is a condition-event net, with an identical structure and label allocation as S while ignoring the identities of the places and transitions of S.

Figure 2 shows a uniquely labelled subnet of a labelled net. Figure 3 shows its pattern.

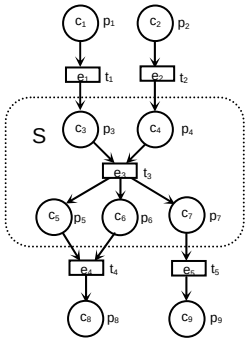

Figure 2. A uniquely labelled subnet S

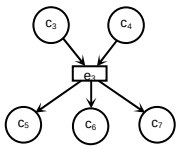

Figure 3. The pattern of subnet S in Figure 2

Definition 3.2. Let L_x and L_y be patterns of subnets in a labelled net. $L_x \cup L_y$ and $L_x \cap L_y$ denote the union and intersection of L_x and L_y, respectively. $L_x \setminus L_y$ denotes the displacement of L_x from L_y. L_x and L_y are said to be *disjoint* if and only if $L_x \cap L_y = \varnothing$.

Definition 3.3. For a labelled net N, a uniquely labelled subnet S is called a *common subnet* if and only if there exists at least one uniquely labelled subnet S' such that $S' \neq S$ and Patt(N, S') = Patt(N, S). Let S be a pattern of the common subnets in N. [N, L] = { S | Patt(N, S) = L } represents the *group of common subnets* having the same pattern L.

3.2. Identifying groups of common subnets

Step 1 of the algorithm is to identify the groups of common subnets for fusion. These groups need to be maximal and disjoint, as explained below.

In general, there are many ways for identifying common subnets. However, an arbitrary way may create problems. In particular, if more than one group of common subnets is identified and the groups are not mutually disjoint, the analysis for preservation of firing sequences becomes very difficult, especially when the same condition label or event label appear across different groups of common subnets. Moreover, after the fusion, the resulting net may still contain duplicate labels. Further fusions may be required.

Let us illustrate using a labelled net (N, M_0) in Figure 4. Two groups of common subnets are identified : [N, L_1] = { S_{11}, S_{12} } and [N, L_2] = { S_{21}, S_{22} }, where L_1 and L_2 are not mutually disjoint. S_{11} and S_{12} are then fused into S_1, and S_{21} and S_{22} into S_2, as shown in Figure 5. The firing sequences $\langle e_4, e_5 \rangle$ and $\langle e_1, e_2, e_4, e_6, e_7 \rangle$ are not preserved. Moreover, duplicate labels still exist, for example, condition label c_5 in p_{12} and p_{13}.

We propose that the groups of common subnets for fusing should be maximal and disjoint and cover all duplicate labels for two purposes.

First, the net so obtained after fusion will become uniquely labelled. Second, the number of groups of common subnets for fusion can be reduced to minimum as they are maximal. Figure 6 shows a labelled net N, where three maximal disjoint groups of common subnets (with patterns L_1, L_2 and L_3) are identified by Step 1 of the algorithm. They are [N, L_1] = { S_{11}, S_{12} }, [N, L_2] = { S_{21}, S_{22} } and [N, L_3] = { S_{31}, S_{32}, S_{33} }.

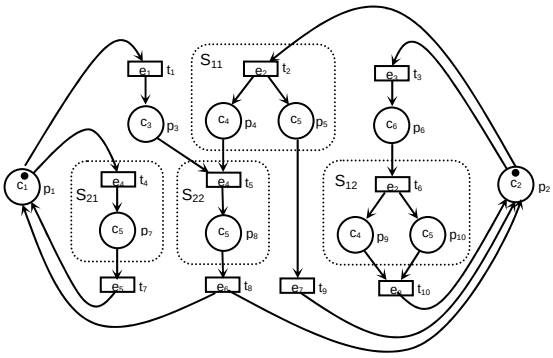

Figure 4. Two groups of common subnets with overlapping (non-disjoint) patterns

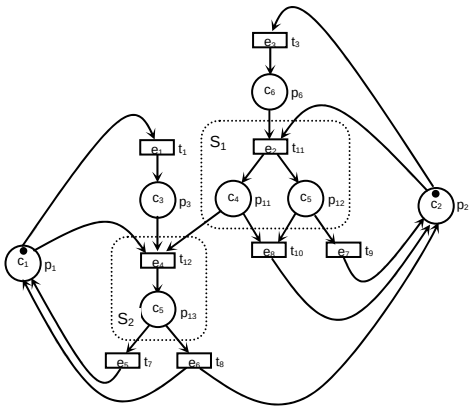

Figure 5. The resulting net (N', M_0') obtained from (N, M_0) in Figure 4 after the fusion

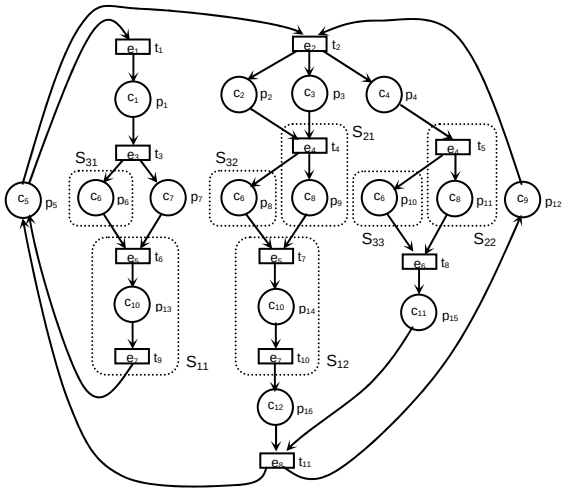

Figure 6. Maximal disjoint groups of common subnets for a labelled net

3.3. Fusing common subnets

After identifying groups of common subnets for fusing, the next step is to fuse the common subnets for each group. It should be noted that, without some special provisions, the fusion may alter the structure of the net and thus alter the firability conditions of transitions and the flow of tokens. As a result, the original firing sequences may not be preserved.

Figure 7 shows a labelled net where two common subnets S_1 and S_2 are to be fused into a single subnet S. Figure 8 shows the resulting net after the fusion. The fusion alters the pre-sets of t_{12} and t_{22} ($^\bullet t_{12}$ or $^\bullet t_{22}$ is different from $^\bullet t_2$), and p_{13} and p_{23} ($^\bullet p_{13}$ or $^\bullet p_{23}$ is different from $^\bullet p_3$). It also alters the post-sets of t_{14} and t_{24} ($t_{14}^\bullet$ or $t_{24}^\bullet$ is different from $t_4^\bullet$), and p_{14} and p_{24} ($p_{14}^\bullet$ and $p_{24}^\bullet$ are different from $p_4^\bullet$). The firability of transitions and flow of tokens may be altered. As a result, the original firing sequences (event sequences) are not preserved.

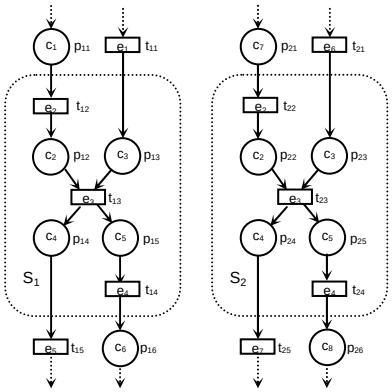

Figure 7. Two common subnets S_1 and S_2 to be fused

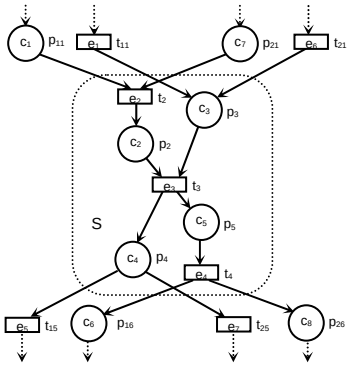

Figure 8. The fusion of subnets S_1 and S_2 in Figure 7

In order to maintain the firability of transitions and flow of tokens, some transformation must be done on the common subnets before fusion. Based on coloured Petri nets [8, 9], we propose to assign a unique colour for each common subnet (as colour labels of its ingoing arcs and outgoing arcs). A token flows into a common subnet is coloured according to the colour label of the ingoing arc. Its colour is reset as it flows out via the corresponding colour-labelled outgoing arc. Moreover, the common subnets have to be converted to a special format, which contains only places in its heads and tails, called PP-type.

Step 2.1 of the algorithm is the conversion of common subnets to PP-type while Step 2.2 is concerned with the assignment of colour labels to the common subnets.

Definition 3.4 For a subnet $S = \langle P', T', F' \rangle$ of a PT-net, $Pre(S) = (^\bullet P' \backslash T') \cup (^\bullet T' \backslash P')$ is called the *pre-set* of S, $Post(S) = (P'^\bullet \backslash T') \cup (T'^\bullet \backslash P')$ is called the *post-set* of S, $Head(S) = Pre(S)^\bullet \cap (P' \cup T')$ is called the *head* of S, and $Tail(S) = {}^\bullet Post(S) \cap (P' \cup T')$ is called the *tail* of S.

Definition 3.5 A subnet S of a PT-net $N = \langle P, T, F \rangle$ is said to be of *PP-type* if and only if $Head(S) \subseteq P$ and $Tail(S) \subseteq P$.

Consider two common subnets S_1 and S_2 of a labelled net in Figure 9. They are converted to PP-type and assigned with colour labels (S_1' and S_2'), as shown in Figure 10, and then fused into one single subnet S', as shown in Figure 11.

For S', p_{31} (respectively, p_{32}) has two separate coloured pre-sets which are corresponding to $^\bullet p_{12}$ and $^\bullet p_{22}$ (respectively, $^\bullet p_{13}$ and $^\bullet p_{23}$). Similarly, p_{34} (respectively, p_{35}) has two separate coloured post-sets which are corresponding to $p_{14}^\bullet$ and $p_{24}^\bullet$ (respectively, $p_{15}^\bullet$ and $p_{25}^\bullet$).

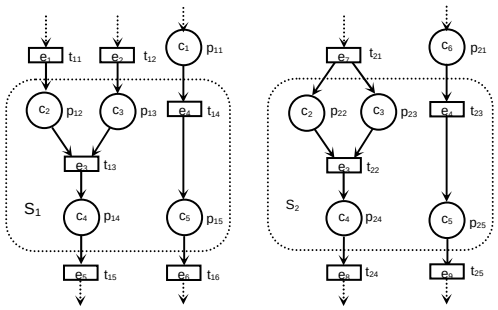

Figure 9. Two common subnets S_1 and S_2

Colour labels κ_1 and κ_2 are used for distinguishing the token flows between the two subnets S_1' and S_2'. It is noted that dummy places and dummy transitions are added and they do not alter firability conditions for transitions. The original firing sequences (event sequences) are preserved.

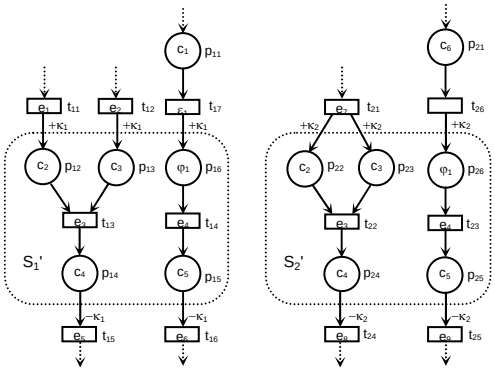

Figure 10. The converted common subnets S_1' and S_2'

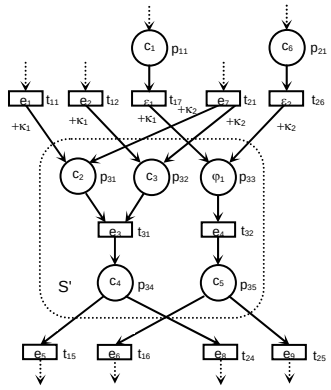

Figure 11. The fusion of subnets S_1' and S_2' in Figure 10

Figure 12 shows a labelled net with duplicate labels. Figure 13 shows the uniquely labelled net obtained by our fusion method, where firing sequences (event sequences) are preserved.

4. A Real-life example

In this section, a real-life example is presented for illustration. It is an Office Access Control System used in a high-tech company for controlling staff access to its 30+ offices. In the company, some offices can be accessed by all staff while others by authorised staff only and/or during specified time periods. Every office entrance is implemented with a card-reader, an emergency switch and an electronic lock, all controlled by a central server.

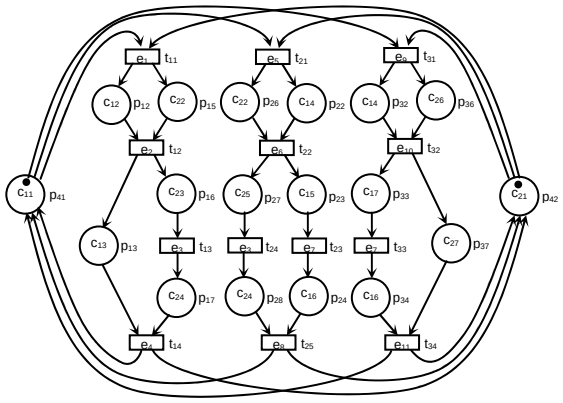

Figure 12. A labelled net with duplicate labels

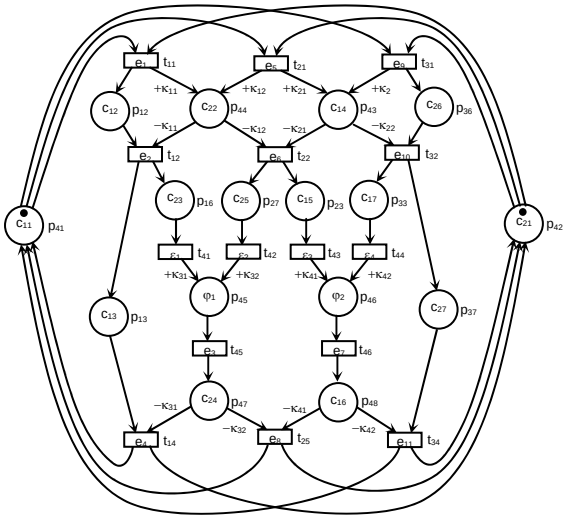

Figure 13. A uniquely labelled net obtained from the net in Figure 12

There are three possible use cases to gain access to an office.

- Successful access : A staff member presents his/her staff card via a card-reader. Access is granted. The door is unlocked for five seconds and then re-locked.

- Unsuccessful access : A staff member presents his/her staff card via a card-reader. Access is not granted. The door remains locked.

- Emergency access : A staff member presses the emergency key, and the door is unlocked immediately. After resetting by a security officer, the door is re-locked.

Figure 14 shows the system specification as a labelled net. The semantic meanings of the condition labels and event labels are shown in Tables 1 and 2, respectively.

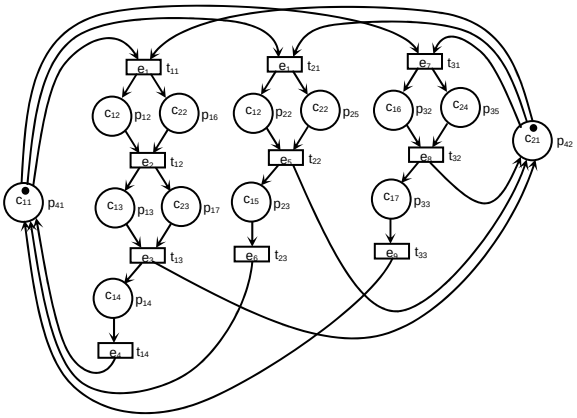

Figure 14. Specification of the Office Access System as a labelled net

Labels	Meanings
c_{11}	Server is ready
c_{12}	Server is processing access request
c_{13}	Server is waiting for re-lock
c_{14}	Server is writing log (successful access)
c_{15}	Server is writing log (unsuccessful access)
c_{16}	Server is waiting for emergency reset
c_{17}	Server is writing log (emergency access)
c_{21}	Door is locked
c_{22}	Door is waiting for response
c_{23}	Door is unlocked (successful access)
c_{24}	Door is unlocked (emergency access)

Table 1. Semantic meanings of condition labels for the labelled net in Figure 14

Labels	Meanings
e_1	Request for access is received
e_2	Access is granted
e_3	Time expires after access granted
e_4	Successful access is committed
e_5	Access is not granted
e_6	Unsuccessful access is committed
e_7	Request for emergency access is received
e_8	Door is reset to normal
e_9	Emergency access is committed

Table 2. Semantic meanings of event labels for the labelled net in Figure 14

The labelled net is not uniquely labelled. As for example, condition label c_{12} appears in places p_{12} and p_{22}, and event label e_1 appears in transitions t_{11} and t_{21}.

Figure 15 shows the uniquely labelled net obtained by applying the fusion algorithm. The original firing sequences (event sequences) are preserved after the fusion. In other words, the original system behaviour is preserved (not distorted).

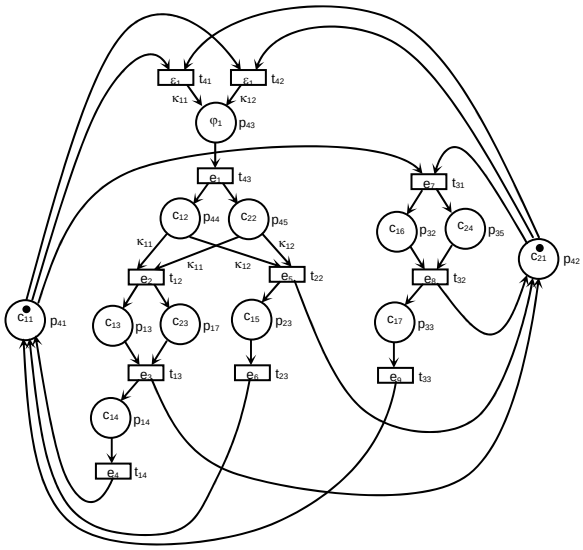

Figure 15. A uniquely labelled net obtained from the net in Figure 14

5. Conclusion

Petri nets provide semantically rich constructs for system specification, yet duplicate condition labels and event labels have to be resolved in order for the specification to become useful for implementation. In this paper, we propose a method for eliminating these duplicate labels by fusion of common subnets, where firing sequences are preserved. A complete algorithm is shown at the end of this paper. Our method has three distinctive features. First, the units of fusing are subnets, instead of individual places and transitions. Second, the groups of common subnets for fusing are maximal and disjoint so that the fusion need to be done once and for all. The net so obtained is uniquely labelled. Third, the common subnets to be fused are transformed to PP-type with colour labels so that firing sequences are preserved after fusion. As the firing sequences are preserved, the original system behaviours are not altered. A real-life example is presented for illustration. These contribute to the effective refinement of labelled-net-based system specification.

6. References

[1] **J.L. Peterson.** Petri Net Theory and the Modeling of System. *Prentice Hall*, 1981.

[2] **W. Reisig.** Petri Nets: An Introduction. *Springer-Verlag*, 1985.

[3] **T. Murata.** Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE, Vol.*77, *No.*4, 1989, 541-580.

[4] **J. Desel, W. Reisig.** Place Transition Petri Nets. *Lectures on Petri Nets* I : *Basic Models, Lecture Notes in Computer Science, Vol.*1491, *Springer-Verlag*, 1998, 122-173.

[5] **J. Esparza.** Synthesis Rules for Petri Nets and How They Lead to New Results. *Proceedings of the International Conference on Concurrency Theory, Lecture Notes in Computer Science, Vol.*458, *Springer-Verlag*, 1990, 182-198.

[6] **J. Desel.** On Abstraction of Nets. Advances in Petri Nets. *Lecture Notes in Computer Science, Vol.*524, *Springer-Verlag*, 1991, 78-92.

[7] **Y. Souissi.** On Liveness Preservation by Composition of Nets via a Set of Places. *Advances in Petri Nets, Lecture Notes in Computer Science, Vol.*524, *Springer-Verlag*, 1991, 277-295.

[8] **K. Jensen.** Coloured Petri Nets: Petri Nets : Central Models and Their Propertie. *Lecture Notes in Computer Science, Vol.*254, *Springer-Verlag*, 1986, 248-299.

[9] **K. Jensen.** Coloured Petri Nets: Basic Concepts. *Analysis Methods and Practical Use, Vol.*1, *Springer-Verlag*, 1992.

Received April 2006.

APPENDIX

We show the algorithm for eliminating the duplicate labels of a labelled net by fusing common subnets. The net so obtained is uniquely labelled, where the firing sequences (event sequences) are preserved.

Algorithm. Fuse common subnets of a labelled net.

Given. A labelled net.

Output. A uniquely labelled net.

Step 1 : Identify maximal disjoint groups of common subnets, as follows :

1.1 Find all possible common subnets from N. Let $\Im = \{ L_1, L_2, ..., L_n \}$ be their patterns.

1.2 Retain only the maximal patterns : Remove any L_i from $\Im$ if there exists $L_j \in \Im$ such that L_i is a sub-pattern of L_j and $\forall\ S_i \in [N, L_i]$, $\exists\ S_j \in [N, L_j]$: S_i is a subnet of S_j.

1.3 Make the overlapping patterns disjoint : For every $L_i, L_j \in \Im$ such that $L_i \neq L_j$ and L_i and L_j are not disjoint, set $\Im = (\Im - \{ L_i, L_j \}) \cup \{ L_i \cap L_j \} \cup \{ L_i \backslash L_j \} \cup \{ L_j \backslash L_i \}$.

1.4 Categorise the common subnets of N into groups $\{ [N, L_i], L_i \in \Im \}$.

Step 2 : For each group of common subnets [N, L_i], do the following :

2.1 Convert each subnet $S \in [N, L_i]$ if S is not of PP-type :

(a) For each transition $t_i \in$ Head(S) :

- Create dummy transition t_i' with unique label ε_i, dummy place p_i' with label φ_i, and arcs (t_i', p_i') and (p_i', t_i).
- For each place $p \in {}^\bullet t_i$: Remove arc (p, t_i), and then create arc (p, t_i').
- Re-define S by including place p_i' and arc (p_i', t_i).

(b) For each transition $t_j \in$ Tail(S) :

- Create dummy transition t_j' with unique label ε_j, dummy place p_j' with label φ_j, and arcs (t_j, p_j') and (p_j', t_j').
- For each place $p \in t_j^\bullet$: Remove arc (t_j, p), and then create arc (t_j', p).
- Re-define S by including place p_j' and arc (t_j, p_j').

2.2 Assign a unique colour label κ for each subnet $S \in [N, L_i]$:

(a) For each arc (t_i, p_i) such that $t_i \in$ Pre(S) and $p_i \in$ Head(S) : Assign colour label κ to the arc (t_i, p_i).

(b) For each arc (p_j, t_j) such that $p_j \in$ Tail(S) and $t_j \in$ Post(S) : Assign colour label κ to the arc (p_j, t_j).

2.3 Fuse the common subnets in $[N, L_i]$ into one single subnet.

DOI: 10.5755/j01.itc.35.2.12042