

Technical Report TR-92-13
Computer and Info. Sciences Dept., Univ. of Florida
January 1992

Synthesizing a Global Net State from Synchronized Local Pieces*

P. David Stotts [†]	J. Cyrano Ruiz [‡]
Computer and Information Sciences	Department of Reliability Engineering
University of Florida	University of Maryland
Gainesville, FL 32611	College park, MD 20742

Abstract

A traditional problem with state space analysis for concurrent systems has been the exponential explosion in the size of the state space relative to the size of the system. This explosion not only consumes an exponential amount of time in generating a state space, but an exponential amount of storage space as well. In this report we show a technique that attacks both the storage half and the time half of this problem. A synchronous net set is a collection of Petri nets that together model a system of concurrent processes. Rather than combine the process models into a large net for analysis, we keep the nets separate and allow special transitions to synchronize their respective executions implicitly. Analysis is done by computing and storing the state spaces for the individual nets, and then virtually combining the state spaces when a system-wide state query is performed. This approach saves a large amount of storage space over using a global net state space, and for certain conditions requires less time for querying the state space. In addition, it promotes modular models, in that a change to one net model requires recomputation of only the corresponding (relatively small) individual state space, rather than recomputation of the entire system-wide state space.

Key words: Petri nets, modular representation, decomposition, state space compaction, reuse.

CR categories:

F.1.1 (Petri nets) D.2.4 (verification)

*This work is based upon work supported by the National Science Foundation under grant numbers IRI-9007746 and IRI-9015439.

[†]Current address: Dept. of Computer Science, Univ. of North Carolina, Chapel Hill, NC 27599-3175; Internet electronic mail address: `stotts@cs.unc.edu`.

[‡]Internet electronic mail address: `ruiz@cs.umd.edu`.

1 Background

The problem of state-space explosion in the analysis of concurrent systems is an old and still-troublesome problem. Several advances have recently been made in alleviating some of this problem in the context of Petri net system models. Specifically, Valmari has worked on an approach [9, 10] that uses heuristics specific to the question being asked to pre-prune sections of a state space that are *likely* to be unimportant in the context of the analyses. This approach reduces the space to be generated and subsequently analyzed, but it also possibly removes from consideration states that would need to be represented for other analysis questions. That is, the method builds reduced spaces based on specific features of the questions to be answered. Other methods for dealing with state explosion are similarly tailored to specific questions [2]. Still other methods [3, 4, 1] are hierarchical in nature, that is, a place or a transition can be expanded into a more detailed net “structure”. State analyses can be performed at various “levels” of expansion or reduction, entailing varying amounts of time and storage space, according to the level of detail required by the question to be answered.

1.1 Summary of report

This report outlines a conservative approach to state-space compaction. To represent a global system state space, the method requires far less storage than traditional approaches, and for certain circumstances required less time to search as well. The result, however, is a full analysis structure containing the information of the traditional coverability graph. Its structure is not otherwise specific to any particular problem or conditions. Though the method works for any arbitrary net model, its biggest savings are realized for a highly structured class of model. In essence, the approach takes advantage of model structure to reduce the storage and time requirements that traditional coverability analysis (ignoring the special structure) would require.

The basic approach is combinational. We implicitly construct a “global” system model as a set of interacting “local” processes and model each process as a separate place/transition net. A state space is computed for each local net, and the global space is inferred from the local spaces. Unlike the Valmari approach, the conservative nature of this method means that any valid global state will be represented in the local collection, but at the expense of storing some extraneous local states that must be weeded out during a query on the state space. The method also differs from compositional approaches in that it is not hierarchical, but rather is an intrinsic (as opposed to explicit) component “attachment” method. Though we present the work in a flat, one-level context, the method should work *in conjunction with* hierarchical composition methods. It is not a substitute for hierarchy.

2 Synchronous net sets

Before giving formal definitions, consider an informal explanation of the structures we are working with. A system consists of a *set* of Petri nets. Structurally the nets are disjoint, but behaviorally they are implicitly linked via certain distinguished “synchronizing” transitions, termed α -transitions, that are shared (through common naming) among nets in the set. Execution of the global system proceeds by firing one transition at a time from among those collectively enabled in the nets of the set, with the exception that an α -transition, when legally fired, causes synchronized simultaneous activity in two (or more) nets.

For example, consider the two nets shown in Figure 1. Each net contains the synchronizing

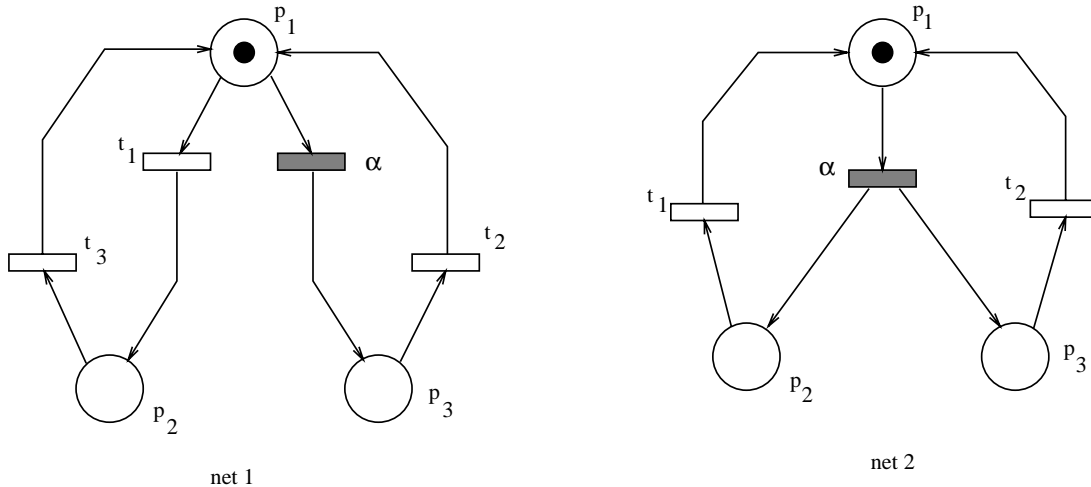


Figure 1: **Set of two Petri nets sharing an α -transition.**

transition α . When interpreting these individual nets collectively as a global system, α actually represents one event. This virtual global net, termed the *synchronous combination* of the set of nets, is shown in Figure 2; α has been used to “hook together” the two nets as a synchronizing event.¹ The places and the normal transitions each net have remained distinct in the combination; each has been renamed, with the superscript indicating its net of origin.

Since global execution of a set proceeds by selecting an enabled transition in one of the nets and firing it, a possible next state for Figure 1 is shown in Figure 3. Here, t_1 in net 1 has been fired resulting in the token distribution as shown. Activity has been limited to net 1. A different possible next state, though, is shown in Figure 4. Since the synchronizing transition α is enabled in all the nets containing it (both of them), it may be fired to give the indicated token distribution. Activity has taken place in *both* nets.

A slightly larger example is given in Figure 5. Here a set of three nets is shown, containing the three synchronizing transitions α , β , and γ . The synchronous combination of these nets is shown in Figure 6.

Any state that can be obtained in a combination net by normal Petri net execution can also be obtained by executing the set of structurally disjoint nets with the distributed execution rule as given. The main contribution of this paper is a method for inferring the state space of the combination net from the individual state spaces for the individual nets in the set. The collection of individual state spaces requires far less storage (average case) than the single state space for the synchronous combination of the individual nets. We also show that *inferring global state* from the individual collection often takes less time than searching the corresponding global space would require.

Following are formal definitions of these concepts, using both a structural perspective and an executional perspective. At times in the paper we use subscripted notation to denote the collection of α -transitions appearing in a set of nets: $\Omega = \{\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_g\}$. At other times, we will find it more convenient to use lower case Greek letters, instead of subscripts, to refer to synchronizing transitions: $\Omega = \{\alpha, \beta, \gamma, \delta, \epsilon, \dots\}$. The context for each use should prevent confusion.

¹Note that the method described herein does not construct the synchronous combination net; it is shown simply as an aid in understanding the execution behavior of a set of nets.

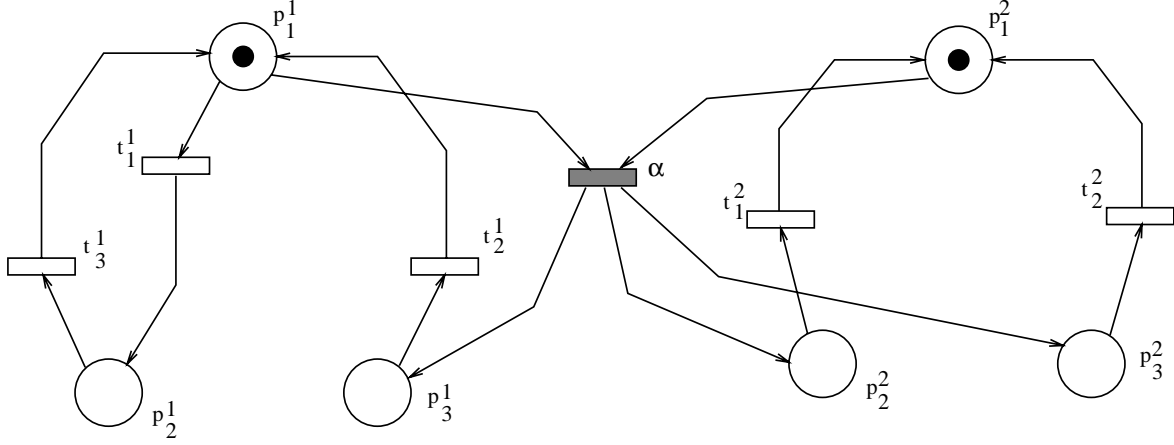


Figure 2: Synchronous combination of nets in Figure 1.

2.1 A structural perspective

Let $N_1 = \langle S_1, T_1, F_1 \rangle$, $N_2 = \langle S_2, T_2, F_2 \rangle$, \dots , $N_k = \langle S_k, T_k, F_k \rangle$ be k Petri nets. Suppose that for each of these Petri nets its set of transitions T_i contains ‘normal transitions’ and a subset of Ω of α -transitions. That is, for each i between 1 and k , $T_i = \{ t_1, t_2, \dots, t_{n_i} \} \cup \Omega_i$, where each of the t_j , is a non α -transition, and Ω_i is a subset of Ω . We refer to these Petri nets $N_1 \dots N_k$ as α -subnets, or α -nets.

Now we define the virtual global system of these α -subnets, termed their *synchronous combination*:

Definition 1 Let $N_1 = \langle S_1, T_1, F_1 \rangle$, $N_2 = \langle S_2, T_2, F_2 \rangle$, \dots , $N_k = \langle S_k, T_k, F_k \rangle$ be k α -subnets. Their synchronous combination is $\mathcal{N} = \langle \mathcal{P}, \mathcal{T}, \mathcal{F} \rangle$ where:

1. \mathcal{P} is the *disjoint union*² of the S_i . If m_i represents the cardinality of each S_i , \mathcal{P} is given by:

$$\mathcal{P} = \{ p_1^1, p_2^1, \dots, p_{m_1}^1, p_1^2, p_2^2, \dots, p_{m_2}^2, \dots, p_1^k, p_2^k, \dots, p_{m_k}^k \}$$

2. \mathcal{T} is the disjoint union of all non α -transitions plus the set Ω of α -transitions. If n_i is the number of non α -transitions in each set T_i , we may write:

$$\mathcal{T} = \{ t_1^1, t_2^1, \dots, t_{n_1}^1, t_1^2, t_2^2, \dots, t_{n_2}^2, \dots, t_1^k, t_2^k, \dots, t_{n_k}^k \} \cup \Omega$$

3. The relation \mathcal{F} is obtained by extending in a natural way the relations F_i to the sets \mathcal{P} and \mathcal{T} . For instance, $(p_i, t_j) \in F_k$ implies that $(p_i^k, t_j^k) \in \mathcal{F}$ (whenever t_j is not one of the α -transitions). Furthermore, for $1 \leq j \leq |\Omega|$, the preset and poset of $\alpha_j \in \Omega$ are given by:

$$\bullet \alpha_j = \{ p_i^k \mid (p_i, \alpha_j) \in F_k \}$$

²We have in mind the mathematical definition of disjoint union, namely the coproduct of a family of sets in the category of sets. This is normally defined as a set of pairs in which the first component is an element of the (usual) union of the sets, and the second component is an index. In this context however, it seems more natural to utilize subscripts and superscripts.

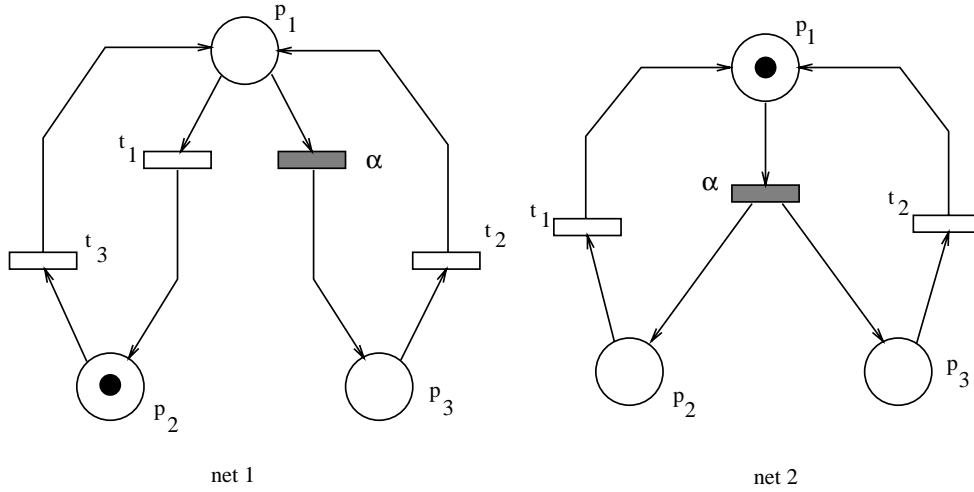


Figure 3: Next state, after firing t_1 in net 1 of Figure 1

$$\alpha_j \bullet = \{p_i^k \mid (\alpha_j, p_i) \in F_k\}$$

2.2 An executional perspective

An *executional* or dynamic interpretation of a synchronous combination of α -subnets can be had by specifying the next-state function δ of this virtual global net. By “state” we mean the common interpretation: a vector of integers indicating the number of tokens in each net place at a given point of execution. As in common automata notation, $\delta(\mu, t)$ produces a next state μ' by firing transition t in current state μ .³ Note that it is convenient in our use of this notation to define $\delta(\mu, t)$ to be μ unchanged when t is not enabled in μ (or does not exist in the net). δ is easily extendible to operate on a string of transitions, known as a “firing sequence.”

For each i between 1 and k , let δ_i be the next-state transition function of the α -subnet N_i . Let \mathcal{N} be the synchronous combination of the k Petri nets N_i . The function δ for \mathcal{N} may be defined in terms of the δ_i . First, notice that a state in the synchronous combination, termed a *global state*, may be treated as the *concatenation* of the individual states of the smaller α -nets. Let $\mu = \mu_1 \dots \mu_j \dots \mu_n$ be an arbitrary global state of \mathcal{N} , with $1 \leq j \leq n$, and with each μ_i being a marking for the corresponding α -net N_i . Let e refer generically to a transition which may be either normal or an α -transition.

Definition 2 The next-state transition function δ of \mathcal{N} is given by:

$$\delta(\mu_1 \dots \mu_j \dots \mu_n, e) = \begin{cases} \mu_1 \dots \delta_j(\mu_j, t_i) \dots \mu_n & \text{if } e = t_i^j \\ \delta_1(\mu_1, \alpha_j) \dots \delta_n(\mu_n, \alpha_j) & \text{if } e = \alpha_j \in \Omega \end{cases}$$

Intuitively, the first case states that if a non α -transition is fired, then no synchronization among subnets occurs. The effect may be understood by firing the corresponding transition in the α -subnet to which it belonged originally, ignoring the others. The second case states that if an α -transition

³We use the older $\delta(\mu, t) = \mu'$ notation rather than the often preferred $\mu[t > \mu'$ found in Reisig [8]. The δ functional format is more convenient and clear for manipulating the *concatenations* of states required by a set of nets, as explained following.

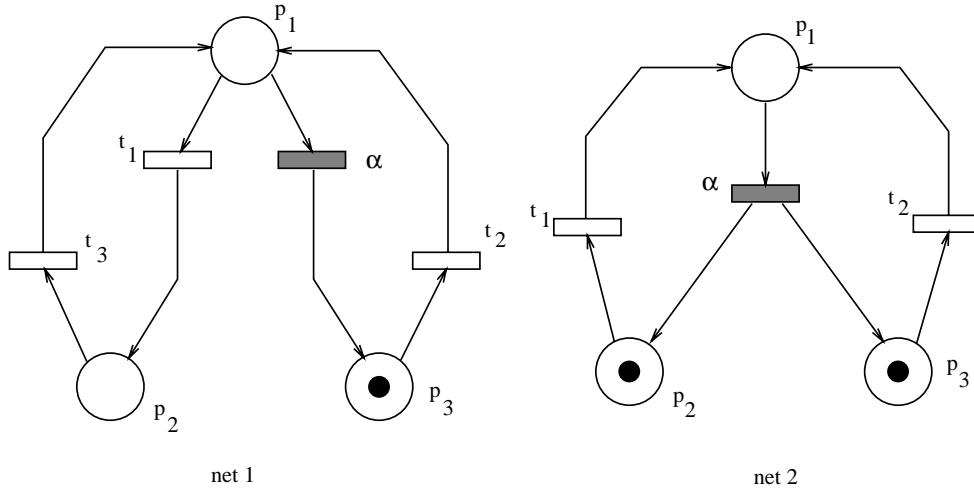


Figure 4: Next state, after firing α in both nets of Figure 1

is fired, then the global effect may be recovered by concatenating the states obtained by firing that α -transition in each of the α -subnets. Remember that $\delta_i(\mu_i, \alpha_j)$ is defined to be μ_i if α_j is not part of N_i .

For a firing sequence $w = e_1 e_2 \dots e_n$, it is easy to define the extended function δ^* recursively. The base case is given by the previous definition, and the inductive one is:

$$\delta^*(\mu_1 \dots \mu_j \dots \mu_n, e_1 e_2 \dots e_n) = \delta^*(\delta(\mu_1 \dots \mu_j \dots \mu_n, e_1), e_2 \dots e_n)$$

3 Piecewise coverability analysis

The coverability tree⁴ of a Petri net plays a central role in the analysis of its execution. In this section, we describe how to use the individual coverability trees of the nets in a set to answer global coverability questions about the synchronous composition of the nets. The method does not require computation of the coverability tree for the global net. It is important to note that in computing the coverability tree of an individual α -net in a set, any α -transitions in it are treated in normal execution fashion, and the synchronizing effects they have in conjunction with the other nets in the set are ignored. This treatment of α -transitions means that each individual coverability tree will most likely contain states that the corresponding α -net will never enter when the synchronous set is executed. When a global coverability query is performed, the global synchronizations are reconstructed as needed and these extraneous states are “weeded out.”

Let N_1, \dots, N_k be α -nets with coverability trees $\Upsilon_1, \dots, \Upsilon_k$ respectively. Let \mathcal{N} be the synchronous combination of the α -nets with Υ being the global coverability tree for \mathcal{N} . Given an arbitrary global state μ , one would like to know if μ is covered by some state in Υ . We wish to solve this problem based *solely* on the coverability trees $\Upsilon_1, \Upsilon_2, \dots, \Upsilon_k$. The storage savings of this approach, and other advantages, are discussed in section 5.

Recall that a global state can be treated as a concatenation of a local state from each of the

⁴We use the expression “coverability tree” interchangeably with “coverability graph.” In the implementation however, we construct trees to take advantage of their properties.

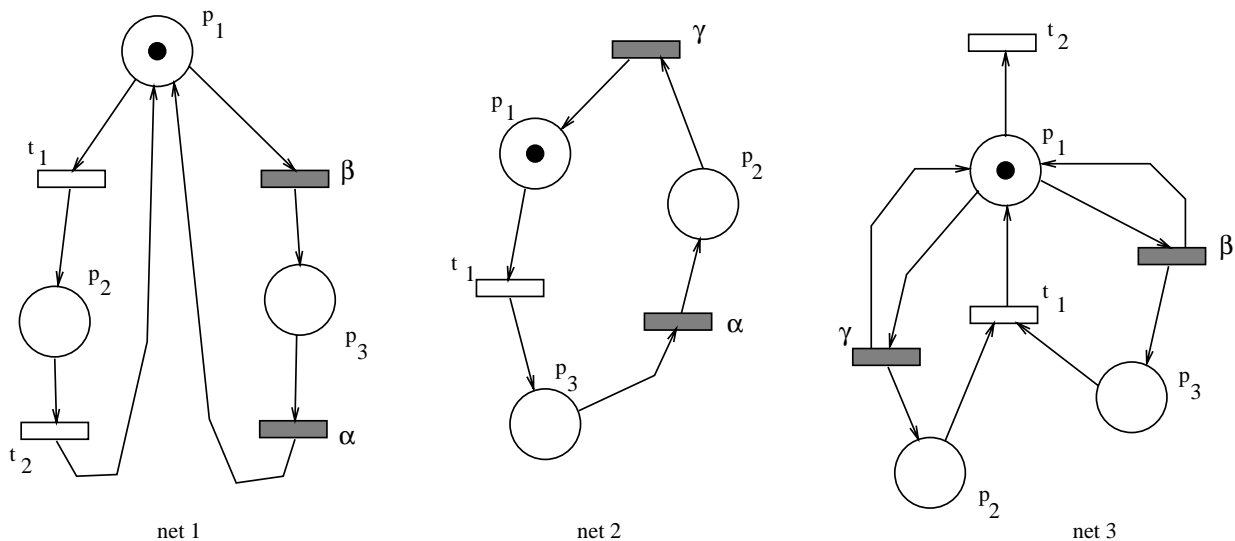


Figure 5: **Example net set with three α -transitions.**

α -subnets. That is, if μ is a state in Υ , then $\mu = \mu_1\mu_2\dots\mu_k$ where μ_i is a state in Υ_i . A few supporting definitions are needed before stating a global coverability theorem.

Definition 3 Let N_i and N_j be two arbitrary α -nets with coverability trees Υ_i and Υ_j respectively. We write T_i and T_j to denote their respective sets of transitions. As defined in section 2.1, let

$$T_i = \{ t_1, t_2, \dots, t_{n_i} \} \cup \Omega_i$$

where n_i represents the number of normal transitions in N_i and Ω_i is a subset of Ω , containing all α -transitions physically present in N_i ; n_j and Ω_j have similar meanings. We use lower case Greek letters to refer to α -transitions.

1. Given a global state $\mu = \mu_1\mu_2\dots\mu_k$, we write Γ_i to denote the set of all firing sequences in Υ_i from its initial state to *some* state covering μ_i .
2. If τ is an arbitrary firing sequence, and T is an arbitrary set of transitions, we write $(\tau)_T$ to refer to the restriction of τ to the set T . This restriction is obtained by dropping the transitions appearing in τ but not belonging to T . For instance, suppose t refers to any non α -transition, then:

$$\text{If } T = \{\alpha, \beta, \gamma\} \text{ and } \tau = tt\beta t\alpha t \text{ then } (\tau)_T \stackrel{\text{def}}{=} \beta\alpha$$

3. Let τ and σ be two firing sequences belonging to Γ_i and Γ_j respectively. Let Ω_{ij} stand for $\Omega_i \cap \Omega_j$. We define

$$\tau \equiv \sigma \pmod{\Omega} \text{ if and only if } (\tau)_{\Omega_{ij}} = (\sigma)_{\Omega_{ij}}$$

For example, suppose that $\Omega_i = \{\alpha, \beta, \gamma\}$ and that $\Omega_j = \{\beta, \gamma\}$. Let $\tau = t\alpha t\alpha t\beta t\alpha t\beta\gamma t$ be a firing sequence in Υ_i and let $\sigma = \beta t t t \beta t t t \gamma t t t$ be a firing sequence in Υ_j . Let t stand for any non α -transition. Then $(\tau)_{\Omega_{ij}} = (\sigma)_{\Omega_{ij}} = \beta\beta\gamma$ and therefore τ and σ are, by definition, congruent to each other modulo Ω .

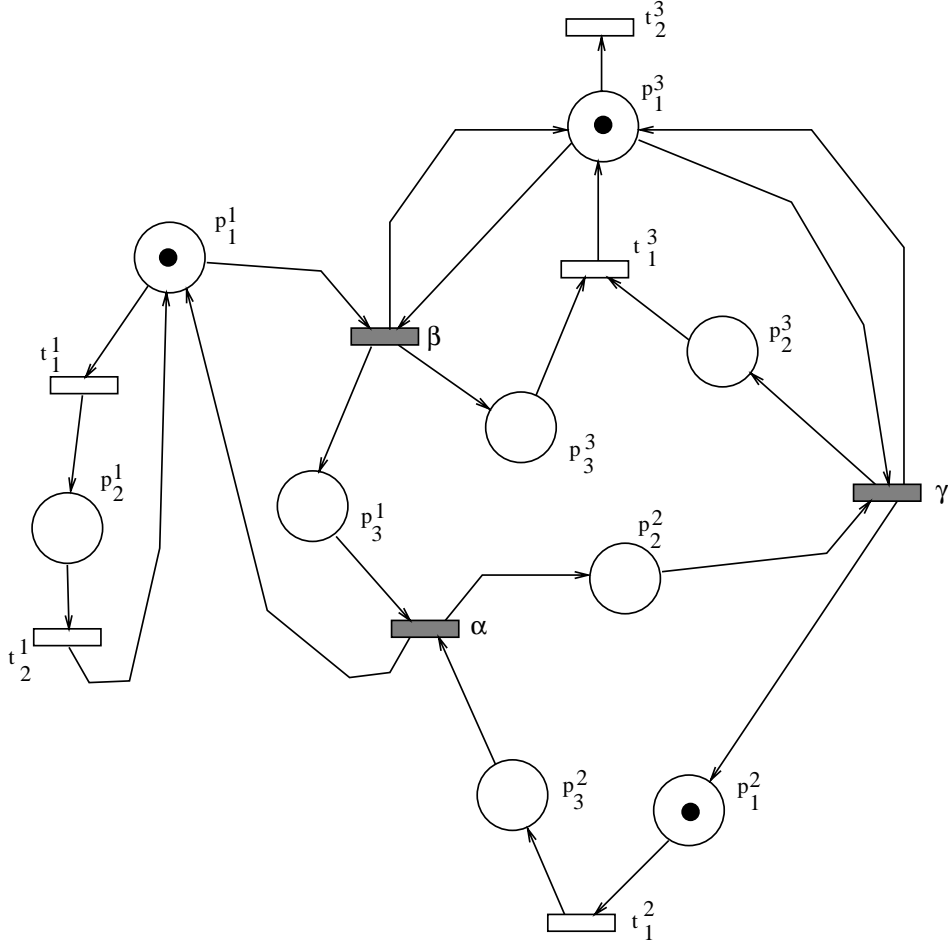


Figure 6: **Synchronous combination of nets in Figure 5.**

Informally, if two firing sequences from two individual nets are congruent modulo Ω , then the α -transitions common to the two nets occur in the firing sequences with the same frequency and in the same relative order. For example, let two nets share synchronizing transitions β and γ . How could two firing sequences from these two nets be compatible if one has transition β firing before γ , but the other has γ before β ? Synchronizing transitions are assumed to fire simultaneously in the nets containing them. Similarly, two sequences are incompatible if one has two γ firings, say, and the other only has one γ firing. Note also that congruence modulo Ω is not transitive. For example, consider $\Omega_1 = \{\alpha, \beta, \gamma\}$, $\Omega_2 = \{\alpha, \beta\}$, and $\Omega_3 = \{\alpha, \gamma\}$; let $\tau_1 = \alpha\epsilon\beta$, $\tau_2 = \alpha\delta\gamma\beta$, and $\tau_3 = \alpha\gamma\beta$. Then $\tau_1 \equiv \tau_2$, and $\tau_2 \equiv \tau_3$, but it is not the case that $\tau_1 \equiv \tau_3$.

Theorem 1 Let N_1, \dots, N_k be α -nets with coverability trees $\Upsilon_1, \dots, \Upsilon_k$ respectively. Let \mathcal{N} be their synchronous combination with global coverability tree Υ . Let $\mu = \mu_1\mu_2 \dots \mu_k$ be an arbitrary global state. State μ is coverable in Υ if and only if every Γ_i contains some τ_i such that

$$\forall l, 1 \leq l \leq k : \forall t, 1 \leq t \leq k : \tau_l \equiv \tau_t \pmod{\Omega}$$

where $1 \leq i \leq k$. In other words, a collection of *pairwise* congruent modulo Ω firing sequences must be found, one from each Γ_i (remember that Γ_i is computed specifically for μ_i).

PROOF: The theorem requires a bidirectional proof. The proof is in fact *constructive* in the sense that we will be able to recover the firing sequences.

“ \Leftarrow ” :

First suppose that μ is coverable in Υ . We need to show the existence of pairwise congruent firing sequences, each one covering its portion of μ within its respective subnet. This is almost immediate from definitions 2.1 and 2.2. Suppose we are trying to construct k firing sequences (for the k α -subnets) given a global firing sequence $e_1 e_2 \dots e_m$ (where each e_i is either an α -transition or a non α -transition). Let $\tau_1, \tau_2, \dots, \tau_k$ be the k firing sequences to be constructed; initially they are all empty. We read through the sequence of e_i and consider for each two mutually exclusive and exhaustive cases (for $i = 1, \dots, k$):

- $e_i = \alpha_j$ for some j . Then concatenate α_j to all the τ_i corresponding to α -subnets physically containing α_j .
- $e_i = t_j^p$ for some j and p . In that case, concatenate t_j to τ_p corresponding to the firing sequence of the α -subnet N_p .

It is clear that each firing sequence τ_i covers its portion μ_i of the global state, and that they are also pairwise congruent modulo Ω .

“ \Rightarrow ” :

Conversely, suppose we have the τ_i as specified in the theorem. To show that μ is coverable in Υ , it suffices to show that there is a *feasible* firing sequence covering μ in the synchronous combination.

First, given a firing sequence τ_i , we need to distinguish among the possibly various occurrences of a same α -transition within τ_i . We use a superscript or a subscript depending on the notation to refer to instances of α -transitions within a firing sequence. For example, using the lower case Greek letter notation, if $\tau_i = \beta\gamma\alpha\beta\alpha$, we will write $\tau_i = \beta_1\gamma_1\alpha_1\beta_2\alpha_2$; using the α -subscript notation however, the firing sequence $\tau_i = \alpha_2\alpha_3\alpha_1\alpha_2\alpha_1$ will be written as $\tau_i = \alpha_2^1\alpha_3^1\alpha_1^1\alpha_2^2\alpha_1^2$.

Secondly, we define a partial ordering on the occurrences of both the α -transitions and the non α -transitions. In this definition, we are assuming a global state $\mu = \mu_1\mu_2 \dots \mu_k$, and Γ_p is the set of firing sequences by means of which the individual net N_p covers μ_p , its portion of the global state (for $p = 1, \dots, k$).

$$\alpha_i^k \stackrel{\text{def}}{\succ} \alpha_j^t \quad \text{if and only if} \quad \begin{cases} \exists \tau \in \Gamma_p, \text{ for some } p, \text{ s.t. } \alpha_i^k \text{ and } \alpha_j^t \text{ appear in } \tau, \text{ and} \\ \alpha_j^t \text{ directly precedes } \alpha_i^k \text{ in } \tau \end{cases}$$

For non α -transitions, we define their partial ordering similarly. Let t_i^j be a non α -transition and e an arbitrary transition (either α or non α -transition). The superscript j indicates that t_i^j belongs to subnet N_j .

$$t_i^j \stackrel{\text{def}}{\succ} e \quad \text{if and only if} \quad \begin{cases} \exists \tau \in \Gamma_j, \text{ s.t. } t_i^j \text{ and } e \text{ appear in } \tau, \text{ and} \\ t_i^j \text{ directly precedes } e \text{ in } \tau \end{cases}$$

The fact of ‘ \succ ’ being *well defined* is clear in virtue of the pairwise congruency of the τ_i . Indeed, suppose for instance that, using our definition, we get an inconsistency like $\alpha_i^k \succ \alpha_j^t$ and $\alpha_j^t \succ \alpha_i^k$. In this case the two firing sequences τ_{p_1} and τ_{p_2} , whose

existence is guaranteed by the definition, clearly could not be congruent modulo Ω , contradicting our assumption on the τ_i ($i = 1, \dots, k$). The case of non α -transitions is even simpler since in this case the the partial order is induced by the sequential ordering of the transitions within a single firing sequence. This is essentially the reason we do not need to distinguish between occurrences of non α -transitions.

Any partial ordering can be ‘completed’ (in a consistent way) into a total ordering, for instance via a topological sort [7]. In our case, transforming this partial ordering of the occurrences of the α -transitions into a total ordering, can be interpreted as consistently ‘merging’ the individual firing sequences τ_i into a global firing sequence. Because we have respected the partial ordering, this global firing sequence is indeed *feasible*.

□

As an example of the non trivial direction, suppose there are three α -subnets N_1, N_2, N_3 , with respective sets of transitions:

1. $T_1 = \{r, s, t\} \cup \{\alpha, \beta, \gamma\}$
2. $T_2 = \{u, v, w\} \cup \{\beta, \delta\}$
3. $T_3 = \{x, y, z\} \cup \{\alpha, \beta, \delta, \zeta\}$

Now, for the three firing sequences in the α -subnets, say, for instance, that:

- $\tau_1 = \alpha r s \beta \alpha s \gamma t$ ($= \alpha_1 r s \beta_1 \alpha_2 s \gamma_1 t$)
- $\tau_2 = u \delta \delta \beta w \delta v$ ($= u \delta_1 \delta_2 \beta_1 w \delta_3 v$)
- $\tau_3 = x \zeta \delta \zeta \delta y \alpha y x \beta \zeta z \delta \alpha$ ($= x \zeta_1 \delta_1 \zeta_2 \delta_2 y \alpha_1 y x \beta_1 \zeta_3 z \delta_3 \alpha_2$)

First, notice that τ_1 , τ_2 , and τ_3 are readily seen to be pairwise congruent modulo Ω . The partial ordering induced by these firing sequences on the transitions is shown in Figure 7. By “completing” this partial ordering, we can obtain a feasible global firing sequence. One possible total orders induced by the partial order in Figure 7 is

$$\tau = x u \zeta_1 \delta_1 \zeta_2 \delta_2 y \alpha_1 r y s x \beta_1 w \zeta_3 z \delta_3 v \alpha_2 s \gamma_1 t \quad (= x u \zeta \delta \zeta \delta y \alpha r y s x \beta w \zeta z \delta v \alpha s \gamma t)$$

For completeness, we now formulate the result we were originally seeking about piecewise coverability.

Theorem 2 Let N_1, \dots, N_k be α -nets with coverability trees $\Upsilon_1, \dots, \Upsilon_k$ respectively. Let \mathcal{N} be their synchronous combination with coverability tree Υ . It is *possible* to determine whether an arbitrary global state $\mu = \mu_1 \mu_2 \dots \mu_k$ is coverable or not, based solely on the $\Upsilon_1, \dots, \Upsilon_k$.

PROOF: The sets of firing sequences $\Gamma_1, \dots, \Gamma_k$ are computable, given each N_i and each corresponding μ_i , as explained for example in [6]. All these Γ_i sets are finite, so we may apply the necessary and sufficient conditions of theorem 1 to obtain the desired global coverability results.

□

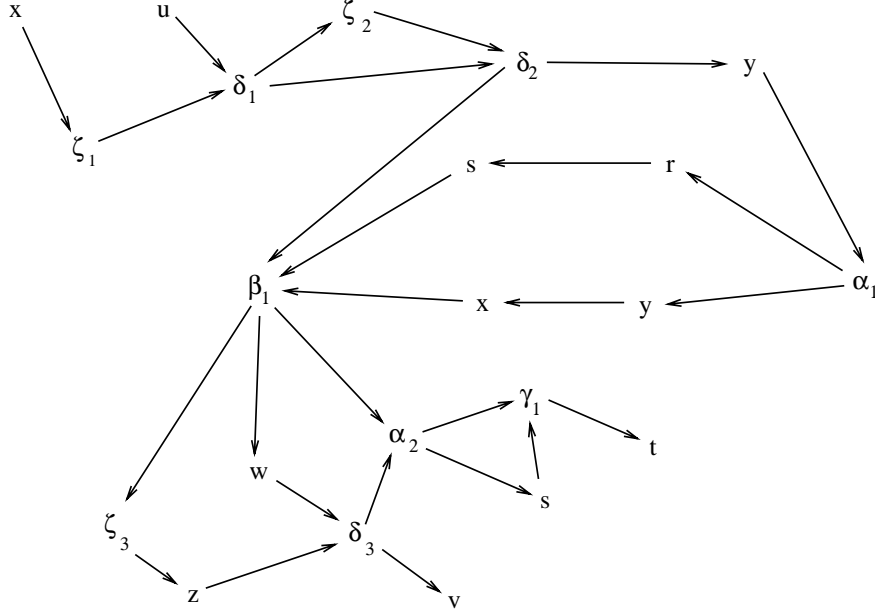


Figure 7: **Partial ordering induced by τ_1, τ_2 and τ_3 . In the figure, we write: ‘ $\alpha \leftarrow \beta$ ’ to mean: ‘ $\alpha \succ \beta$.’**

4 An algorithm for bounded Petri nets

The coverability problem is answered by searching the nodes in a coverability tree for a covering state. Assuming the tree is threaded into a list, the cost of this checking is, on the average, a visit to half the nodes in the coverability tree (assuming that the state we are checking is coverable, that is, a state can be found in the tree without exhausting the list).

Let m_i be the number of places in α -nets N_i . Clearly, the number of places in the synchronous combination of a set of nets is the sum of all the m_i . Call this sum M . By definition, a B -bounded Petri net is one in which no place will contain more than B tokens during execution. From net theory, we know that in the case of *bounded* nets, the size (number of states) in a coverability tree is in general exponential in the number of places in a net, on the order of $(B + 1)^{m_i}$ (where m_i is the number of places). In the special case of a *safe* Petri net, the number of nodes in a coverability tree is on the order of 2^{m_i} .

For the sake of simplicity, in the rest of this section, we assume that we are dealing with *safe* Petri nets. We present an efficient algorithm to check global coverability for the synchronous combination of a set of safe α -nets. The algorithm requires the coverability tree of each individual α -net.

Given the previous size estimates for reachability trees, if we were given the coverability tree Υ of the synchronous combination containing M states it would cost⁵ on the average

$$\frac{2^M}{2} \text{ that is: } 2^{M-1} \quad (1)$$

to check if a global state is covered by synchronous combination of the subnets. This reflects a search over a list of states.

For each i , let δ_i be the depth of the tree Υ_i , let $\Delta = \max(\delta_1, \dots, \delta_k)$ and let $g = |\Omega|$ be the total number of α -transitions. On the average, each of the δ_i is a logarithmic function of the number

⁵When we use the word “cost,” we mean cost in searching time.

M	1 net	4 subnets	8 subnets	16 subnets
10	512	4,124	—	—
20	524,288	4,194,624	262,182	—
30	536,870,912	4,294,970,011	47,453,267	524,316
40	549,755,813,888	4,398,046,531,584	8,589,935,019	16,777,273

Table 1: **Table showing the relative costs of checking coverability for safe Petri nets. M is the number of places in the net produced by synchronous combination.**

of nodes in its respective coverability tree. If n_i represents the number of transitions, and m_i the number of places in each subnet, then

$$\delta_i = \log_{n_i}(2^{m_i})$$

These numbers δ_i may be obtained “for free” when constructing the coverability trees.

Let m_Δ be the number of nodes in *the* (or maybe one of the) α -subnet with coverability tree having depth Δ . We will show that the algorithm that we propose here has a cost of:

$$\sum_{i=1}^k \delta_i 2^{m_i} + k g^{m_\Delta} \quad (2)$$

For large values of M , and where these M nodes are divided over more than a small number ($\tilde{4}$) of subnets, the cost shown in expression (1) is much higher than the cost shown in expression (2). This may be appreciated in table (4). For simplicity, to compute this table we assume the following:

- The synchronous combination net as well as the subnets have worst-case coverability trees.
- Each α -subnet has a coverability tree of the same size.
- In each α -subnet, the number of places is equal to the number of transitions.
- All the nets are *safe* Petri nets (as explained previously).

We now give the algorithm. We first define a “bit hypercube.” Remember from definition 3 that given a global state $\mu = \mu_1 \mu_2 \dots \mu_k$, Γ_i denotes the set of all firing sequences in Υ_i , from its initial state to *some* state covering μ_i . This bit hypercube will encode the firing sequences contained in the Γ_i sets. For technical reasons which shall soon be clear, α_0 denotes the *empty string*.

Definition 4 Let $\Upsilon_1, \dots, \Upsilon_k$ be the coverability trees of k α -nets. For each state $\mu = \mu_1 \mu_2 \dots \mu_k$ of their synchronous combination, we define a hypercube Φ_μ whose elements are boolean arrays, each one having k bits. This hypercube has Δ dimensions (one for each transition in the longest firing sequence), with each subscript having a value from 0 to g (representing each α -transition in Ω , plus 0 for no transition), and

$$\Phi_\mu(i_1, i_2, \dots, i_\Delta)[j] = \mathbf{0N} \text{ if and only if } \exists \tau \in \Gamma_j : (\tau)_\Omega = \alpha_{i_1} \alpha_{i_2} \dots \alpha_{i_\Delta}$$

Notice that we allow some of the i_j to be zero. In this case, by the remark preceding this definition, the corresponding α_{i_j} would be the empty string.

We also need to define a $k \times g$ bit matrix representing in the obvious way whether or not the net N_i physically contains transition α_j . We call this matrix the α -cross matrix, and refer to it as D . Its rows indicate α -transitions, and its columns α -subnets. The matrix D may be considered as given for all practical purposes. More specifically:

Definition 5 The α -cross matrix D is defined by:

$$\forall i, 1 \leq i \leq k \quad \forall j, 1 \leq j \leq g, D[i, j] = \text{ON} \text{ if and only if } \alpha_j \in T_i$$

Having defined these matrices, we now propose a more efficient algorithm containing two phases. In the first step, we construct the hypercube. This may be considered the expensive part, and we will show that it actually costs the first summand shown in (2). Secondly, based on the above hypercube, we answer the coverability question at a cost equal to the second summand of (2).

COVERABLE ($\mu = \mu_1 \mu_2 \dots \mu_k$, $\Upsilon_1, \dots, \Upsilon_k$)

Part I: Constructing Φ_μ .

```

Initialize  $\Phi_\mu$  to OFF's;
FOR  $i = 1$  TO  $k$  BEGIN
  FOR each node  $\eta$  in  $\Upsilon_i$  BEGIN
    IF  $\eta$  covers  $\mu_i$  THEN BEGIN
      recover the path  $\tau$  from  $\eta$  to the root of  $\Upsilon_i$ ;
      Say:  $(\tau)_\Omega = \alpha_{i_1} \alpha_{i_2} \dots \alpha_{i_\Delta}$ 
      REM Because some of the  $i_j$  may be zero, there would be many possible
      REM ways to write the above sequence. However, to have uniqueness, we
      REM required that if some  $i_j$  is zero, then all subsequent indexes be also zero.
       $\Phi_\mu(i_1, i_2, \dots, i_\Delta)[i] \leftarrow \text{ON}$ ;
    END
  END
END
END
END

```

Part II: Examining Φ_μ

```
FOR each  $\Phi_\mu(i_1, i_2, \dots, i_\Delta)$  BEGIN
   $b \leftarrow \mathbf{true}$ ;
  FOR  $j = 1$  TO  $k$  BEGIN
     $b \leftarrow \Phi_\mu(i_1 \star D(1, j), i_2 \star D(2, j), \dots, i_g \star D(g, j))[j] \wedge b$ ;
  END
  IF  $b$  THEN BEGIN
    OUTPUT yes;
    EXIT;
  END
END
OUTPUT false;
```

The integer-boolean multiplication: \star is defined in the obvious way, namely:

$$i_t \star D(t, j) = \begin{cases} i_t & \text{if } D(t, j) = \mathbf{ON} \\ 0 & \text{if } D(t, j) = \mathbf{OFF} \end{cases}$$

The analysis of this algorithm is straightforward. Part I clearly has a cost of:

$$\sum_{i=1}^k \delta_i 2^{m_i}$$

Similarly, part II takes:

$$kg^{m_\Delta}$$

Therefore, in total, the coverability checking can be done in a cost as indicated by expression (2).

5 Discussion and conclusions

The time-cost estimates in this analysis seem to say that the piecewise coverability algorithm is faster than the traditional coverability checking algorithm on one large system tree. Actually, what the results most likely say is that for large systems that can be described as a collection of loosely coupled “processes,” the bounds we give are far better estimates of the time-cost than those obtained from worst case estimates of the size of a global coverability tree. Since the system is highly partitioned, the worst case tree is almost certainly not going to be realized.

The storage saving, however, are real. The total size of the coverability trees for a collection of small nets will be very much smaller than the size of the tree for an equivalent global system net. This savings is practical and significant.

In addition to storage savings, some time saving is realized during the construction of a system model, when the structure is being changed frequently. When a subnet is altered, only the individual coverability tree for that subnet must be recomputed, and the trees for all other subnets can remain unchanged. Since an individual state space is small compared to the global system state space, far less time will be required to reclaim an accurate representation of the global state space after a structural change. Petri nets in this manner are *reusable*, or modular.

Our analysis technique assumes the process structure of the total system is created *a priori*, as if it were derived, say, from a parallel program or other system description (like CSP [5]). An interesting problem that we have not yet worked on is to try and identify processes *within* existing global nets, thereby making this technique applicable to existing large models. The technique would be to identify loosely connected subnets, and make the connecting transitions α -transitions. The resulting net set would gain for the system modeler the space and time advantages we have outlined.

References

- [1] G. Berthelot, G. Roucairol, and R. Valk. reductions of nets and parallel programs. In W. Brauer, editor, *LNCS 84: Net Theory and Applications*, pages 277–290. Springer–Verlag, 1980.
- [2] J. Billington. Protocol engineering and nets. In *Proceedings of the 8th European Workshop on Application and Theory of Petri Nets*, pages 137–156, 1987.
- [3] G. Comparin, G. A. Lanzarone, K. Lautenbach, A. Pagnoni, W. Panzeri, and A. Torgano. Guidelines on using net analysis techniques with large specifications. In G. Rozenberg, editor, *LNCS 222: Advances in Petri Nets 1985*, pages 142–161. Springer–Verlag, 1985.
- [4] L. Czaja. Making nets abstract and structured. In G. Rozenberg, editor, *LNCS 222: Advances in Petri Nets 1985*, pages 181–202. Springer–Verlag, 1985.
- [5] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.
- [6] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3:147–195, May 1969.
- [7] Donald Knuth. *The Art of Computer Programming: Fundamental Algorithms (Volume 1)*. Addison Wesley, 1970.
- [8] Wolfgang Reisig. *Petri Nets: An Introduction*. Springer-Verlag, 1985.
- [9] Antti Valmari. *State Space Generation: Efficiency and Practicality*. Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, 1988. Issued as Technical Publication No. 55.
- [10] Antti Valmari. Stubborn sets for reduced state space generation. In G. Rozenberg, editor, *LNCS 483: Advances in Petri Nets 1990*, pages 491–515. Springer–Verlag, 1990.