Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

R.J. van Glabbeek, U. Goltz

Partial order semantics for refinement of actions -neither necessary nor always sufficient but appropriate when used with care-

Computer Science/Department of Software Technology Note CS-N8901 July

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Partial Order Semantics for Refinement of Actions
- neither necessary nor always sufficient but appropriate when used with care -

Rob van Glabbeek
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Nederlands

Ursula Goltz
Gesellschaft für Mathematik und Datenverarbeitung
Postfach 1240, 5205 Sankt Augustin 1, Federal Republic of Germany

1980 Mathematics Subject Classification (Zentralblatt für Mathematik): 68B10.
1985 Mathematics Subject Classification (Mathematical Reviews): 68Q55.
1987 CR Classification scheme (Computing Reviews): F.3.2.
Key Words & Phrases: Concurrency, Semantic equivalences, Action refinement, Bisimulation, Interleaving vs. Partial orders, Petri nets.
Note: Sponsored in part by Esprit project no. 432, METEOR.

This note continues a series of papers in the Bulletin of the EATCS about the relative merits of partial order semantics and interleaving semantics, starting with [CDP]. That paper pointed out a significant advantage of partial order semantics, by formulating a desirable property of semantic equivalences that is not met by interleaving equivalences. This property is *preservation under refinement of actions.* A semantic equivalence is *preserved under action refinement* if two equivalent processes remain equivalent after replacing all occurrences of an action a by a more complicated process $r(a)$. For example, $r(a)$ may be a sequence of two actions a_1 and a_2. This property may be desirable in applications where concurrent systems are modelled at different levels of abstraction, and where the actions on an abstract level turn out to represent complex processes on a more concrete level. Therefore for example [Pratt] and [Lamport] already advocate the use of semantic equivalences that are not based on action atomicity.

[CDP] showed by means of a simple example that none of the interleaving equivalences - not even bisimulation - is preserved under action refinement. Furthermore they claim that 'on the other hand, the approaches based on partial order are not constrained to the assumption of atomicity'. Indeed, they give a proof that "linear time" partial order semantics, where a system is identified with the set of its possible (partially ordered) runs, is preserved by refinement. They conclude that 'interleaving semantics is adequate only if the abstraction level at which the atomic actions are defined is fixed. Otherwise, partial order semantics should be considered'.

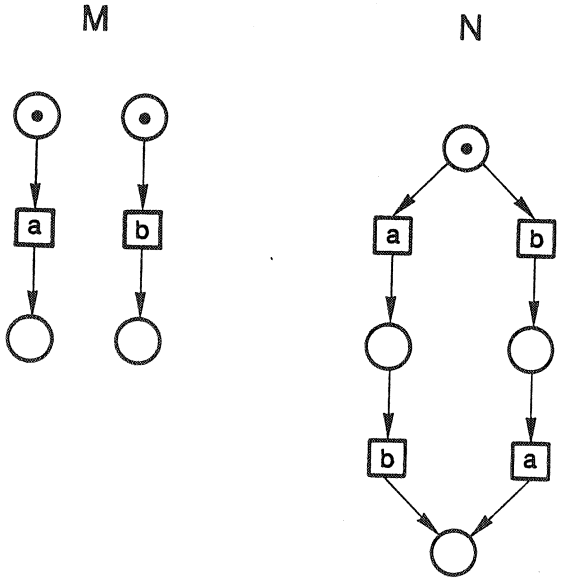
In this note we would like to point out that this conclusion is not so obvious. In particular we will argue

- that there are several equivalences based on partial orders which are *not* preserved by refinement (namely when taking the choice structure of systems into account);
- that nevertheless a "branching time" partial order equivalence can be found that is preserved under refinement;
- but that, in order to achieve preservation under refinement it is not necessary to employ partial order semantics: there exist equivalences that abstract from the causal structure of concurrent systems and are still preserved under refinement.

In interleaving semantics, the possible runs of a system are represented as sequences of action occurrences, modelling parallelism by arbitrary interleaving of actions. The example of [CDP] consisted of the two systems M and N which may not be distinguished in this kind of semantics:

$\mathrm{M} = a \parallel b$ (two actions a and b, executed independently);
$\mathrm{N} = a;b + b;a$ (either the sequence ab or the sequence ba is executed).

They have the following Petri net representations (labelling transitions by action names):



It was shown that after refining a into the sequential composition of a_1 and a_2, thereby obtaining the systems

$$M' = (a_1;a_2) \parallel b \text{ and } N' = (a_1;a_2);b + b;(a_1;a_2),$$

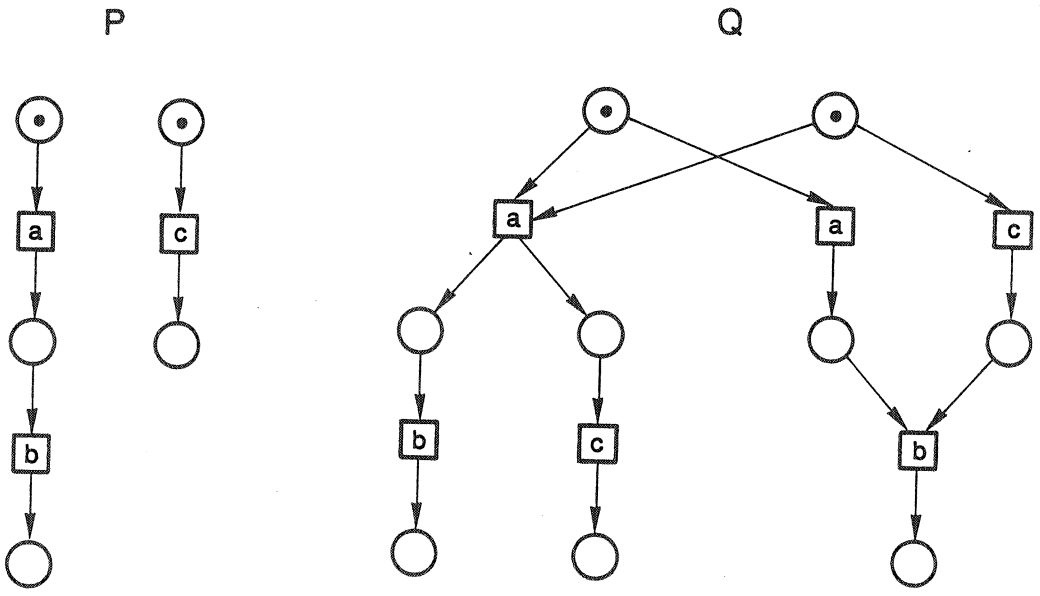
M' can perform the sequence of actions $a_1 b a_2$, while N' cannot do this.
Hence M' and N' are not equivalent in interleaving semantics.

A first attempt to capture parallelism more precisely is made by so called *step semantics*. Here it is specified that in a run of a parallel system several independent actions may occur together in one *step*. We can think of a system having a global clock where at each clock tick several actions occur simultaneously. This view is taken in calculi like SCCS [Milner], CIRCAL [Milne] and MEIJE [AB]. Step semantics also have been given to CCS in [DDMa] and to TCSP in [TV].

It is easy to see that the two systems M and N considered above are already distinguished in step semantics: In M it is possible to execute the step $\{a, b\}$ whereas in N it is not. So the example in [CDP] is not well chosen to advocate partial order semantics; already step semantics would be sufficient in this case. Therefore, we will now give a slightly more elaborate example. Consider the following two systems:

$P = (a; b) \parallel c,$
$Q = a; (b \parallel c) + (a \parallel c); b.$

In both of these systems the actions a, b and c are executed, and b occurs after completion of a. However, in P the c action occurs independently of both a and b whereas in Q c may only occur either "causally behind" a or "causally before" b. P and Q may be represented by the following Petri nets (using a construction explained for instance in [GV] for implementing $+$).



P and Q are identified when considering their possible sequences of steps. Both of them take into account the five possibilities for c: occurring before a, simultaneous with a, between a and b, simultaneous with b, or after b. However, after substituting $(c_1; c_2)$ for c only the first system can perform the sequence of actions $c_1 a b c_2$. Thus

also this semantics is not preserved under refinement.

On the other hand, P and Q can be distinguished by considering the *partial orders of action occurrences* they allow.

$$\begin{array}{ccc} a & \rightarrow & b \\ & c & \end{array} \qquad \begin{array}{l} (a \text{ followed by } b \text{ and} \\ \quad \text{independently } c) \end{array}$$

is a computation of P but not of Q. In [CDP] it was shown that partial order semantics - when identifying a system with its set of possible (partially ordered) runs - is preserved under action refinement.

However, when taking the choice structure of systems into account, the situation becomes less obvious.

Before discussing the problem in detail, we would like to give an overview, by classifying the equivalences being currently investigated (without claiming completeness). They may be positioned in a two dimensional diagram as shown below, distinguishing them firstly with respect to the preserved level of detail in runs of systems (as discussed above) and secondly with respect to the preserved level of detail of the choice structure of systems (we do not consider abstraction from internal actions here). In trace semantics ("linear time" semantics), a system is fully determined by its set of possible runs, thereby completely neglecting the branching structure. On the other end, bisimulation semantics preserve the information where two different courses of action diverge (although branching of identical courses of action is still neglected). In between there are several "decorated trace semantics", where part of the branching structure is taken into account. Mostly these are motivated by the observable behaviour of processes, according to some testing scenerio.

runs / branching structure	sequences of actions	sequences of steps	partial orders
paths	interleaving trace equivalence	step trace equivalence	pomset trace equivalence
⋮ e.g. testing			
bisimulation	interleaving bisimulation equivalence	step bisimulation equivalence	e.g. pomset bisimulation equivalence

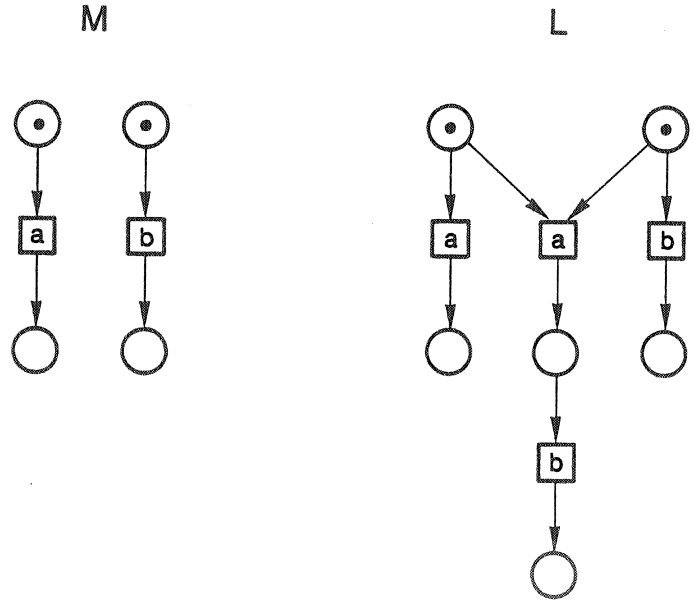
Up to now we have only considered the trace equivalences in the upper row of the diagram. We recalled from [CDP] that pomset trace equivalence is preserved under action refinement, while interleaving trace equivalence is not. Moreover we have shown that also step trace equivalence is not preserved under refinement. Next we will try to establish similar results for the corresponding branching time equivalences and for the testing equivalences in between.

In interleaving semantics this generalisation is quite simple. As observed in [CDP], the systems M and N are identified even in interleaving bisimulation semantics while the refined systems M' and N' are not even identified in interleaving trace semantics. So there is one single example showing that neither interleaving bisimulation equivalence nor interleaving trace equivalence is preserved under refinement. As a consequence, also none of the decorated trace equivalences based on interleaving, which are more discriminating then interleaving trace equivalence, but less discriminating then interleaving bisimulation equivalence, is preserved under refinement; in each of the decorated trace semantics based on interleaving, M and N are identified, while M' and N' are distinguished.

Our example against step trace equivalence however cannot be used to show that also step bisimulation equivalence is not preserved under refinement; the systems P and Q happen to be different in step bisimulation semantics already: after performing an a-action the system P is always able to continue with a b-action, whereas Q can perform an a-action and reach a state where it is not possible to continue with b. Nevertheless, the following example shows that also step bisimulation semantics is not preserved under refinement. Consider the two systems M and L which may not be distinguished in step bisimulation semantics:

$M = a \parallel b$ (two actions a and b, executed independently);
$L = a \parallel b + a;b$ (either a and b are executed independently or the sequence ab is executed).

They have the following Petri net representations:



The systems $M' = (a_1;a_2) \parallel b$ and $L' = (a_1;a_2) \parallel b + (a_1;a_2);b$ which are obtained by substituting $a_1;a_2$ for a are no longer step bisimulation equivalent; only L' can perform a_1, and reach a state where it is not possible to continue with b.

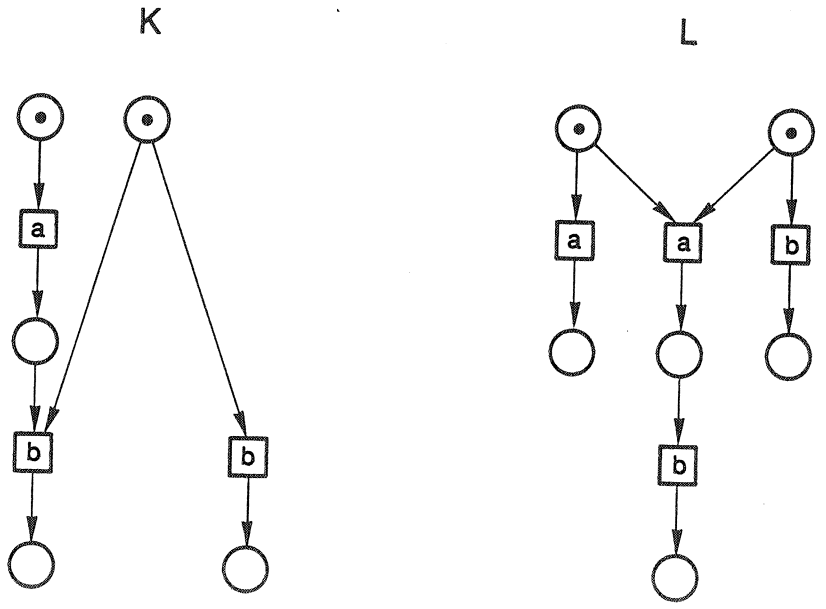
Hence, neither step trace nor step bisimulation equivalence is preserved under refinement. However, M' and L' happen to be step trace equivalent, so none of the previous two examples is adequate for both equivalences. In order to tackle the whole range of equivalences included between step trace and step bisimulation equivalence we need yet another example, which simultaneously shows that both step trace and step bisimulation equivalence are not preserved under refinement. Consider the systems

$$Q = a;(b \parallel c) + (a \parallel c);b \quad \text{and}$$
$$R = Q + P = a;(b \parallel c) + (a \parallel c);b + (a;b) \parallel c.$$

The Petri net associated to Q has been shown before, and the net for R is drawn in [GV], where it was also pointed out that Q and R are step bisimulation equivalent. However, after refining c into $c_1;c_2$ the two systems are not even interleaving trace equivalent; only the second system can perform the sequence of actions c_1abc_2. As a consequence, none of the decorated trace equivalences based on steps, such as the step failure semantics of [TV], is preserved under refinement.

A rather straightforward combination of the ideas of bisimulation and of capturing causal dependencies by partial orders has been proposed in [BC]. They suggest to consider transition systems as for the usual interleaving bisimulation, but to label the arcs in these transition systems by *pomsets* (partially ordered multisets of action occurrences) instead of single actions. However, it turns out that the obtained equivalence, usually called *pomset bisimulation*, is not preserved by refinement of actions.

Consider the two systems K and L below.



In both systems either a and b are executed independently or the sequence ab is executed. However, in L the choice between these two options is made at the beginning, while in K this choice can be postponed until the execution of a has been completed.
The system K can behave as follows:

- it performs the single action a and the remaining behaviour is $b + b$, which is identified with b;
- it performs the single action b and the remaining behaviour is a;
- it performs the pomset $\begin{matrix} a \\ b \end{matrix}$ (a and b executed independently) and there is no remaining behaviour;
- or it performs the pomset $a \rightarrow b$ (a followed by b) and again there is no remaining behaviour.

The behaviour of L can be described in exactly the same way and for this reason the two systems are pomset bisimulation equivalent.

Now let us imagine that a is refined into $a_1; a_2$. The systems K' and L' which are obtained in this way can be distinguished in pomset bisimulation semantics, and even in interleaving bisimulation: only L' can refuse to do a b-action after execution of a_1.

Hence pomset bisimulation semantics is not preserved under refinement of actions. Another example for this are the two terms

$$a;(b+c)+(a \parallel b) \text{ and } a;(b+c)+(a \parallel b)+(a;b)$$

(again refining a into $a_1; a_2$). However the example given before can also be used to show that even the notion of generalised pomset bisimulation, as discussed in [GV], is not preserved under refinement. Of course we cannot find an example tackling the whole range of equivalences included between pomset trace and pomset bisimulation semantics, since we already observed that pomset trace equivalence is preserved under refinement. However, the systems K' and L' can already be distinguished in interleaving failure semantics, as employed in [BHR, DH]. Thus no equivalence that is at least as discriminating as interleaving failure equivalence but less discriminating then pomset bisimulation equivalence can be preserved under refinement.

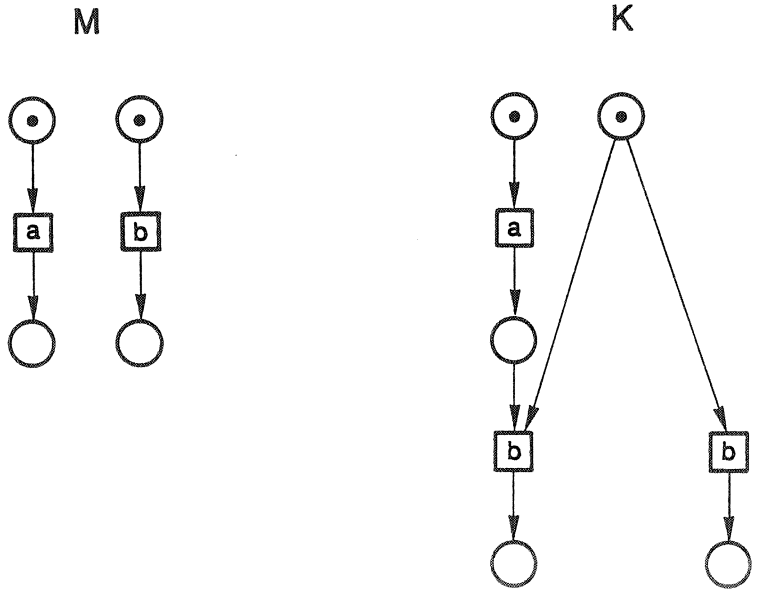
The interplay of equivalence notions and refinement of actions as discussed up to now has been investigated in detail in [GG]. There all the equivalence notions and examples presented so far are given formally in the framework of event structures; refinement of actions is performed by replacing actions by finite non-empty pomsets. That paper concludes by showing that another "partial order bisimulation" is indeed preserved by refinement. In order not to bore readers with technical details, we just ouline these results here.

After we realised that pomset bisimulation is not preserved by refinement, another equivalence was considered, hoping that it would solve the problem (see e.g. [Devillers]). This equivalence had been considered before under the name NMS partial ordering equivalence in [DDM]. The main idea is to bisimulate transition systems where the states are labelled by their (partially ordered) histories. In [GG] it is shown that this equivalence is indeed preserved by refinement when we restrict ourselves to systems *without autoconcurrency*, that is to systems which do not allow concurrent occurrences of the same action like in $a \parallel a$. However, for systems with autoconcurrency it turns out that NMS po equivalence is not preserved by refinement. Even more, it does not even respect pomset bisimulation equivalence in this case. The example showing both these facts was suggested to us by Alex Rabinovich who used it to show that this equivalence is not a congruence with respect to a TCSP-like parallel composition. To obtain a conguence, a stronger version of NMS po equivalence was suggested in [RT]. In [GG] it is shown that *this* "partial order bisimulation equivalence" is indeed always preserved by refinement.

So we have shown that it is *not automatically sufficient* to move to partial order semantics for refinement of actions. When considering the choice structure, this has to been done with care. In the remaining part of this note, we argue that on the other hand it is *not even necessary* to move to partial orders (as one may conclude from [CDP]).

A branching time semantics lying strictly between step semantics and partial order semantics has been proposed in [GV]. This *ST-bisimulation semantics* is based on the idea that actions have a duration, and may overlap in time. Contrary to step semantics, it recognises the possibility that, in $P = (a;b) \parallel c$, action c may have an overlap with both a and b, while b can only occur after completion of a. However when in a run of a system an action b happens after completion of a, it is not taken into account whether or not there is a causal link between the two actions.

Compare for instance the systems M and K that have been presented before.



Both systems perform an a-action and a b-action. In M these actions are always independent, whereas in K it is possible to perform a b-action which causally depends on a: so M and K are distinguished in partial order semantics. However, in ST-bisimulation semantics the only execution of K which is not possible in M (first a and then the b which is causally dependent on this a) can not be distinguished from another execution of K (and of M), namely: first a and then the b which is independent of this a. In K, the choice between both runs is only made after completion of a, and in that state the remaining part of both executions is the same: just b. Hence M and K are identified in ST-bisimulation semantics.

So ST-bisimulation equivalence abstracts from the causal structure of concurrent systems. Nevertheless it is preserved under refinement [van Glabbeek]. A similar result can be proved for linear time semantics as well. A variant of this can be found in [NEL]. Furthermore a variant of failure semantics, based on the same ideas that underly ST-bisimulation semantics has been proposed in [Vogler]. There it is proven that also this equivalence respects refinement.

This shows that indeed partial order semantics (in the strong sense) are not necessary for the type of refinement we have considered. Nevertheless, we need partial order bisimulation semantics when it is required to model the interplay of causality and branching in full detail.

We hope that this note, and the formal versions of it [GG, van Glabbeek], help to clarify the relationship between various equivalences being currently considered. However, we do not intend to advocate any particular type of equivalence here. We just want to illustrate that the appropriate equivalence notion has to be chosen carefully with regard to the considered questions.

Acknowledgements

The idea for this note arose in discussions with Albert Meyer and Ernst-Rüdiger Olderog at ICALP 87 in Karlsruhe.

References

[AB] D. Austry, G. Boudol: *Algèbre de processus et synchronisations*, Theoretical Computer Science, Vol. 30, pp. 91-131, 1984

[BC] G. Boudol, I. Castellani: *On the Semantics of Concurrency: Partial Orders and Transition Systems*, Proc. TAPSOFT 87, Vol. I, LNCS 249, Springer-Verlag, pp 123-137, 1987

[BHR] S.D. Brookes, C.A.R. Hoare, A.W. Roscoe: *A Theory of Communicating Sequential Processes*, Journal of the ACM, Vol. 31, No. 3, pp 560-599, 1984

[CDP] L. Castellano, G. De Michelis, L. Pomello: *Concurrency vs Interleaving: An Instructive Example*, Bulletin of the EATCS 31, pp 12-15, 1987

[DDM] P. Degano, R. De Nicola, U. Montanari: *Observational Equivalences for Concurrency Models*, in : Formal Description of Programming Concepts - III, Proc. of the third IFIP WG 2.2 working conference, ed. M. Wirsing, Elsevier Science Publishers B.V. (North Holland), pp 105-129, 1987

[DDMa] P. Degano, R. De Nicola, U. Montanari: *A Distributed Operational Semantics for CCS Based on Condition/Event Systems*, Acta Informatica Vol. 26, pp. 59-91, 1988

[Devillers] R. Devillers: *On the Definition of a Bisimulation Notion Based on Partial Words*, Petri Net Newsletter 29, pp 16-19, April 1988

[DH] R. De Nicola, M. Hennessy: *Testing Equivalences for Processes*, Theoretical Computer Science, Vol. 34, pp. 83-133, 1984

[van Glabbeek] R.J. van Glabbeek: *The Refinement Theorem for ST-bisimulation semantics*, manuscript, 1989

[GG] R.J. van Glabbeek, U. Goltz: *Equivalence Notions for Concurrent Systems and Refinement of Actions*, Arbeitspapiere der GMD 366, February 1989, Extended Abstract to appear in Proc. MFCS 89, LNCS, Springer-Verlag, 1989

[GV] R.J. van Glabbeek, F.W. Vaandrager: *Petri Net Models for Algebraic Theories of Concurrency*, Proc. PARLE, Vol. II, LNCS 259, Springer-Verlag, pp 224-242, 1987

[Lamport] L. Lamport: *On Interprocess Communication*, Distributed Computing 1, pp 77-101, 1986

[Milne] G.J. Milne: *CIRCAL and the Representation of Communication, Concurrency and Time*, Transactions on Programming Languages and Systems (ACM), Vol. 7, No. 2, pp 270-298, 1985

[Milner] R. Milner: *Calculi for Synchrony and Asynchrony*, Theoretical Computer Science, Vol. 25, No. 3, pp 267-310, 1983

[NEL] M. Nielsen, U. Engberg, K.S. Larsen: *Partial Order Semantics for Concurrency*, in: Course Material of the REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, 1988

[Pratt] V.R. Pratt: *Modelling Concurrency with Partial Orders*, International Journal of Parallel Programming, Vol. 15, No. 1, pp 33-71, 1986

[RT] A. Rabinovich, B.A. Trakhtenbrot: *Behavior Structures and Nets*, Fundamenta Informaticae, Vol. XI, No. 4, pp 357-404, 1988

[TV] D. Taubner, W. Vogler: *The Step Failure Semantics*, Proc. STACS 87, LNCS 247, Springer-Verlag, pp 348-359, 1987

[Vogler] W. Vogler: *Failure Semantics Based on Interval Semiwords is a Congruence for Refinement*, manuscript, 1989