TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

A Polynomial Algorithm to Compute the Concurrency Relation of Free–Choice Signal Transition Graphs

Andrei Kovalyov, Javier Esparza

TUM-I9528
SFB-Bericht Nr.342/15/95 A
Oktober 1995

TUM–INFO–10-95-I28-0/1.–FI

Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©1995 SFB 342 Methoden und Werkzeuge für
die Nutzung paralleler Architekturen

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
Arcisstr. 21 / Postfach 20 24 20
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

A polynomial algorithm to compute the concurrency relation of free-choice Signal Transition Graphs *

Andrei Kovalyov and Javier Esparza
Institut für Informatik
Technische Universität München

Abstract

The concurrency relation of a Petri net contains the pairs of transitions that can be concurrently enabled. We present a polynomial algorithm to compute the concurrency relation of free-choice Signal Transition Graphs, a class of Petri nets with applications to the verification and synthesis of speed-independet circuits.

1 Introduction

Signal Transition Graphs (STGs) have become a popular and much studied formalism for the specification and verification of speed independent circuits [3, 8, 9]. STGs are 1-bounded Petri nets whose transitions carry labels of the form y^+, y^-, where y is a circuit signal.[1] The occurrence of a transition with label y^+ *raises* y, i.e., sets its value to 1, while the occurrence of a transition with label y^- *lowers* y, i.e., sets its value to 0.

A speed-independent circuit should never reach a state in which one circuit element wishes to lower the signal while another independently wishes to raise it, because then the behaviour of the circuit depends on the timing characteristics of the elements involved [12]. In an STG model this situation corresponds to the existence of a reachable marking which concurrently enables two transitions with opposite labels like y^+ and y^-. Only STGs avoiding this situation can be implemented.

Another necessary condition for an STG to be implementable is the Complete State Coding property or the more restrictive Unique State Coding property [3] [2]. Pastor and Cortadella have recently described a procedure to verify if a free-choice STG satisfies the Unique State Coding property [10, 11] which requires to compute the concurrency relation, i.e., the pairs of transitions that can be concurrently enabled.

*This work was Partially supported by the Sonderforschungsbereich 342 of the Deutsche Forschungsgemeinschaft, Teilprojekt A3.

[1] Originally, Chu defined STGs as free-choice Petri nets [3]. So Chu's STGs are what we call free-choice STGs in this paper.

[2] The precise definition of this property is not important for this paper.

The discussion of [12] and the result of [10] show that the concurrency relation of STGs has important applications. In this paper we present an algorithm which computes the concurrency relation for live and bounded free-choice nets, and therefore for live free-choice STGs, in $O(n^3)$ time, where n is the number of places and transitions of the net. The restriction to live STGs is not significant, because non-live STGs are considered incorrect models, and liveness of free-choice STGs can be efficiently decided (see, for instance, [4, 7]). The algorithm is obtained following the lines of [6]. After some basic definitions (Section 2), we define the structural concurrency relation of a Petri net (Section 3). Then we prove that the concurrency and the structural concurrency relation coincide for live and bounded free-choice Petri nets (Section 4). Finally, we show that the structural concurrency relation can be computed in $O(n^3)$ time, improving the algorithm given in [6], whose time complexity was $O(n^5)$ (Section 5).

2 Basic definitions

A *net* N is a triple (S, T, F), where S and T are two disjoint, finite sets of *places* and *transitions*, and $F \subseteq (S \times T) \cup (T \times S)$ is a *flow relation*. Places and transitions are generically called *nodes*. We identify F with its characteristic function $(S \times T) \cup (T \times S) \to \{0, 1\}$.
Given a node x of N, ${}^\bullet x = \{y \mid (y, x) \in F\}$ is the *preset* of x and $x^\bullet = \{y \mid (x, y) \in F\}$ is the *postset* of x. Given a set of nodes X of N, we define ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$.
A triple (S', T', F') is a *subnet* of N if $S' \subseteq S$, $T' \subseteq T$ and $F' = F \cap ((S' \times T') \cup (T' \times S'))$. If X is a set of elements of N, then the triple $(S \cap X, T \cap X, F \cap (X \times X))$ is a subnet of N, called the subnet of N *generated by* X.
A net (S, T, F) is a *T-net* if $|{}^\bullet s| = 1 = |s^\bullet|$ for every place s. It is a *free-choice net* if $(s, t) \in F$ implies ${}^\bullet t \times s^\bullet \subseteq F$ for every place s.[3]
A *labelled net* N is a fourtuple (S, T, W, l), where (S, T, F) is a net, and l is a mapping $T \to Act$, where Act is a set of actions. A *marking* of N is a mapping $M\colon S \to \mathbb{N}$. A marking M *enables* a transition t if $M(s) \geq F(s, t)$ for every place s. If t is enabled at M, then it can *occur*, and its occurrence leads to the successor marking M' which is defined for every place s by

$$M'(s) = M(s) + \sum_{t \in T} F(t, s) - F(s, t)$$

A *Petri net* or *net system* is a pair $\Sigma = (N, M_0)$ where N is a connected net and M_0 is a marking of N. The connectedness of N is not a constraint, because the concurrency relation of a net system can be easily obtained from the concurrency relations of its connected components. The expression $M_1 \xrightarrow{t} M_2$, where M_1, M_2 are markings of N, denotes that M_1 enables transition t, and that the marking reached by the occurrence of t is M_2. The empty sequence ϵ is an occurrence sequence: we have $M \xrightarrow{\epsilon} M$ for every marking M.
A *T-system* (*free-choice system*, *labelled system*) is a pair $\Sigma = (N, M_0)$ where N is a T-net (free-choice net, labelled net) and M_0 is a marking of N.
A net system is *live* if for every reachable marking M and every transition t there exists a marking M' reachable from M which enables t. If (N, M_0) is a live system, then we also say that M_0 is a live marking of N.

[3] We follow the terminology of [4]. These nets are also called extended free-choice nets in the literature.

A net system is *b-bounded* if $M(s) \leq b$ for every place s and every reachable marking M. A net system is *bounded* if it is b-bounded for some number b. If (N, M_0) is a bounded system, we also say that M_0 is a bounded marking of N.

3 Concurrency relations

The concurrency relation is usually defined as a set of pairs of transitions. We use a more general definition.

Let (N, M_0) be a net system, and let X be the set of nodes of N. Given $x \in X$, define the marking M_x of N as follows:

- if x is a place, then M_x is the marking that puts one token on s, and no tokens elsewhere;
- if x is a transition, then M_x is the marking that puts one token on every input place of x, and no tokens elsewhere.

The *concurrency relation* $\| \subseteq X \times X$ contains the pairs (x_1, x_2) such that $M \geq M_{x_1} + M_{x_2}$ for some reachable marking M. In particular, two transitions t_1, t_2 belong to the concurrecy relation if they can occur concurrently from some reachable marking, and two places belong to the concurrency relation if they are simultaneously marked at some reachable marking.

The concurrency relation is very related to the *co* relation used in the theory of nonsequential processes [1]: (x_1, x_2) belongs to the concurrency relation if and only if some nonsequential process of (N, M_0) contains two elements in *co* labelled by x_1 and x_2. This is in fact a more elegant definition, but, since it requires to introduce a number of concepts, we use the one above.

We now define the structural concurrency relation, first presented in [6]:

Definition 3.1 *Structural concurrency relation*

Let (N, M_0) be a system, where $N = (S, T, F)$, and let $X = S \cup T$. The *structural concurrency relation* $\|^A \subseteq X \times X$ is the smallest symmetric relation such that:

(i) $\forall s, s' \in S$: $M_0 \geq M_s + M_{s'} \Rightarrow (s, s') \in \|^A$

(ii) $\forall t \in T$: $({}^\bullet t \times {}^\bullet t) \setminus id_T \subseteq \|^A \Rightarrow (t^\bullet \times t^\bullet) \setminus id_T \subseteq \|^A$

(iii) $\forall x \in X \; \forall t \in T$: $\{x\} \times {}^\bullet t \subseteq \|^A \Rightarrow (x, t) \in \|^A \wedge \{x\} \times t^\bullet \subseteq \|^A$

where id_T denotes the identity relation on T.

Loosely speaking, condition (i) states that any two places marked at the initial marking are structurally concurrent (actually, this is the case for a pair (s, s) only if M_0 puts at least two tokens on s). Condition (ii) states that if all the input places of a transition are structurally concurrent, then so are its output places. Clearly, these two conditions are fulfilled by the concurrency relation $\|$. At first sight it could seem that $\|$ also fulfills condition (iii), but this is not the case. This condition states that if a node is structurally concurrent with all the input places of a transition, then it is also structurally concurrent with all its output places.

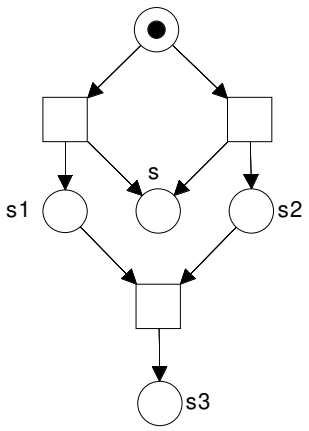

Fig. 1 A system for which $\| \neq \|^A$

Figure 1 shows a system in which $\|$ does not satisfy (iii): (s, s_1) and (s, s_2) are concurrent, because there are two *different* reachable markings which mark s, s_1 and s, s_2, respectively, but there is no reachable marking which puts tokens simultaneously on s, s_1 and s_2. So for this system we have $\| \neq \|^A$. Another example in which the system is live and 1-bounded can be found in [6].

The following results are proven in [6]:

Theorem 3.2

(i) For every net system, $\| \subseteq \|^A$.

(ii) For live T-systems, $\| = \|^A$.

■

4 The main result

We prove in this section that $\|^A$ and $\|$ coincide for live and bounded free-choice systems (and therefore for live free-choice STGs). The reader which is not interested in the details of the proof can safely jump to the next section.

The proof has much in common with the proof of the Second Confluence Theorem [4], which we now recall.

4.1 The Second Confluence Theorem

All the definitions and results of this subsection are taken from [4]. The Second Confluence Theorem states that if two live and bounded markings M_1 and M_2 of a free-choice net agree on all S-invariants, then they have a common successor, i.e., there exists a marking that is reachable from both M_1 and M_2. Since it can be easily shown that any two reachable markings agree on all S-invariants, it follows that any two reachable markings have a common

successor. The result can be generalised to a set $M_1, \ldots, M_n$ of markings which agree pairwise on all invariants, and in the sequel we consider this more general version.

Let us first recall the notions of S-invariant and markings that agree on all S-invariants.

Definition 4.1

An S-invariant of a net N is a rational-valued solution of the equation $X \cdot \mathbf{N} = \mathbf{0}$.

Two markings M and M' of N agree on all S-invariants if $I \cdot M = I \cdot M'$ for every S-invariant I of N.

Theorem 4.2

Let (N, M_0) be a system, and let M be a reachable marking. Then M and M_0 agree on all S-invariants. ■

The proof of the Second Confluence Theorem distinguishes two cases, according to whether the free-choice net N is a T-net or not. The first case is easily solved using the following result, which states that for T-systems the converse of Theorem 4.2 holds:

Theorem 4.3 *Reachability Theorem for T-systems*

Let (N, M_0) be a live T-system. A marking M is reachable iff it agrees with M_0 on all S-invariants. ■

Since $M_1, \ldots, M_n$ are live and bounded and agree on all S-invariants, they are all reachable from each other. Therefore, any of them is a common successor of all the others.

In the second case, when N is not a T-net, the proof makes use of a reduction procedure. N is split into two: a subnet $\widehat{N} = (\widehat{S}, \widehat{T}, \widehat{F})$ called CP-subnet in [4], and the subnet generated by all the nodes that do not belong to $\widehat{N}$, denoted by $N \setminus \widehat{N}$.

Definition 4.4 *CP-subnets*

A subnet $N' = (S', T', F')$ of a net N is a CP-subnet if it is

(i) nonempty and weakly connected,

(ii) transition-bordered (i.e., contains pre- and post-sets of all its places),

(iii) a T-net (i.e., every place has exactly one input transition and one output transition), and

(iv) the net $N \setminus N'$ is strongly connected and contains some transition.

A transition $t \in T'$ such that ${}^\bullet t \not\subseteq S'$ is called a *way-in* transition.

The following result guarantees that N can be split:

Proposition 4.5

A well-formed free-choice net is either a T-net or has a CP-subnet. ■

Once N is split, we let n particular sequences occur from $M_1, \ldots, M_n$. These sequences contain only transitions of $\widehat{N}$ which are not way-in transitions.

Proposition 4.6 *Fundamental property of CP-subnets*

Let N be a well-formed free-choice net, and let $M_1, \ldots, M_n$ be live and bounded markings of N that agree on all S-invariants. Let $\widehat{N}$ be a CP-subnet of N, let $\hat{T}_i$ be the set of way-in transitions of $\widehat{N}$, and let $\overline{N} = N \setminus \widehat{N}$. There exist occurrence sequences $M_1 \xrightarrow{\sigma_1} M_1', \ldots, M_n \xrightarrow{\sigma_n} M_n'$, where $\sigma_1, \ldots, \sigma_n$ contain only transitions of $\hat{T} \setminus \hat{T}_{in}$, such that

(1) No transition of $\hat{T} \setminus \hat{T}_{in}$ is enabled at $M_1', \ldots, M_n'$,

(2) $\widehat{M_1'} = \cdots = \widehat{M_n'}$, where $\widehat{M}$ denotes the projection of M onto the places of $\widehat{N}$,

(3) $\overline{M_i} \leq \overline{M_i'}$ for $1 \leq i \leq n$, where $\overline{M}$ denotes the projection of M onto the places of $\overline{N}$, and

(4) $\overline{M_1'}, \ldots, \overline{M_n'}$ are live and bounded markings which agree on all S-invariants of $\overline{N}$.

■

After the occurrence of these sequences we 'freeze' the transitions of the CP-subnet, i.e., we forbid them to occur again, and preserve so the equality $\widehat{M_1'} = \cdots = \widehat{M_n'}$. If $\overline{N}$ is a T-net, then Theorem 4.3 can be applied, and we are done. Otherwise, by Proposition 4.5 and Proposition 4.6(3) we can iterate the procedure until we get two markings which coincide everywhere, and are therefore the same. This marking is a common successor of $M_1, \ldots, M_n$.

Instead of freezing the transitions of the CP-subnet, we can equivalently remove it and consider thereafter the remaining net $\overline{N}$.

4.2 The new result

In order to adapt these results to the concurrency problem for free-choice systems, we have a closer look at the proof of Proposition 4.5, which shows how to construct a CP-subnet of a live and bounded free-choice system which is not a T-system. The proof is based on the notion of T-component.

Definition 4.7 *T-components, T-covers*

Let N' be the subnet of a net N generated by a nonempty set X of nodes. N' is a T-component of N if:

- ${}^\bullet t \cup t^\bullet \subseteq X$ for every transition t of X, and
- N' is a strongly connected T-net.

Let $\mathcal{C}$ be a set of T-components of N. $\mathcal{C}$ is a *T-cover* if every transition of N belongs to a T-component of $\mathcal{C}$. A net is covered by T-components if it has a T-cover.

It is easy to see that every node of the net, and not only every transition, belongs to some element of a T-cover.

We have the following well-known result:

Theorem 4.8 *T-coverability Theorem*

Well-formed free-choice nets are covered by T-components. ■

Now, in order to find a CP-subnet, we proceed as follows. We take a minimal T-cover $\mathcal{C}$ of N, i.e., no proper subset of $\mathcal{C}$ is a cover. Since N is not a T-net, we have $|\mathcal{C}| > 1$. We construct the (non-directed) graph $G = (V, E)$ as follows.

$$V = \mathcal{C}$$
$$E = \{(N_i, N_j) \mid N_i \text{ and } N_j \text{ have at least one common node}\}$$

The graph G is connected because $\mathcal{C}$ is a cover of N and N is connected. Moreover, G has at least two vertices because $|\mathcal{C}| > 1$.

We choose an spanning tree[4] of G, and select one of its leaves, say N_1. We then construct a maximal set of nodes X of N_1 satisfying the following properties:

(a) The net generated by X is connected, and

(b) No element of X belongs to a T-component of $\mathcal{C} \setminus \{N_1\}$.

The set X is nonempty, because $\mathcal{C}$ is a minimal cover. The subnet N_X generated by X is a CP-subnet.

We call the subnets N_X *private* subnets, because they are generated by nodes that N_1 does not share with any other T-component of $\mathcal{C}$. So we have that private subnets are CP-subnets.

Notice that a T-component may have more than one private subnet. However, if this is the case then the private subnets are disjoint (by the maximality condition on X).

We are now ready to prove the following result.

Proposition 4.9

Let (N, M_0) be a live and bounded free-choice system, and let s and t be a place and a transition of N such that ${}^\bullet t = \{r_1, \ldots, r_n\}$. Assume that for every $1 \leq i \leq n$ there exists a reachable marking M_i such that $M_i \geq M_s + M_{r_i}$. Then there exists a reachable marking $M \geq M_s + \sum_{i=1}^{n} M_{r_i}$.

Proof:

Let $\mathcal{C}$ be a minimal T-cover of N, which exists by the T-coverability Theorem. We consider two cases:

(a) Some T-component N_1 of $\mathcal{C}$ contains both s and t.

Let G be the graph described above, and let G_s be a spanning tree of G. We construct systems $(N_1, M_1^f), \ldots, (N_1, M_n^f)$ as follows. If G_s has only one node, then $N = N_1$, and we take $(N_1, M_i^f) = (N_0, M_i)$. If G_s has more than one node, we select one of its leaves, different from N_1 (this is possible, because a spanning tree with at least two nodes has at least two leaves, and so we are never forced to select N_1). Once such a

[4] A spanning tree is a cycle-free connected graph (V, E'); it can be obtained from G by successive deletion of edges that belong to a cycle.

leaf N_i is chosen, we consider its private subnets one by one. For each private subnet, we execute the occurrence sequences of Proposition 4.6 from the markings $M_1, \ldots, M_n$. After that, we remove the private subnet. We proceed like this with all the private subnets of N_i. We thus obtain systems $(N', M'_1), \ldots, (N', M'_n)$, where N' is minimally covered by $\mathcal{C}' = \mathcal{C} \setminus \{N_i\}$. Moreover, the graph G' corresponding to the minimal cover $\mathcal{C}'$ is the graph obtained from G by removing the node N', and the graph G'_s obtained from G_s by removing the vertex N' is a spanning tree of G'. If G'_s contains more than one node, we iterate the procedure, this time starting from $(N', M'_1), \ldots, (N', M'_n)$ and G'_s.

Since each iteration removes one node from G, the procedure terminates when the spanning tree contains only one node. Since N_1 is never removed, this node is n_1. So the procedure outputs systems $(N_1, M_1^f), \ldots, (N_1, M_n^f)$.

Let $M_{11}, \ldots, M_{1n}$ be the projection of $M_1, \ldots, M_n$ onto the places of N_1. Since Proposition 4.6(2) can be applied each time we remove a private subnet, we have:

(i) $M_i^f \geq M_{1i}$ for $1 \leq i \leq n$, and

(ii) $M_1^f \ldots, M_n^f$ agree on all the invariants of N_1.

By (i), $M_i^f \geq M_s + M_{r_i}$. the result follows from (ii) and Theorem 3.2(i).

(b) No T-component of $\mathcal{C}$ contains both s and t.

Let N_1 be a T-component of $\mathcal{C}$ containing s. We choose a spanning tree G_s of G, and proceed as in (a), iteratively selecting a leaf different from N_1. However, we no longer stop when the spanning tree contains a node, but as soon as t belongs to a private subnet $\widehat{N}$ of some leaf. Notice that this eventually happens, because otherwise the reduction process could continue until only N_1 remains, which contradicts the assumption that no T-component of $\mathcal{C}$ contains both s and t.

Let N' be the net obtained after termination, and let $M_1^f, \ldots, M_n^f$ be the corresponding markings. Further, let $M'_1, \ldots, M'_n$ be the projection of $M_1, \ldots, M_n$ onto the places of N'. By Proposition 4.6(3), $M_i^f \geq M'_i$, and therefore M_i^f marks both s and r_i. Now, by Proposition 4.6, there exist occurrence sequences $\sigma_1, \ldots, \sigma_n$ enabled at $M_1^f, \ldots, M_n^f$, which contain only transitions of $\widehat{T} \setminus \widehat{T}_{in}$, and lead to markings satisfying two conditions:

(i) the projections of $M_1^f, \ldots, M_n^f$ onto the places of $\widehat{N}$ coincide, and

(ii) no transition of $\widehat{T} \setminus \widehat{T}_{in}$ is enabled at $M_1^f, \ldots, M_n^f$.

Since $\widehat{N}$ is a T-net, (i) and (ii) can only hold if all of $\sigma_1, \ldots, \sigma_n$ contain the transition t. Since s remains marked along the execution of these sequences, some reachable marking marks simultaneously s and all the input places of t.

■

Proposition 4.9 leads to our main result:

Theorem 4.10 *Concurrency Theorem for free-choice systems*

The relations $\parallel$ and $\parallel^A$ coincide for live and bounded free-choice systems.

Proof:

We have $\parallel \subseteq \parallel^A$ by Theorem 3.2(i). We prove that the $\parallel$ relation of a live and bounded free-choice system (N, M_0) satisfies the three conditions of Definition 3.1. Since $\parallel^A$ is the smallest symmetric relation satisfying these conditions, we have $\parallel^A \subseteq \parallel$, which finishes the proof. Condition (i) follows easily from the definition of $\parallel$. Condition (ii) is a direct consequence of the liveness of (N, M_0). Condition (iii) follows immediately from Proposition 4.9. ■

5 Computing the structural concurrency relation

In [6], the first author presented a $O(n^5)$ algorithm for the computation of $\parallel^A$ in an arbitrary net system, where n is the number of places and transitions of the net. In this paper we show that $\parallel^A$ can be computed in $O(n^4)$ time, and in $O(n^3)$ time for free-choice systems.

Algorithm 5.1

Input: A live system (N, M_0), where $N = (S, T, F)$.
Output: $R \subseteq X \times X$.

begin
  $R := \{(s, s') \mid M_0 \geq M_s + M_{s'}\} \cup \bigcup_{t \in T} t^\bullet \times t^\bullet$;
  $E := R \cap (X \times S)$;
  while $E \neq \emptyset$ **do**
    choose $(x, s) \in E$; $E := E \setminus \{(x, s)\}$;
    for every $t \in s^\bullet$ **do**
      if $\{x\} \times {}^\bullet t \subseteq R$ **then**
        $E := E \cup ((\{x\} \times t^\bullet) \setminus R)$;
        $R := R \cup \{(x, t)\} \cup (\{x\} \times t^\bullet)$
      endif
    endfor
  endwhile
end

Proposition 5.2

Algorithm 5.1 terminates, and after termination $R = \parallel^A$.

Proof:

Observe that $E \subseteq R$ is an invariant of the while loop and holds initially. Therefore, each execution of the while loop removes from E an element of $E \cap R$. This element is never added to E again. So the algorithm terminates.

Let Q be the value of R after termination. We prove:

(1) $Q \subseteq \|^A$.

We first prove that $R \subseteq \|^A$ holds initially. We have $\{(s, s') \mid M_0 \geq M_s + M_{s'}\} \subseteq \|^A$ by definition. Moreover, since (N, M_0) is live, for every transition t there is a reachable marking which simultaneously marks every output place of t. Therefore, $\bigcup_{t \in T} t^\bullet \times t^\bullet \subseteq \|$, and since $\| \subseteq \|^A$, we have $\bigcup_{t \in T} t^\bullet \times t^\bullet \subseteq \|^A$. So initially $R \subseteq \|^A$.

Moreover, it follows easily from the definition of $\|^A$ that $R \subseteq \|^A$ is an invariant of the while loop. So we have $Q \subseteq \|^A$.

(2) Q satisfies the three conditions of Definition 3.1.

Condition (i) and (ii) follow immediately from the initialisation of R. For condition (iii), let $x \in X$ and $t \in T$. We have to prove:

$$\{x\} \times {}^\bullet t \subseteq Q \implies (x, t) \in \|^A \ \wedge \ \{x\} \times t^\bullet \subseteq Q$$

If $\{x\} \times {}^\bullet t$ is not included in Q, we are done. So assume $\{x\} \times {}^\bullet t \subseteq Q$.

Let (x, s) be the last element of $\{x\} \times {}^\bullet t$ which is removed from E during the execution of the algorithm. As we have seen above, (x, s) is never added to E again.

Assume that immediately after (x, s) is removed from E, we have $(x, s') \notin R$ for some $s' \in {}^\bullet t$. We prove that (x, s') is never added to R later on. Every new element added to R is also added to E, and every element of E is removed before termination. Therefore, if (x, s') were added to R it would later be removed from E, contradicting the definition of (x, s).

Since $\{x\} \times {}^\bullet t \subseteq R$ and no element of $\{x\} \times {}^\bullet t$ is added to R after (x, s) is removed from E, we already have $\{x\} \times {}^\bullet t \subseteq R$ immediately after (x, s) is removed from E. Then, the next execution of the for loop adds $(\{x\} \times t^\bullet)$ to Q. So $(\{x\} \times t^\bullet) \subseteq Q$ after termination.

$Q = \|^A$ follows from (1), (2) and the minimality of $\|^A$. ■

We calculate the complexity of the algorithm when the subsets $X \times X$ (in particular, the incidence relations of the net) are encoded as bidimensional arrays $X \times X \to \{0, 1\}$. In this case, the algorithm needs $O(|X|^2)$ space, and the following operations take constant time for every $(x, y) \in X \times X$ and $R \subseteq X \times X$:

- determine if $x \in {}^\bullet y$ ($x \in y^\bullet$);
- determine if $(x, y) \in R$;
- add (x, y) to R;

- remove (x, y) from R.

The initialisation of Q, E and N takes $O(|S|^2 \cdot |T|)$ time. The **while** loop is executed at most $O(|S| \cdot |X|)$ times, because each iteration removes one element from E which is never added to it again. One iteration takes $O(|S| \cdot |T|)$ time ($O(|T|)$ iterations of the **for** loop requiring $O(|S|)$ time each). So the algorithm runs in $O(|S|^2 \cdot |T| \cdot |X|)$ time.

It is possible to give a faster algorithm for free-choice systems, because they satisfy the following property: ${}^\bullet t_1 = {}^\bullet t_2$ for every two output transitions t_1, t_2 of a place s. In this case, the condition of the if instruction in the algorithm above holds either for all or for none of the transitions $t \in s^\bullet$. So, instead of checking the condition for each transition of $s^\bullet$, we may just check it for one of them. If the condition holds, we may then immediately add to R the set $\{x\} \times \bigcup_{t \in s^\bullet} t^\bullet = \{x\} \times (s^\bullet)^\bullet$. If we precompute the set $\{(s, s') \mid s' \in (s^\bullet)^\bullet\}$ in the initialisation step, which can be done in $O(|S|^2 \times |T|)$, this assignment requires only $O(|S|)$ time.

We get the following algorithm:

Algorithm 5.3

Input: A live free-choice system (N, M_0), where $N = (S, T, F)$.
Output: $R \subseteq X \times X$.

begin
  $R := \{(s, s') \mid M_0 \geq M_s + M_{s'}\} \cup \bigcup_{t \in T} t^\bullet \times t^\bullet$;
  $A := \{(s, s') \mid s' \in (s^\bullet)^\bullet\}$;
  $E := R \cap (X \times S)$;
  while $E \neq \emptyset$ **do**
    choose $(x, s) \in E$; $E := E \setminus \{(x, s)\}$;
    choose $t \in s^\bullet$;
    if $\{x\} \times {}^\bullet t \subseteq R$ **then**
      $E := E \cup (\{(x, s') \mid (s, s') \in A\} \setminus R)$;
      $R := R \cup \{(x, s') \mid (s, s') \in A\}$
    endif
  endwhile
end

An iteration of the while loop requires only $O(|X|)$ time, and not $O(|S| \cdot |T|)$, as was the case in Algorithm 5.1. Since the initialisation step can still be executed in $O(|S|^2 \times |T|)$, Algorithm 5.3 runs in $O(|S| \cdot |X|^2)$ time.

6 Conclusions

We have presented an $O(n^3)$ algorithm for the computation of the concurrency relation of live and bounded free-choice systems, where n is the number of nodes of the net. Our work was motivated by the interesting applications of the concurrency relation to the design

and verification of asynchronous circuits. Our algorithm can be used to detect undesirable race conditions in free-choice Signal Transition Diagrams (STGs) modelling specifications of speed-independent circuits. It can also be used as subroutine in the algorithm for checking the Unique State Coding property presented in [10, 11].

Pastor and Cortadella give in [11] an $O(n^3)$ algorithm for the computation of the concurrency relation. The algorithm is based on some plausible but so far unproved properties of free-choice systems. It also has some restrictions: it only works for 1-bounded free-choice STGs without linearly dependent places, in which the initial marking is a home marking (i.e., can be reached from any other reachable marking). We have managed to solve these problems: our algorithm has a formal proof, is simpler, more general, and has the same complexity.

Our paper adds one more to the list of results on the concurrency problem, i.e., the problem of deciding if two given transitions of a Petri net are concurrently enabled at some reachable marking. The problem is EXPSPACE-hard for arbitrary net systems, and PSPACE-complete for 1-bounded systems [2]. It has been shown to be polynomial for live T-systems [6], 1-bounded conflict-free systems [13, 5] (although the algorithm of [5] can be easily generalised to the n-bounded case), and now for live and bounded free-choice systems.

Our algorithm also can be used to solve the 1-boundedness problem: decide if a given live and bounded free-choice system is 1-bounded. It follows easily from the definition of the concurrency relation that a net system is 1-safe iff its concurrency relation is irreflexive. So the 1-boundedness problem can be solved in $O(n^3)$ as well. This improves the complexity of earlier algorithms based on linear programming.

Acknowledgements

Many thanks to Jordi Cortadella and Enric Pastor for helpful discussions, and for drawing our attention to the concurrency problem for STGs.

References

[1] E. Best and C. Fernández. Nonsequential Processes – A Petri Net View. *EATCS Monographs on Theoretical Computer Science*, 13, 1988.

[2] A. Cheng, J. Esparza, and J. Palsberg. Complexity Results for 1-safe Nets. *Theoretical Computer Science*, 147:117–136, 1995.

[3] T.A. Chu. *Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications.* Phd thesis, MIT, 1987.

[4] J. Desel and J. Esparza. *Free-choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1995.

[5] J. Esparza. A Solution to the Covering Problem for 1-Bounded Conflict-Free Petri Nets Using Linear Programming. *Information Processing Letters*, 41:313–319, 1992.

[6] A. V. Kovalyov. Concurrency Relations and the Safety Problem for Petri Nets. In *Proceedings of the 13th International Conference on Application and Theory of Petri Nets*, number 616 in Lecture Notes in Computer Science, 1992.

[7] A. V. Kovalyov. Good behaviour in extended free choice nets. Technical Report FBI-HH-B-176/95, Fachbereich Informatik, Universität Hamburg, 1995.

[8] L. Lavagno and A. Sangiovanni-Vincentelli. *Algorithms for Synthesis and Testing of Asynchronous Circuits*. Kluwer Academic Publishers, 1993.

[9] T.H.Y. Meng, editor. *Synchronization Design for Digital Systems*. Kluwer Academic Publishers, 1991.

[10] E. Pastor and J. Cortadella. An efficient unique state coding algorithm for signal transition graphs. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 174–177, 1993.

[11] E. Pastor and J. Cortadella. An efficient unique state coding algorithm for signal transition graphs. Technical Report RR-93/13, Department of Computer Architecture, Universidad Politècnica de Catalunya, 1993.

[12] M. Tiusanen. Some Unsolved Problems in Modelling Self-Timed Circuits Using Petri Nets. *EATCS Bulletin*, 36:152–160, 1988.

[13] H. C. Yen. A Polynomial Time Algorithm to Decide Pairwise Concurrency of Transitions for 1-Bounded Conflict Free Petri Nets. *Information Processing Letters*, 38:71–76, 1991.

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

342/1/90 A	Robert Gold, Walter Vogler: Quality Criteria for Partial Order Semantics of Place/Transition-Nets, Januar 1990
342/2/90 A	Reinhard Fößmeier: Die Rolle der Lastverteilung bei der numerischen Parallelprogrammierung, Februar 1990
342/3/90 A	Klaus-Jörn Lange, Peter Rossmanith: Two Results on Unambiguous Circuits, Februar 1990
342/4/90 A	Michael Griebel: Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen Transformations-Mehrgitter-Methode
342/5/90 A	Reinhold Letz, Johann Schumann, Stephan Bayerl, Wolfgang Bibel: SETHEO: A High-Performance Theorem Prover
342/6/90 A	Johann Schumann, Reinhold Letz: PARTHEO: A High Performance Parallel Theorem Prover
342/7/90 A	Johann Schumann, Norbert Trapp, Martin van der Koelen: SETHEO/PARTHEO Users Manual
342/8/90 A	Christian Suttner, Wolfgang Ertel: Using Connectionist Networks for Guiding the Search of a Theorem Prover
342/9/90 A	Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Jaroslav Kremenek, Robert Lindhof, Thomas Ludwig, Peter Luksch, Thomas Treml: TOPSYS, Tools for Parallel Systems (Artikelsammlung)
342/10/90 A	Walter Vogler: Bisimulation and Action Refinement
342/11/90 A	Jörg Desel, Javier Esparza: Reachability in Reversible Free- Choice Systems
342/12/90 A	Rob van Glabbeek, Ursula Goltz: Equivalences and Refinement
342/13/90 A	Rob van Glabbeek: The Linear Time - Branching Time Spectrum
342/14/90 A	Johannes Bauer, Thomas Bemmerl, Thomas Treml: Leistungsanalyse von verteilten Beobachtungs- und Bewertungswerkzeugen
342/15/90 A	Peter Rossmanith: The Owner Concept for PRAMs
342/16/90 A	G. Böckle, S. Trosch: A Simulator for VLIW-Architectures
342/17/90 A	P. Slavkovsky, U. Rüde: Schnellere Berechnung klassischer Matrix-Multiplikationen
342/18/90 A	Christoph Zenger: SPARSE GRIDS
342/19/90 A	Michael Griebel, Michael Schneider, Christoph Zenger: A combination technique for the solution of sparse grid problems

Reihe A

342/20/90 A Michael Griebel: A Parallelizable and Vectorizable Multi- Level-Algorithm on Sparse Grids

342/21/90 A V. Diekert, E. Ochmanski, K. Reinhardt: On confluent semi-commutations-decidability and complexity results

342/22/90 A Manfred Broy, Claus Dendorfer: Functional Modelling of Operating System Structures by Timed Higher Order Stream Processing Functions

342/23/90 A Rob van Glabbeek, Ursula Goltz: A Deadlock-sensitive Congruence for Action Refinement

342/24/90 A Manfred Broy: On the Design and Verification of a Simple Distributed Spanning Tree Algorithm

342/25/90 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Peter Luksch, Roland Wismüller: TOPSYS - Tools for Parallel Systems (User's Overview and User's Manuals)

342/26/90 A Thomas Bemmerl, Arndt Bode, Thomas Ludwig, Stefan Tritscher: MMK - Multiprocessor Multitasking Kernel (User's Guide and User's Reference Manual)

342/27/90 A Wolfgang Ertel: Random Competition: A Simple, but Efficient Method for Parallelizing Inference Systems

342/28/90 A Rob van Glabbeek, Frits Vaandrager: Modular Specification of Process Algebras

342/29/90 A Rob van Glabbeek, Peter Weijland: Branching Time and Abstraction in Bisimulation Semantics

342/30/90 A Michael Griebel: Parallel Multigrid Methods on Sparse Grids

342/31/90 A Rolf Niedermeier, Peter Rossmanith: Unambiguous Simulations of Auxiliary Pushdown Automata and Circuits

342/32/90 A Inga Niepel, Peter Rossmanith: Uniform Circuits and Exclusive Read PRAMs

342/33/90 A Dr. Hermann Hellwagner: A Survey of Virtually Shared Memory Schemes

342/1/91 A Walter Vogler: Is Partial Order Semantics Necessary for Action Refinement?

342/2/91 A Manfred Broy, Frank Dederichs, Claus Dendorfer, Rainer Weber: Characterizing the Behaviour of Reactive Systems by Trace Sets

342/3/91 A Ulrich Furbach, Christian Suttner, Bertram Fronhöfer: Massively Parallel Inference Systems

342/4/91 A Rudolf Bayer: Non-deterministic Computing, Transactions and Recursive Atomicity

342/5/91 A Robert Gold: Dataflow semantics for Petri nets

342/6/91 A A. Heise; C. Dimitrovici: Transformation und Komposition von P/T-Netzen unter Erhaltung wesentlicher Eigenschaften

342/7/91 A Walter Vogler: Asynchronous Communication of Petri Nets and the Refinement of Transitions

Reihe A

342/8/91 A Walter Vogler: Generalized OM-Bisimulation

342/9/91 A Christoph Zenger, Klaus Hallatschek: Fouriertransformation auf dünnen Gittern mit hierarchischen Basen

342/10/91 A Erwin Loibl, Hans Obermaier, Markus Pawlowski: Towards Parallelism in a Relational Database System

342/11/91 A Michael Werner: Implementierung von Algorithmen zur Kompaktifizierung von Programmen für VLIW-Architekturen

342/12/91 A Reiner Müller: Implementierung von Algorithmen zur Optimierung von Schleifen mit Hilfe von Software-Pipelining Techniken

342/13/91 A Sally Baker, Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Udo Graf, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Rainer Knödlseder, Jaroslav Kremenek, Siegfried Langenbuch, Robert Lindhof, Thomas Ludwig, Peter Luksch, Roy Milner, Bernhard Ries, Thomas Treml: TOPSYS - Tools for Parallel Systems (Artikelsammlung); 2., erweiterte Auflage

342/14/91 A Michael Griebel: The combination technique for the sparse grid solution of PDE's on multiprocessor machines

342/15/91 A Thomas F. Gritzner, Manfred Broy: A Link Between Process Algebras and Abstract Relation Algebras?

342/16/91 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Thomas Treml, Roland Wismüller: The Design and Implementation of TOPSYS

342/17/91 A Ulrich Furbach: Answers for disjunctive logic programs

342/18/91 A Ulrich Furbach: Splitting as a source of parallelism in disjunctive logic programs

342/19/91 A Gerhard W. Zumbusch: Adaptive parallele Multilevel-Methoden zur Lösung elliptischer Randwertprobleme

342/20/91 A M. Jobmann, J. Schumann: Modelling and Performance Analysis of a Parallel Theorem Prover

342/21/91 A Hans-Joachim Bungartz: An Adaptive Poisson Solver Using Hierarchical Bases and Sparse Grids

342/22/91 A Wolfgang Ertel, Theodor Gemenis, Johann M. Ph. Schumann, Christian B. Suttner, Rainer Weber, Zongyan Qiu: Formalisms and Languages for Specifying Parallel Inference Systems

342/23/91 A Astrid Kiehn: Local and Global Causes

342/24/91 A Johann M.Ph. Schumann: Parallelization of Inference Systems by using an Abstract Machine

342/25/91 A Eike Jessen: Speedup Analysis by Hierarchical Load Decomposition

342/26/91 A Thomas F. Gritzner: A Simple Toy Example of a Distributed System: On the Design of a Connecting Switch

342/27/91 A Thomas Schnekenburger, Andreas Weininger, Michael Friedrich: Introduction to the Parallel and Distributed Programming Language ParMod-C

Reihe A

342/28/91 A	Claus Dendorfer: Funktionale Modellierung eines Postsystems
342/29/91 A	Michael Griebel: Multilevel algorithms considered as iterative methods on indefinite systems
342/30/91 A	W. Reisig: Parallel Composition of Liveness
342/31/91 A	Thomas Bemmerl, Christian Kasperbauer, Martin Mairandres, Bernhard Ries: Programming Tools for Distributed Multiprocessor Computing Environments
342/32/91 A	Frank Leßke: On constructive specifications of abstract data types using temporal logic
342/1/92 A	L. Kanal, C.B. Suttner (Editors): Informal Proceedings of the Workshop on Parallel Processing for AI
342/2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS
342/2-2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS - Revised Version (erschienen im Januar 1993)
342/3/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems
342/4/92 A	Claus Dendorfer, Rainer Weber: Development and Implementation of a Communication Protocol - An Exercise in FOCUS
342/5/92 A	Michael Friedrich: Sprachmittel und Werkzeuge zur Unterstüt- zung paralleler und verteilter Programmierung
342/6/92 A	Thomas F. Gritzner: The Action Graph Model as a Link between Abstract Relation Algebras and Process-Algebraic Specifications
342/7/92 A	Sergei Gorlatch: Parallel Program Development for a Recursive Numerical Algorithm: a Case Study
342/8/92 A	Henning Spruth, Georg Sigl, Frank Johannes: Parallel Algorithms for Slicing Based Final Placement
342/9/92 A	Herbert Bauer, Christian Sporrer, Thomas Krodel: On Distributed Logic Simulation Using Time Warp
342/10/92 A	H. Bungartz, M. Griebel, U. Rüde: Extrapolation, Combination and Sparse Grid Techniques for Elliptic Boundary Value Problems
342/11/92 A	M. Griebel, W. Huber, U. Rüde, T. Störtkuhl: The Combination Technique for Parallel Sparse-Grid-Preconditioning and -Solution of PDEs on Multiprocessor Machines and Workstation Networks
342/12/92 A	Rolf Niedermeier, Peter Rossmanith: Optimal Parallel Algorithms for Computing Recursively Defined Functions
342/13/92 A	Rainer Weber: Eine Methodik für die formale Anforderungsspezifkation verteilter Systeme

Reihe A

342/14/92 A Michael Griebel: Grid– and point–oriented multilevel algorithms

342/15/92 A M. Griebel, C. Zenger, S. Zimmer: Improved multilevel algorithms for full and sparse grid problems

342/16/92 A J. Desel, D. Gomm, E. Kindler, B. Paech, R. Walter: Bausteine eines kompositionalen Beweiskalküls für netzmodellierte Systeme

342/17/92 A Frank Dederichs: Transformation verteilter Systeme: Von applikativen zu prozeduralen Darstellungen

342/18/92 A Andreas Listl, Markus Pawlowski: Parallel Cache Management of a RDBMS

342/19/92 A Erwin Loibl, Markus Pawlowski, Christian Roth: PART: A Parallel Relational Toolbox as Basis for the Optimization and Interpretation of Parallel Queries

342/20/92 A Jörg Desel, Wolfgang Reisig: The Synthesis Problem of Petri Nets

342/21/92 A Robert Balder, Christoph Zenger: The d-dimensional Helmholtz equation on sparse Grids

342/22/92 A Ilko Michler: Neuronale Netzwerk-Paradigmen zum Erlernen von Heuristiken

342/23/92 A Wolfgang Reisig: Elements of a Temporal Logic. Coping with Concurrency

342/24/92 A T. Störtkuhl, Chr. Zenger, S. Zimmer: An asymptotic solution for the singularity at the angular point of the lid driven cavity

342/25/92 A Ekkart Kindler: Invariants, Compositionality and Substitution

342/26/92 A Thomas Bonk, Ulrich Rüde: Performance Analysis and Optimization of Numerically Intensive Programs

342/1/93 A M. Griebel, V. Thurner: The Efficient Solution of Fluid Dynamics Problems by the Combination Technique

342/2/93 A Ketil Stølen, Frank Dederichs, Rainer Weber: Assumption / Commitment Rules for Networks of Asynchronously Communicating Agents

342/3/93 A Thomas Schnekenburger: A Definition of Efficiency of Parallel Programs in Multi-Tasking Environments

342/4/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: A Proof of Convergence for the Combination Technique for the Laplace Equation Using Tools of Symbolic Computation

342/5/93 A Manfred Kunde, Rolf Niedermeier, Peter Rossmanith: Faster Sorting and Routing on Grids with Diagonals

342/6/93 A Michael Griebel, Peter Oswald: Remarks on the Abstract Theory of Additive and Multiplicative Schwarz Algorithms

342/7/93 A Christian Sporrer, Herbert Bauer: Corolla Partitioning for Distributed Logic Simulation of VLSI Circuits

342/8/93 A Herbert Bauer, Christian Sporrer: Reducing Rollback Overhead in Time-Warp Based Distributed Simulation with Optimized Incremental State Saving

Reihe A

342/9/93 A Peter Slavkovsky: The Visibility Problem for Single-Valued Surface (z = f(x,y)): The Analysis and the Parallelization of Algorithms

342/10/93 A Ulrich Rüde: Multilevel, Extrapolation, and Sparse Grid Methods

342/11/93 A Hans Regler, Ulrich Rüde: Layout Optimization with Algebraic Multigrid Methods

342/12/93 A Dieter Barnard, Angelika Mader: Model Checking for the Modal Mu-Calculus using Gauß Elimination

342/13/93 A Christoph Pflaum, Ulrich Rüde: Gauß' Adaptive Relaxation for the Multilevel Solution of Partial Differential Equations on Sparse Grids

342/14/93 A Christoph Pflaum: Convergence of the Combination Technique for the Finite Element Solution of Poisson's Equation

342/15/93 A Michael Luby, Wolfgang Ertel: Optimal Parallelization of Las Vegas Algorithms

342/16/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: Pointwise Convergence of the Combination Technique for Laplace's Equation

342/17/93 A Georg Stellner, Matthias Schumann, Stefan Lamberts, Thomas Ludwig, Arndt Bode, Martin Kiehl und Rainer Mehlhorn: Developing Multicomputer Applications on Networks of Workstations Using NXLib

342/18/93 A Max Fuchs, Ketil Stølen: Development of a Distributed Min/Max Component

342/19/93 A Johann K. Obermaier: Recovery and Transaction Management in Write-optimized Database Systems

342/20/93 A Sergej Gorlatch: Deriving Efficient Parallel Programs by Systemating Coarsing Specification Parallelism

342/01/94 A Reiner Hüttl, Michael Schneider: Parallel Adaptive Numerical Simulation

342/02/94 A Henning Spruth, Frank Johannes: Parallel Routing of VLSI Circuits Based on Net Independency

342/03/94 A Henning Spruth, Frank Johannes, Kurt Antreich: PHIroute: A Parallel Hierarchical Sea-of-Gates Router

342/04/94 A Martin Kiehl, Rainer Mehlhorn, Matthias Schumann: Parallel Multiple Shooting for Optimal Control Problems Under NX/2

342/05/94 A Christian Suttner, Christoph Goller, Peter Krauss, Klaus-Jörn Lange, Ludwig Thomas, Thomas Schnekenburger: Heuristic Optimization of Parallel Computations

342/06/94 A Andreas Listl: Using Subpages for Cache Coherency Control in Parallel Database Systems

342/07/94 A Manfred Broy, Ketil Stølen: Specification and Refinement of Finite Dataflow Networks - a Relational Approach

Reihe A

342/08/94 A Katharina Spies: Funktionale Spezifikation eines Kommunikationsprotokolls

342/09/94 A Peter A. Krauss: Applying a New Search Space Partitioning Method to Parallel Test Generation for Sequential Circuits

342/10/94 A Manfred Broy: A Functional Rephrasing of the Assumption/Commitment Specification Style

342/11/94 A Eckhardt Holz, Ketil Stølen: An Attempt to Embed a Restricted Version of SDL as a Target Language in Focus

342/12/94 A Christoph Pflaum: A Multi-Level-Algorithm for the Finite-Element-Solution of General Second Order Elliptic Differential Equations on Adaptive Sparse Grids

342/13/94 A Manfred Broy, Max Fuchs, Thomas F. Gritzner, Bernhard Schätz, Katharina Spies, Ketil Stølen: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems

342/14/94 A Maximilian Fuchs: Technologieabhängigkeit von Spezifikationen digitaler Hardware

342/15/94 A M. Griebel, P. Oswald: Tensor Product Type Subspace Splittings And Multilevel Iterative Methods For Anisotropic Problems

342/16/94 A Gheorghe Ştefănescu: Algebra of Flownomials

342/17/94 A Ketil Stølen: A Refinement Relation Supporting the Transition from Unbounded to Bounded Communication Buffers

342/18/94 A Michael Griebel, Tilman Neuhoeffer: A Domain-Oriented Multilevel Algorithm-Implementation and Parallelization

342/19/94 A Michael Griebel, Walter Huber: Turbulence Simulation on Sparse Grids Using the Combination Method

342/20/94 A Johann Schumann: Using the Theorem Prover SETHEO for verifying the development of a Communication Protocol in FOCUS - A Case Study -

342/01/95 A Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids

342/02/95 A Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law

342/03/95 A Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space

342/04/95 A Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes

342/05/95 A Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems

342/06/95 A Maximilian Fuchs: Formal Design of a Model-N Counter

342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology

Reihe A

342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies

342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness

342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design

342/11/95 A Thomas Schnekenburger: The ALDY Load Distribution System

342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm

342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming

342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks

342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B Wolfgang Reisig: Petri Nets and Algebraic Specifications

342/2/90 B Jörg Desel: On Abstraction of Nets

342/3/90 B Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems

342/4/90 B Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme

342/1/91 B Barbara Paech1: Concurrency as a Modality

342/2/91 B Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier- Toolbox -Anwenderbeschreibung

342/3/91 B Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen

342/4/91 B Werner Pohlmann: A Limitation of Distributed Simulation Methods

342/5/91 B Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis

342/6/91 B Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme

342/7/91 B W. Reisig: Concurrent Temporal Logic

342/1/92 B Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support
Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support

342/2/92 B Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen

342/1/93 B Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung

342/2/93 B Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick

342/1/94 B Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS