Theoretical Computer Science 124 (1994) 169–179
Elsevier

Note

Finite Petri nets as models for recursive causal behaviour*

Ursula Goltz

Gesellschaft für Mathematik and Datenverarbeitung, St. Augustin, Germany

Arend Rensink

Tele-Informatics and Open Systems, University of Twente, Enschede, The Netherlands

Communicated by M. Wirsing
Received December 1991
Revised January 1993

Abstract

Goltz, U., A. Rensink, Finite Petri nets as models for recursive causal behaviour, Theoretical Computer Science 124 (1994) 169–179.

Goltz (1988) discussed whether or not there exist finite Petri nets (with unbounded capacities) modelling the causal behaviour of certain recursive CCS terms. As a representative example, the following term is considered:

$$B=(a.nil \mid b.B)+c.nil.$$

We will show that the answer depends on the chosen notion of behaviour. It was already known that the interleaving behaviour and the branching structure of terms as B can be modelled as long as causality is not taken into account. We now show that also the causal behaviour of B can be modelled as long as the branching structure is not taken into account. However, it is not possible to represent both causal dependencies and the behaviour with respect to choices between alternatives in a finite net. We prove that there exists no finite Petri net modelling B with respect to both pomset trace equivalence and failure equivalence.

Correspondence to: U. Goltz, Institut für Informatik, Universität Hildesheim, Postfach 101363, 31113 Hildesheim, Germany. Email: goltz@informatik.uni-hildesheim.de.

* The work presented here was supported partly by the Esprit Basic Research Action 3148 (DEMON), the Sonderforschungsbereich 182 of the University of Erlangen, and HOOP funding by the Dutch Ministry of Education.

SSDI 0304-3975(93)E0083-G

1. Introduction

When modelling concurrent systems as Petri nets, it is sometimes possible to have finite representations for infinite behaviours, by playing the token game on finite net structures. In particular, when generating net representations using composition operations like in process algebras, some recursive processes may be represented as finite nets where transitions are labelled by action names; see [4, 5, 12]. For instance, the constructions of [4] allow one to build certain processes with unboundedly growing parallelism as finite P/T systems with unbounded capacity of places.

The constructions in [4] work only for behaviours described by CCS terms without restriction or relabelling and with only guarded choice (which forbids initial parallelism in choice components). Taubner [12] shows that adding restriction to the process algebra from [4] already yields Turing power, and so in general there is no finite net representation in this case. Up to now it has not been considered formally to what extent the restriction to guarded choice is necessary in order to make finite net representations possible. This question will be addressed here.

The following simple term with unguarded choice was discussed as a representative example in [4]:

$$B = (a.nil \mid b.B) + c.nil.$$

It was shown that applying the constructions presented there to this term would not yield the desired behaviour. It was however left open whether any finite net correctly modelling B exists at all.

We will show that the answer to this question depends on the chosen notion of correctness, or – putting it more formally – on the chosen equivalence notion. Two aspects need to be considered.

(1) The aim of giving a Petri net representation is to capture *causal dependencies* between action occurrences. This can be done by adopting a *pomset semantics* where a system is equated with the set of its possible "executions" modelled as partially ordered multisets of actions, where the partial ordering represents the causal dependencies.

(2) The *branching structure* of a system refers to the places where the choices between alternative executions take place. The branching structure is captured in greater or lesser detail by various kinds of interleaving semantics, an account of which is given in [3] as the "linear time-branching time spectrum". The most prominent in this spectrum are testing or failure semantics and bisimulation semantics.

It has been shown before that the behaviour of B may be modelled correctly as a finite net with respect to interleaving bisimulation, and even with respect to interleaving of steps (collecting in one step actions that may happen in parallel) [4, 11]. The question remained whether the causal behaviour of B may be modelled correctly as well, and similarly if causality and branching structure both have to be preserved.

We show here that one may still find a finite net representation which gives the expected partially ordered executions, even when distinguishing terminated behaviours. However, we will prove that it is not possible to find a finite net that models both causality and branching structure correctly, when requiring at least failure (or testing) equivalence as a criterion for the preservation of the branching structure.

2. Preliminaries

We start by introducing some basic notions concerning Petri nets. For this we need multisets. $\mathbb{N}^X$ will denote the set of multisets over X, and $\mathbb{N}_+^X$ the set of nonempty multisets. We use $\leqslant$ to denote "multisubset", and $+$ and $-$ to denote multiset addition and subtraction. "Membership" of a multiset, denoted by $\in$, will be interpreted as positive multiplicity of the respective element. Finally, a set $Y \subseteq X$ appearing in a multiset equation should be interpreted as the multiset that assigns multiplicity 1 to the members of Y and 0 to all elements in $X - Y$.

Definition 2.1. Let A be an alphabet.

An *A-labelled unmarked finite P/T net* N is a pair (S, T), where

- S is a finite set of *places*, ranged over by s;
- $T \subseteq \mathbb{N}_+^S \times A \times \mathbb{N}_+^S$ is a finite set of *transitions*, ranged over by t.

A *marking* of N is a multiset $M \in \mathbb{N}^S$. A *marked net* is a 3-tuple $(S, T; M_0)$, where

- (S, T) is an unmarked net;
- $M_0 \in \mathbb{N}^S$ is the *initial marking*.

We will often write S_N, T_N for the sets of places and transitions of a net N. We also use ${}^\bullet t$ and $t^\bullet$ to denote the multisets of places forming, respectively, the first and third component of a transition t (the *preplaces* and *postplaces* of t together with the corresponding arc weights, respectively). $l_N(t)$ denotes the second component (the *label*) of t. Hence $t = ({}^\bullet t, l_N(t), t^\bullet)$ for all $t \in T_N$. We will refer to the resulting function $l_N : T_N \to A$ as the *labelling* of N.

We will often use *multisets of transitions*. The notions of preplaces and postplaces can be extended to such multisets in a straightforward way, with notations ${}^\bullet G$ and $G^\bullet$, respectively, where $G \in \mathbb{N}^T$ is a multiset of transitions.

We can derive the *flow relation* of a net N as in the usual definition of nets: it is the relation $F_N \subseteq (S_N \times T_N) \cup (T_N \times S_N)$ defined by

$$F_N =_{\text{def}} \{(s, t) \in S_N \times T_N \mid s \in {}^\bullet t\} \cup \{(t, s) \in T_N \times S_N \mid s \in t^\bullet\}.$$

We will simply use the term *net* for the marked P/T nets defined above.

Next we define the dynamic behaviour of a marked net. A transition $t \in T$ is *enabled* in a marking M, denoted $M[t\rangle$, if ${}^\bullet t \leqslant M$. We also write $M[t\rangle M'$ if $M[t\rangle$ and $M' = M - {}^\bullet t + t^\bullet$.

The causal behaviour of a net is usually defined in terms of its *processes*. It is sufficient for us to consider finite executions, so we consider only finite processes here.

We will not define processes formally; instead we show the general form that they take, and we show that for the purposes of this paper we can do with a slightly simpler concept, viz. that of *process words*; see below. A process of N is a tuple (K, p) consisting of an unmarked net K with unweighted arcs, unbranched places and a cycle-free flow relation (a so-called *causal net*), and a mapping $p: K \to N$ satisfying a number of constraints. In terms of Definition 2.1, K is labelled by the set T_N of transitions of N, and this labelling coincides with the transition part of p:

$$\forall t \in T_K . p(t) = l_K(t).$$

As mentioned above, in our treatment of the behaviour of nets, instead of processes we use the derived notion of *process words*, which are essentially pomsets[1] labelled over the transitions of the net (called *abstract processes* in [10]). We first define some necessary concepts.

Definition 2.2. Let A be an arbitrary alphabet.

A *partial A-word* is a structure $[E, \leqslant, l]$, where

- E is an arbitrary set of node names;
- $\leqslant \subseteq E \times E$ is a partial ordering relation over E;
- $l: E \to A$ is a labelling function;
- $[\cdot]$ indicates the isomorphism class of $(E, \leqslant, l)$.

The set of partial A-words is denoted $W(A)$.

We will use w or α to range over $W(A)$. The following defines some auxiliary notions about partial words.

Definition 2.3. Let A be an arbitrary alphabet.

- If $w = [E, \leqslant, l] \in W(A)$ is a partial A-word, then the *label multiset of w* is the multiset $l^*(w) \in \mathbb{N}^A$ defined by[2]

$$l^*(w) =_{\text{def}} \sum_{e \in E} \{l(e)\}.$$

Note that the label multiset is independent of the chosen representative $(E, \leqslant, l)$.

- If $w_1, w_2 \in W(A)$ are two partial A-words, then w_1 is called a *prefix of w_2* if there are representatives $w_i = [E_i, \leqslant_i, l_i]$ such that

$$E_1 \subseteq E_2 \wedge \leqslant_1 = \leqslant_2 \cap (E_2 \times E_1) \wedge l_1 = l_2 \lceil E_1.$$

Note that this guarantees that E_1 is left-closed with respect to $\leqslant_2$.

- A subset $W \subseteq W(A)$ is said to be *prefix closed* if it satisfies

$$\forall w_1 \in W . \forall w_2 \in W(A) . w_2 \text{ is a prefix of } w_1 \Rightarrow w_2 \in W.$$

[1] Isomorphism classes of labelled partially ordered sets.

[2] We need the "sum" notation to get a multiset; the construction $\{l(e) \mid e \in E\}$ yields just the *set* of labels. An alternative multiset construction is the function $a \mapsto |\{e \in E \mid l(e) = a\}|$ for all $a \in A$.

Now we can define the *process words* and *pomset traces* of a net, which are, respectively, partial T_N-words and partial A-words derived from the net (where A is the alphabet of the net).

Definition 2.4. Let N be an A-labelled net with set of transitions T.
The *process words* of N are partial T-words $w \in W(T)$ derived from the processes of N such that for a given process (K, p) the corresponding process word is $[T_K, \leqslant, l_K]$, where

- T_K is the set of transitions of K;
- $\leqslant = F_K^* \lceil T_K$, where F_K^* is the reflexive transitive closure of the flow relation of K;
- $l_K : T_K \to T$ is the labelling function of K.

The *pomset traces* of N are partial A-words $\alpha \in W(A)$ derived from the process words of N such that for a given process word $w = [E, \leqslant, l]$ the corresponding pomset trace is $\alpha_{w,N} = [E, \leqslant, l_N \circ l]$.

It follows that there are three levels on which the causal behaviour of a given net can be represented: in ascending order of abstraction they are the processes, the process words and the pomset traces of the net. Essentially, process words are derived from processes by forgetting the places and pomset traces are derived from process words by applying the labelling function of the net, and thereby forgetting the transition names. Note that this definition of pomset traces via process words is equivalent to the one in which the pomset traces are derived directly from the processes.

The following lemma states that to determine the marking after a process it is sufficient to have the multiset of transitions of N that occur as labels in the process.

Lemma 2.5. *Let N be a net; let (K, p) be a process of N.*

The marking M reached after (K, p) is completely determined by the multiset of transitions of N occurring in K, which is given by $l^*(w)$, where w is the process word derived from (K, p); we have $M = (M_0 + l^*(w)^\bullet) - {}^\bullet l^*(w)$.

Proof. This follows from the definition of the dynamic behaviour, which is reflected in the notion of a process. □

The following proposition now states some properties of process words that allow us to disregard processes in the remainder of this paper.

Lemma 2.6. *Let N be a net with set of transitions T.*

- *The set of process words of N is prefix closed.*
- *If w is a process word of N and $((M_0 + l^*(w)^\bullet) - {}^\bullet l^*(w))[t\rangle$ for some transition $t \in T$, then N has a process word w' such that w is a prefix of w' and $l^*(w') = l^*(w) + \{t\}$.*

Proof. Straightforward from properties of processes [1]. □

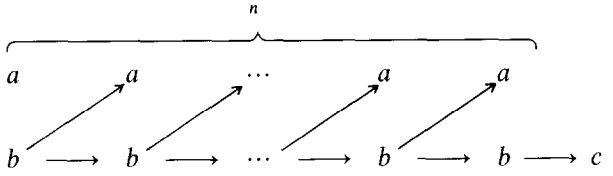
3. Results

Now we come to the actual contents of this note. We consider the term $B=(a.nil \mid b.B)+c.nil$ discussed in the introduction. We use an alphabet $A=\{a, b, c\}$. Let us first analyse the causal behaviour of B in terms of its (finite) pomset traces.

There are several ways to derive the pomset traces of a given CCS expression. One possibility is to consider a well-established Petri net semantics yielding an infinite net. For instance, [9] generates the net shown in Fig. 1.

Alternatively, we can apply, e.g., [2], which defines *pomset transitions* between terms. For instance, B allows the transition $B \xrightarrow{a \mid (b.(a \mid (b.(a \mid b.c)))} nil \mid (nil \mid (nil \mid nil))$, with an obvious pomset interpretation.

Lemma 3.1. *The finite pomset traces of B are prefixes of the partial A-words of the following form for unbounded $n \geqslant 0$:*



Proof. This follows from either of the constructions discussed above. □

Two systems are called *pomset trace equivalent* iff they have the same set of pomset traces. Pomset trace equivalence will be used as the weakest criterion to decide whether two systems have the same causal behaviour. A slightly stronger criterion is obtained by extending pomset trace equivalence to distinguish terminated executions from partial executions, thus considering *completed pomset trace equivalence*, the obvious generalization of completed trace equivalence [3] to pomset semantics.

We consider the question whether the causal behaviour of B can be modelled correctly by finite nets. Let us first recall the argument from [4] showing why the most straightforward construction does not work. Applying the constructions from [4] to the term B directly would yield the net shown in Fig. 2.

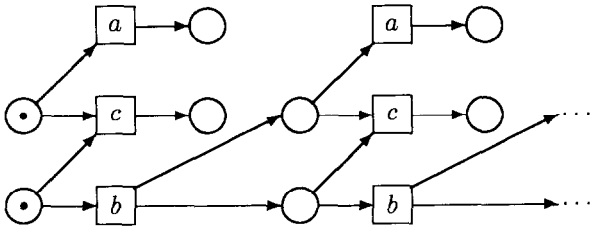


Fig. 1.

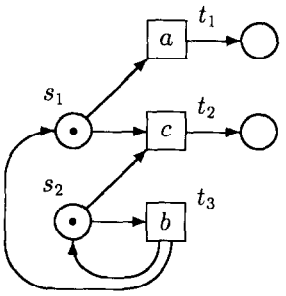
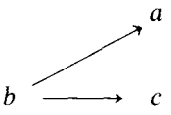


Fig. 2.

It turns out that this net actually models neither the causal behaviour nor the branching structure of B correctly. To see that it does not correctly model the causal behaviour, note that the above net has the pomset trace depicted by



which according to Lemma 3.1 is not a pomset trace of B. Hence the net is not pomset trace equivalent to B. To see that the net does not correctly model the branching structure, note that after firing first the b transition and then the a transition, the net above cannot refuse c: if $M_0[t_3\rangle M_1[t_1\rangle M_2$, then $M_2(s_1)=1$ and $M_2(s_2)=1$ and hence $M_2[t_2\rangle$. However, $B \xrightarrow{b} a.nil \mid B \xrightarrow{a} a.nil \mid (nil \mid b.B)$ which *does* refuse c. Hence the net is not (interleaving) failure equivalent to B. (It is however interleaving trace equivalent to B, although proving that is outside the scope of this paper.)

Partly because of the failure of the above net to model the branching structure of B correctly, [4] restricts itself to a subset of CCS which excludes terms of the form $(P_1 \mid P_2)+P_3$ (and thereby also the term B). We note however that the constructions of [4] still work for such terms as long as they appear *outside the scope of all recursion*. Using this fact we will prove that there does exist of finite net modelling B correctly with respect to pomset trace equivalence and even completed pomset trace equivalence.

Theorem 3.2. *The behaviour of B is modelled up to (completed) pomset trace equivalence by the net shown in Fig.* 3.

Proof. The net in Fig. 3 is the result of applying the constructions of [4] to the term

$$B_1 = (a.nil \mid (b.B_2 + b.c.nil)) + c.nil,$$

where

$$B_2 = a.nil \mid (b.B_2 + b.c.nil).$$

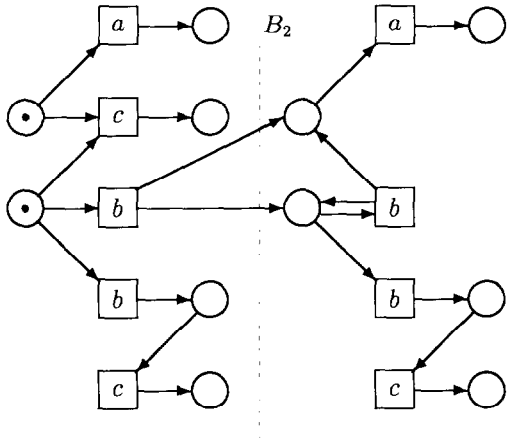


Fig. 3.

Note that B_1 is an example of a term of the form $(P_1 \mid P_2)+P_3$ outside the scope of recursion.

We will show that B_1 is completed pomset trace equivalent to B. Our proof is based not on the definition of the equivalence but on the known property that the equivalence does not respect nondeterminism. That is, completed pomset trace equivalence, denoted $\simeq_{\text{cpt}}$, satisfies the following rule for all CCS terms P, Q which are unable to deadlock or diverge immediately:

$$a.P + a.Q \simeq_{\text{cpt}} a.(P+Q).$$

Using this rule, unfolding and folding, we can derive

$$\begin{aligned} B_1 &\simeq_{\text{cpt}} (a.nil \mid b.(B_2 + c.nil)) + c.nil \\ &= (a.nil \mid b.(a.nil \mid (b.B_2 + b.c.nil) + c.nil)) + c.nil \\ &= (a.nil \mid b.B_1) + c.nil \end{aligned}$$

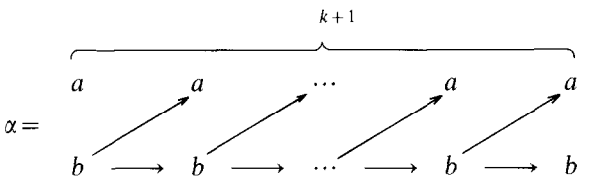
and hence B_1 is a solution for B with respect to completed pomset trace equivalence. It follows that the net above models B modulo completed pomset trace equivalence. □

Problems arise when trying to encode the branching structure of B. References [4, 11] show that B can still be modelled up to step bisimulation. However, if one wants to represent causality as well as some information about the branching structure then one can no longer use finite nets.

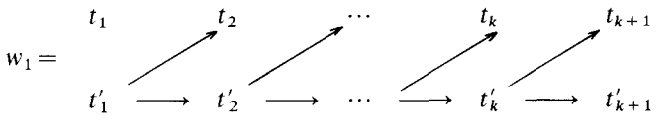
Theorem 3.3. *Let $\simeq$ be any equivalence relation that implies both pomset trace equivalence and interleaving failure equivalence; then there does not exist a finite net that models B up to $\simeq$.*

Proof. Suppose that $N=(S, T; M_0)$ represents a finite net that models B up to $\simeq$. The number of a-labelled transitions in N is finite; let it be k.

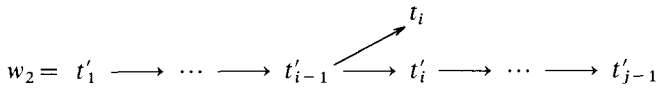
According to Lemma 3.1, the following partial A-word α is a pomset trace of B:



α is a pomset trace of N because $\simeq$ respects pomset trace equivalence; hence (Definition 2.4) N has a process word w_1 such that $\alpha_{w_1, N}=\alpha$; let w_1 be given by the following figure, where $l_N(t_i)=a$ and $l_N(t'_i)=b$ for all $1 \leqslant i \leqslant k+1$:



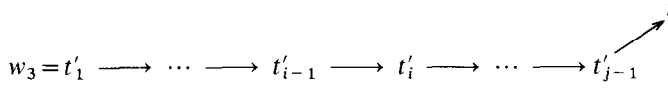
The labels in this figure represent the T-labels of w_1. Now because there are only k different a-transitions in N there must be a pair $i, j \leqslant k+1$ such that $i<j$ and $t_i=t_j$. Let i, j be some such pair, and consider the following partial T-word:



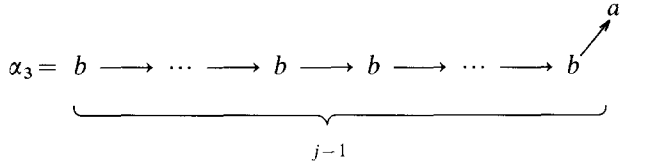
Clearly w_2 is a prefix of w_1 above, and hence Definition (2.6) it is a process word of N.

Let us now look at failures. One interleaving of w_2 is the transition sequence $t'_1 \cdots t'_{i-1} t_i t'_i \cdots t'_{j-1}$. The corresponding action sequence is given by $b^{i-1}ab^{j-i}$. Note that this sequence ends in at least one b. On the other hand, after this sequence B can only refuse the empty set, because the final b-action introduces another level of recursion, so each action in A may occur next. Because $\simeq$ respects failure equivalence by hypothesis, it follows that after w_2 there must be a c-transition enabled in N. This implies that there is a transition t such that $l_N(t)=c$ and $((M_0+l^*(w_2)^\bullet)-{}^\bullet l^*(w_2))[t\rangle$.

Now consider the following partial T-word:



Again, w_3 is a prefix of w_1, hence it is a process word of N. The corresponding pomset trace $\alpha_3=\alpha_{w_3, N}$ is shown by



Now because $l^*(w_3)=l^*(w_2)$ (since $t_i=t_j$), it follows by Definition 2.5 that $((M_0+l^*(w_3)^\bullet)-{}^\bullet l^*(w_3))[t\rangle$. But then (Lemma 2.6) there is a process word w_4 such that w_3 is a prefix of w_4 and $l^*(w_4)=l^*(w_3)+\{t\}$. Hence α_3 is a prefix of $\alpha_4=\alpha_{w_4,N}$ and $l^*(\alpha_4)=l^*(\alpha_3)+\{c\}$.

Now Lemma 3.1 shows that the only pomset traces of B in which c occurs are such that c depends on a b-action that does not precede an a-action; but in α_3 there are no such b-actions. It follows that α_4 is not a pomset trace of B. But this contradicts the assumption that $\simeq$ respects pomset trace equivalence. □

4. Related work

Besides the approaches mentioned earlier in the paper, we would like to comment briefly on some more work related to this note.

A similar behaviour notion for nets as used here has been considered in e.g. [7, 8, 13]. However, the notion investigated there assumes closure under augmentation of ordering, and in this respect is different from our process words (*abstract processes* in [10]) or pomset traces. In [8] it has been shown that the behaviour notion we have used is retrievable for one-safe nets, but not for general P/T nets.

Finite net representation for CCS are also considered in [6]. In this approach, the behaviour of nets is considered on another level of abstraction such that sequences of transitions may in a sense be considered as atomic. By mapping such atomic sequences of transitions to actions, [6] may indeed construct a finite net for our CCS term B.

Acknowledgment

We wish to thank Alan Mycroft for taking the time to look at this problem once more and for his comments on draft versions of this note. Thanks also to Walter Vogler and Dirk Taubner for helpful comments.

References

[1] E. Best and C. Fernandez, *Non-Sequential Processes*, EATCS Monographs on Theoretical Computer Science, Vol. 13 (Springer, Berlin, 1988).

[2] G. Boudol and I. Castellani, Concurrency and atomicity, *Theoret. Comput. Sci.* **59** (1988) 25–84.

[3] R.J. van Glabbeek, The linear time – branching time spectrum, SFB-Bericht 342/13/90 A, Technische Universität München, July 1990; extended abstract in: J.C.M. Baeten and J.W. Klop, eds., *CONCUR '90*, Lecture Notes in Computer Science, Vol. 458 (Springer, Berlin, 1990) 278–297.

[4] U. Goltz, Über die Darstellung von CCS-Programmen durch Petrinetze, Ph.D. Thesis, Aachen, GMD-Bericht 172, 1988; see also: U. Goltz, On representing CCS programs by finite Petri nets, in: M.P. Chytil, L. Janiga and V. Koubek, eds., *Proc. MFCS '88*, Lecture Notes in Computer Science, Vol. 324 (Springer, Berlin, 1988) 339–350.

[5] U. Goltz and A. Mycroft, On the relationship of CCS and Petri nets, in: J. Paredaens, ed., *Proc. ICALP '84*, Lecture Notes in Computer Science, Vol. 172 (Springer, Berlin, 1984) 196–208.
[6] R. Gorrieri and U. Montanari, SCONE: A simple calculus of nets, in: J.C.M. Baeten and J.W. Klop, eds., *Proc. CONCUR '90*, Lecture Notes in Computer Science **458** (Springer, Berlin, 1990) 2–30.
[7] J. Grabowski, On Partial Languages, *Fund. Inform.* **IV.2** (1981) 428–498.
[8] A. Kiehn, On the interrelationship between synchronous and non-synchronous behaviour of Petri nets, *EIK* **24** (1988) 3–18.
[9] E.-R. Olderog, Operational Petri net semantics of CCSP, in: G. Rozenberg, ed., *Advances in Petri Nets 1987*, Lecture Notes in Computer Science, Vol. 266 (Springer, Berlin, 1987) 196–223.
[10] W. Reisig, On the semantics of Petri Nets, in: E.J. Neuhold and G. Chroust, eds., *Formal Methods in Programming*, IFIP (Elsevier Science Publishers B.V., North-Holland, 1985) 347–372.
[11] D. Taubner, personal communication, 1987.
[12] D. Taubner, Finite representations of CCS and TCSP programs by automata and Petri nets, Ph.D. Thesis, TU Munich, Lecture Notes in Computer Science, Vol. 369 (Springer, Berlin, 1989).
[13] W. Vogler, Failures semantics based on interval semiwords is a congruence for refinement, *Distributed Comput.* **4** (1991) 139–162.