Verification of Cooperating Systems by Simple Homomorphisms Using the Product Net Machine

Peter Ochsenschläger
Institut für Telekooperationstechnik der GMD

1. Introduction

Semantics of most operational specification methods for distributed systems is based on labelled transition systems /BW,Mi,Peh,Ta/. Such an approach describes the dynamics of a system by linear sequences of actions defined by a labelled transition system. Concurrency of actions is only represented implicitly by arbitrary interleaving of actions. This interleaving semantics captures essential aspects of distributed systems as for example nondeterminism, termination, absence of deadlocks and livelocks, etc. /Ta/.

In that context verification of formal specifications means comparing labelled transition systems which represent different levels of abstraction /Peh/. An evidently necessary criterion is language equivalence. It is well known that this equivalence is not deadlock sensitive and does not reflect liveness properties. Therefore equivalence notions have been defined which are deadlock sensitive and stronger than language equivalence /BW,Mi,Ta/. Furthermore for verification purposes liveness properties are investigated explicitely by temporal logic /CES/ or ω-languages /ACW,As2/.

In this paper equivalence of labelled transition systems is investigated in a formal language theoretic manner. A property of language homomorphisms is defined which guarantees that not only safety but also liveness properties of a homomorphic image of a language imply corresponding properties of the language itself. In that case language equivalence of labelled transition systems is strong enough for verification purposes. This definition formally captures the equivalence of systems which are intuitively equivalent but not in the sense of known equivalence notions for labelled transition systems.

In the title of this paper the notion of cooperating instead of distributed systems is used. Both notions are not formally defined. By using the first one we concentrate our interest on systems that are characterized by freedom of decisions and loose coupling of the cooperating partners. That causes a high degree of nondeterminism which is appropriately controlled by the notion of simplicity.

In the following chapter simplicity is defined for language homomorphisms and for labelled transition systems. Simple properties are shown and some theorems are stated. Chapter 3 presents two examples, for which special safety and liveness properties are proven using the product net machine /Oc3/.

2. Simple Homomorphisms

A labelled transition system /BW,Mi,Ta/ is a directed graph with an initial node. Its edges are labelled by elements of $\Sigma \cup \{\tau\}$. Thereby Σ is a set of "visible actions" and $\tau \notin \Sigma$ denotes the "invisible action". The nodes correspond to the states of a system and the edges to the state transitions.

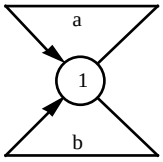

Fig. 2.1

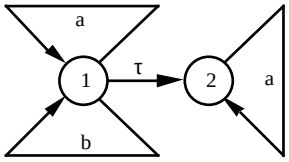

Fig. 2.2

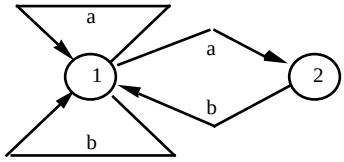

Fig. 2.3

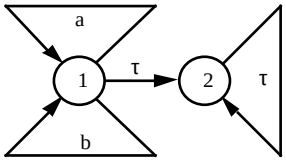

Fig. 2.4

Concerning only the visible actions the labelled transition systems in fig. 2.1 and fig. 2.2 (with initial state 1) describe systems which may repeatedly perform the actions a and b in arbitrary order. Additionally the system of fig. 2.2 may enter a phase (by an invisible action) in which it only performs the action a repeatedly.

The label of a path in a labelled transition system is the corresponding sequence of edge labels omitting τ labels. It is an element of Σ^*. (Σ^* is the set of all sequences of elements of Σ including the empty sequence ε.) The label of an empty path is ε.

The language of a labelled transition system is the set of all labels of paths starting from the initial node. In fig. 2.1 as well as in fig.2.2 $\{a,b\}^*$ is the language of the labelled transition system. So the language of a system in general does not reflect the structure of the system.

From a language theoretic point of view a labelled transition system is an automaton /Be,Per/ whose set of final states consists of all states. Nodes of the labelled transition system are states of the automaton and edges are state transitions. Those edges labelled by τ correspond to so called ε-moves of the automaton.

As we have seen above the language of labelled transition systems does not reflect the difference between fig. 2.1 and fig 2.2. To capture such differences several equivalence notions for labelled transition systems have been developed

/BW,Mi,Ta/. Test equivalence is one of the coarsest equivalences which is finer than language equivalence. It is based on so called failure pairs (w,X) with w Σ* and X Σ. They describe the property that there exist a path labelled with w and leading from the initial node to a node q such that there exist no outgoing edge of q labelled with an element of X. It is easy to show that (a,{b}) is a failure pair of fig. 2.2 but not of fig. 2.1. Therefore the two labelled transition systems are not test equivalent.

The language of the third example in fig. 2.3 is {a,b}* and the system is not test equivalent to fig. 2.1 and to fig. 2.2 because (a,{a}) is a failure pair of fig. 2.3 but not of fig. 2.1 and not of fig. 2.2. From an intuitive point of view however there is not such a difference between fig. 2.3 and fig. 2.1 as between fig. 2.2 and fig. 2.1. In contrast to fig. 2.2 after each sequence of actions the system of fig. 2.3 may eventually perform the actions a and b in arbitrary order.

If we interpret a as the negative and b as the positive answer to a request for a service then the system of fig. 2.3 intuitively provides the service as well as the system of fig. 2.1.

Such properties that "something good" eventually happens are usually called liveness properties /AS1/. They are the counterpart to safety properties /AS1/ which state that "some bad thing" does not happen in the sequences of actions. In general the language of a labelled transition system only expresses safety properties but no liveness properties as we have seen in our examples.

To develop a formal basis for our intuition concerning the differences between fig. 2.1, fig. 2.2 and fig. 2.3 we use some further language theoretic notation. Let Δ be the set of edges of a labelled transition system i.e. a set of triples (r,x,s) where r and s are nodes and x Σ {τ}. If P denotes the set of all paths starting from the initial node then P Δ* and P is prefix closed. (A formal language is called prefix closed /Be,Per/ if all prefixes of its elements are elements of the language.)

Let h : Δ* → ∑* be the alphabetic language homomorphism defined by h((r,x,s)) = x for x Σ and h((r,τ,s)) = ε. (A mapping f : Σ1* → Σ2* is called a language homomorphism /Be,Per/ if f(yz) = f(y)f(z) and f(ε) = ε. It is called alphabetic, if f(Σ1) Σ2 {ε}.)

With this notation h(P) is the language of a labelled transition system. The system itself is uniquely represented by P because the triples of Δ reflect the structure of the system. In automata theory P is called a local set. If we replace each label of an edge by the corresponding triple representing the labelled edge then we obtain a deterministic automaton recognizing the language P /Be,Per/.

For the rest of this chapter we assume each formal language to be prefix closed and each language homomorphism to be alphabetic. So a homomorphic image of a language is prefix closed too. Let Σ be a set, L Σ* a formal language (an arbitrary subset of Σ*) and x Σ*. The set $x^{-1}(L) = \{ y \in \Sigma^* \mid xy \in L \}$ is called the set of continuations of x in L or the left quotient of L with respect to x /Be,Per/.

If P Δ* denotes the set of all paths of a labelled transition system then the sets $h(x^{-1}(P))$ for x P contain information about liveness properties of the system. In fig. 2.2 for example $h((1,\tau,2)^{-1}(P)) = h(\{(2,a,2)\}^*) = \{a\}^*$ which means that the system only performs actions a after it reached state 2.

The language h(P) of the system only contains information about the sets $h(x)^{-1}(h(P))$ which are the unions of all sets $h(y^{-1}(P))$ with h(y) = h(x). Therefore $h(x^{-1}(P)) \subseteq h(x)^{-1}(h(P))$ and the inclusion can be proper. This is the reason why h(P) generally does not contain any information about liveness properties of the system. In fig. 2.2 $h((1,\tau,2))^{-1}(h(P)) = \varepsilon^{-1}(\{a,b\}^*) = \{a,b\}^*$ which says nothing about the restricted behaviour of the system after it reached state 2.

Failure pairs additionally contain information about sets $h(x^{-1}(P)) \cap \Sigma$. But as our examples show these sets don't capture the meaning of "something desired eventually happens" because they only look at the "next possible action". In contrast to this the following definition looks at the set of "possible sequences of actions in a certain future".

Let L Σ1* be a formal language, x L and f : Σ1* → Σ2* a homomorphism. f is called simple on L in x if there exists u $f(x)^{-1}(f(L))$ with $u^{-1}(f(x^{-1}(L))) = u^{-1}(f(x)^{-1}(f(L)))$. f is called simple on L if it is simple on L in each x L.

This definition means that "from a certain future $f(x^{-1}(L))$ equals $f(x)^{-1}(f(L))$". If f is simple on L in x then it is easy to see that f is simple on L in each prefix of x /Oc5/.

A labelled transition system is called simple if the corresponding homomorphism h : Δ* → Σ* is simple on the set P of all paths.

The simplicity of a homomorphism f on L (labelled transition system) in a certain sense guarantees that liveness properties of f(L) (the language of the system) which are captured by the sets of continuations in f(L) in a "modified form" also hold for L (the system) itself. In chapter 4 this will be demonstrated on a more complex example.

Since in fig. 2.1 $h(x^{-1}(P)) = h(x)^{-1}(h(P))$ for each path x P the simplicity of this labelled transition system is evident (u = ε). For each path x P of fig. 2.2 leading to node 2 $h(x^{-1}(P)) = \{a\}^*$ and $h(x)^{-1}(h(P)) = \{a,b\}^*$. Therefore there does not exist u $h(x)^{-1}(h(P))$ with $u^{-1}(h(x^{-1}(P))) = u^{-1}(h(x)^{-1}(h(P)))$. So the system of fig. 2.2 is

not simple. However P is simple ($u = \varepsilon$) on each path $y \in P$ leading to node 1, because in this case $h(y^{-1}(P)) = \{a,b\}^* = h(y)^{-1}(h(P))$. In fig. 2.3 the simplicity condition is fulfilled by $u = \varepsilon$ for paths leading to node 1 and by $u = b$ for those leading to node 2.

This shows that language equivalence together with simplicity reflect our intuition in comparing the three examples. In general for two language equivalent labelled transition systems three cases have to be distinguished concerning their liveness properties /Oc6/ :

(1) Both systems are simple. Then liveness properties of their language also hold for both systems in a "modified form".

(2) One system is simple and the other one is not simple. This leads to different liveness properties of the two systems.

(3) Both systems are not simple. In that case nothing can be said about liveness properties.

If $f(x^{-1}(L)) = \{\varepsilon\}$ for a homomorphism $f : \Sigma 1^* \rightarrow \Sigma 2^*$, $L \subseteq \Sigma 1^*$ and $x \in L$ then f is simple on L in x only if $f(x)^{-1}(f(L)) = \{\varepsilon\}$. In an earlier paper /Oc1/ this situation has been formalized by so called deadlock languages. The deadlock language DL of a language L with respect to a homomorphism f is defined by $DL = \{ u \in f(L) \mid$ there exist $x \in L$ with $u = f(x)$ and $f(x^{-1}(L)) = \{\varepsilon\} \}$ and the deadlock language DP of a labelled transition system is defined correspondingly substituting L by P and f by h. If a system has performed a (visible) sequence of actions $u \in DP$ then it may have reached a phase in which no more visible actions are possible. In the example of fig. 2.4 $DP = \{a,b\}^*$.

The termination language TL of a language L with respect to a homomorphism f is defined by $TL = \{ u \in f(L) \mid u^{-1}(f(L)) = \{\varepsilon\} \}$. With corresponding substitutions TP denotes the termination language of a labelled transition system. In fig. 2.4 TP = . It is easy to see that generally $TL \subseteq DL$ and that $TL = DL$ if the homomorphism f is simple on L /Oc5/. If f(L) is finite then each $x \in L$ is prefix of a $x' \in L$ with $f(x'^{-1}(L)) = \{\varepsilon\}$. Therefore by that assumption $TL = DL$ implies the simpicity of f on L. It is also easy to see that $TL = DL$ implies the simpicity if f(L) is non branching ($u^{-1}(f(L)) \cap \Sigma 2$ contains at most one element for each $u \in f(L)$). Therefore the homomorphism given in /Oc3/ for verifying the alternating bit protocol is simple.

At the end of this chapter we state some theorems about simple homomorphisms which are used in chapter 4. Their proofs can be found in /Oc5/. Let $f : \Sigma 1^* \rightarrow \Sigma 2^*$ and $g : \Sigma 2^* \rightarrow \Sigma 3^*$ be mappings. The composition $g \circ f : \Sigma 1^* \rightarrow \Sigma 3^*$ is defined by $(g \circ f)(x) = g(f(x))$ for $x \in \Sigma 1^*$. If g and f are homomorphisms then $g \circ f$ is a homomorphism too. The following theorem express the compatibility of simplicity with composition of homomorphisms.

Theorem 1 : If f is simple on $L \subseteq \Sigma 1^*$ and g is simple on $f(L) \subseteq \Sigma 2^*$ then $g \circ f$ is simple on L.

Theorem 2 : If $g \circ f$ is simple on $L \subseteq \Sigma 1^*$ then g is simple on f(L).

A formal language is called regular if it is recognized by a finite automaton /Be,Per/ (if it is the language of a finite labelled transition system in case of prefix closed languages).

Theorem 3 : If $L \subseteq \Sigma 1^*$ is regular and $f : \Sigma 1^* \rightarrow \Sigma 2^*$ is a homomorphism then it is decidable if f is simple on L.

The decision algorithm in the proof of this theorem /Oc5/ is very complex. Therefore it is important having a sufficient condition for simplicity which is easy to decide and applicable in many cases.

It is well known, that a directed graph can be partitioned into strongly connected components. A component with no outgoing edge is called dead. By a slide modification of the proofs in /Oc5/ we get the following sufficient condition:

Theorem 4 : Let L be a Language recognized by a finite automaton $\mathbb{A}$ and let h be a homomorphism on L. If for each $x \in L$ there exists $y \in x^{-1}(L)$ leading to a dead component of $\mathbb{A}$, such that each $z \in L$ with $h(z) = h(xy)$ leads to the same dead component, then h is simple on L.

The condition of theorem 4 is fulfilled for example, if each dead component contains a lable a of an edge with $h(a) \neq \varepsilon$, such that no edge exists outside of this component, whose lable has the same image h(a). This can be easily checked by the product net machine, and in many examples /Gi,Ne,Oc5,Oc7/ the simplicity of certain homomorphisms has been shown by this way.

3. Examples

Now we consider a simple version of a connection establishment and release protocol. It describes the cooperation of a sender A, a receiver B and a transport system T. A and B are coupled each by two FIFO-queues with the transport system T.

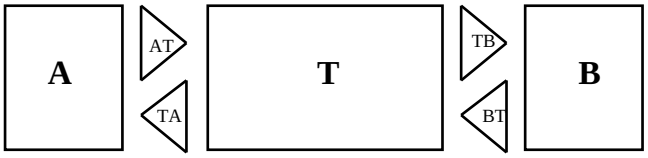

Fig. 3.1

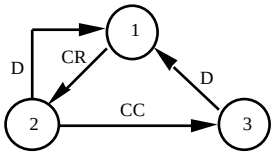

Fig. 3.2

If A wants to establish a connection to B using T then it informs T by sending CR (connect request). T may refuse this request by sending D (disconnect) because for example at the moment no resources are available for this connection or it may transmit this request to B. Now B may refuse the request by sending D or accept it by sending CC (connect confirm). In both cases T transmits the answer to A. By sending CC B enters the data phase which means that now B is ready to receive data. On the other side Aenters the data phase by receiving CC. Then Amay send data.

Each of A, B and T is free to release the connection independently and at any time i. e. not only in the data phase but also in the connection establishment phase. So each cooperating partner shows the pattern of behaviour as described in fig. 3.2.

Now the behaviour of A, B and T must be coordinated in such a way that connections are established and released "consistently". So it must be guaranteed that no partner resides in an "old" data phase while the other one enters a "new" one. This is formally specified by a complex product net in /Oc5/ using the product net machine /Oc3/ and a reachability graph with 370 markings and 1083 occurrence steps has been computed by this tool.

Looking at the reachability graph special dynamical properties of our specification can be investigated, for example the just mentioned separation of data phases. Taking into account that A and B enter and leave data phases by the occurrence of “special” transitions this separation property could be investigated using a relabelling of the reachability graph: The “special” transitions become AD (A disconnect), BD (B disconnect), AC (A connect) or BC (B connect) and all other labels are replaced by τ.

Instead of looking at such a labelled transition system we prefer a language theoretic approach. Let Γ be the set of all occurrence steps of the product net (described by triples (M,t,M’), where M’ is the successor marking of M by the occurrence of t) and let the occurrence language $L \subseteq \Gamma^*$ be the set of all occurrence sequences (paths in the reachability graph) starting with the initial marking. L is a unique representation of the reachability graph and the labelled transition system described above corresponds to a language homomorphism $s : \Gamma^* \rightarrow \Sigma^*$ with $\Sigma = \{AC, AD, BC, BD\}$.

Now the homomorphic image s(L) is the language of the labelled transition system. As the reachability graph is finite s(L) is a regular set. The minimal automaton of s(L) (computed by the product net machine) consists of 14 states. By inspecting this automaton it is easy to see that $\{y \in (\Sigma\setminus\{AD\})^* \mid xACy \in s(L)\} \subseteq \{\varepsilon, BD\}$ and $\{y \in (\Sigma\setminus\{BD\})^* \mid xBCy \in s(L)\} \subseteq \{\varepsilon, AC, AD, AC\,AD, AD\,AD, AC\,AD\,AD\}$ for each $x \in s(L)$.

The first inclusion implies that B does not enter a data phase as long as A is in a data phase; however B may have entered a data phase before. This means that B does not enter a "second" data phase while A is in a data phase. Additionally the second inclusion implies that A passes through at most one data phase as long as B is in a data phase. Both properties together exactly mean "global" separation of successive data phases of Aand B.

These properties are typical safety properties. Their purpose is to prevent some bad thing happening in the sequence of actions. More precisely: If $F \subseteq \Sigma^*$ denotes the set of sequences in Σ^* with certain bad properties then it has been proven that $s(L) \cap F = \emptyset$, which implies $L \cap s^{-1}(F) = \emptyset$ ($s^{-1}(F) = \{ w \in \Gamma^* \mid s(w) \in F \}$). This means that there does not exist an occurrence sequence $w \in L$ with certain corresponding bad properties represented by $s^{-1}(F)$. So if a safety property is proven for an image s(L) its corresponding property is true for the language L.

Using the simplicity of s the following liveness property has been proven in /Oc5/: again and again a situation is reachable in which A and B are in a data phase. In the same way we now investigate a liveness property of a simple client-server-example (Fig 3.3). Arequest of a client is reject or responded by a server depending on resources.

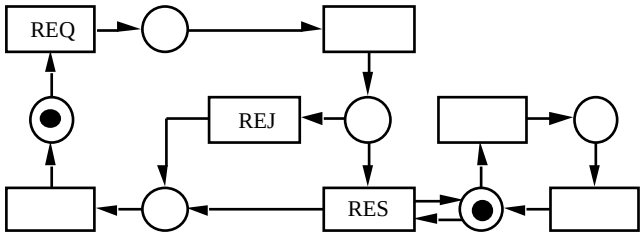

Fig. 3.3

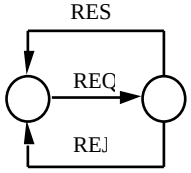

Fig. 3.4

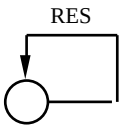

Fig. 3.5

If $L \subseteq \Gamma^*$ is the occurrence language of the net, the labelling of the transitions in fig. 3.3 can be described by a homomorphism $s : \Gamma^* \rightarrow \Sigma^*$ with $\Sigma = \{REQ, RES, REJ\}$. Fig. 3.4 shows the minimal automaton of s(L). Inspecting this automaton it is easy to see that for each $x \in s(L)$ there exists $y \in \Sigma^*$ with $xyRES \in s(L)$. This is a liveness property of s(L).

The reachability graph of our net is strongly connected. So by theorem 4 the homomorphism s is simple on L.

Therefore for each u L there exists v s(u)$^{-1}$(s(L)) with v^{-1}(s(u^{-1}(L))) = v^{-1}(s(u)$^{-1}$(s(L))) = (s(u)v)$^{-1}$(s(L)) . On account of the proven liveness property of s(L) (x = s(u)v) there exists y Σ* with yRES (s(u)v)$^{-1}$(s(L)) = v^{-1}(s(u^{-1}(L))) . This implies vyRES s(u^{-1}(L)) . So there exists w u^{-1}(L) with s(w) = vyRES . On account of this for each u L there exist w´ Γ* and markings M and M´ with uw´(M,RES,M´) L . Now for the specified system this means that whatever happened before a situation is reachable in which a request is responded.

This property could also be proven directly by the strong connectness of the reachability graph, but in general we don't have this simple situation. Experiences have shown /Gi,Ne,Oc7/ that in such cases liveness properties can be proved as demonstrated above by theorem 4 using the product net machine.

If we omit in our net the transition returning the resource then theorem 4 fails. Moreover, the reachability graph contains a dead component with no edge labelled by RES. To describe this situation in our notation we define the homomorphism t : Γ* → Σ´* (Σ´= {RES}) by t((M,RES,M´)) = RES and t((M,X,M´)) = ε for X ≠ RES. Fig 3.5 shows the minimal automaton of the deadlock language DL corresponding to t (computed by the product net machine).

This automaton is also the minimal automaton of t(L). Therefore the termination language TL with respect to t is empty. By our considerations in chapter 2 t is not simple on L because DL ≠ TL . For investigating the simplicity of s we define the homomorphism t´ : Σ* → Σ´* by t´(RES) = RES and t´(X) = ε for X ≠ RES . This definition implies t = t´ o s . By theorem 4 t´ is simple on s(L) because the automaton in fig. 3.4 is strongly connected. Now on account of theorem 1 s is not simple on L. So the defect of our faulty specification can be detected by simplicity investigations of appropriate homomorphisms without using the complex decision algorithm for simplicity.

4. Conclusions

In this paper the notion of simplicity of language homomorphisms and labelled transition systems have been defined. It has been shown that for simple labelled transition systems not only safety but also liveness properties of their language imply corresponding properties of the system itself. Using the product net machine such properties have been verified by this method, and errors in a faulty specification have been detected.

We used the terminology of formal language theory to formulate safety and liveness properties. Usually temporal logic is used for this purpose /CES/. Assuming simlicity of language homomorphisms in /Ni2,Ni3,Ni4,Ni5/ the translation of properties from the homomorphic image back to the language itself is investigated in the context of temporal logic.

The presented verification method depends on finiteness of the reachability graph. It is known /Ha/ that in case of place transition nets this property is decidable. But for product nets /Ni1/ as well as for other powerful specification languages finiteness of the reachability graph is undecidable. However a lot of distributed system specifications admit a level of abstraction with a finite reachability graph where interesting properties of the system can be investigated. So in defining product nets importance has been attached to the computability of reachability graphs if they are finite /BOP/.

A well known problem concerning reachability graphs is state explosion. For special classes of homomorphisms in /Oc1,Oc2,Oc4/ reduced reachability graphs are therefore defined. They contain enough information to compute the corresponding minimal automata and to investigate the simplicity of the homomorphism /Oc5/. In /Oc8/ a compositioal method is developed to compute minimal automata and preserve simplicity of homomorphisms. Assuming "sufficiently well structured specifications" this method makes the product net machine practicable even in case of astronomical reachability graphs.

Acknowledgements

For interesting discussions on the topics dealt with in this paper I thank my colleagues Uli Nitsche, Wolfgang Orth and Jürgen Repp.

References

/ACW/ S. Aggarwal, C. Coucourbetis, P. Wolper: Adding Liveness Properties to Coupled Finite State Machines
ACM Transactions on Programming Languages and Systems 12 1990

/AS1/ B. Alpern, F.B. Schneider: Defining Liveness
Information Processing Letters 24 1985

/AS2/ B. Alpern, F.B. Schneider: Verifying Temporal Properties without Temporal Logic
ACM Transactions on Programming Languages 11 1989

/BW/ J.C.M. Baeten, W.P. Weijland: Process Algebra
Cambridge University Press 1990

/Be/ J. Berstel: Finite automata and rational languages. An introduction
LITP Spring School on Theoretical Computer Science 1988 LNCS 386

/BOP/ H.J. Burkhardt, P. Ochsenschläger, R. Prinoth: Product Nets
AFormal Description Technique for Cooperating Systems
GMD - Studien 165 1989

/CES/ E.M. Clarke, E.A. Emerson, A.P. Sistla:
Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications
ACM TOPLAS 8 1986

/Gi/ H. Giehl
Verifikation von Smartcard-Anwendungen mittels Produktnetzen
GMD-Studien 225 1993

/Ha/ M. Hack: Petri Net Languages
Laboratory for Computer Science MIT T.R.161 1976

/Mi/ R. Milner: Operational and Algebraic Semantics of Concurrent Processes
Handbook of Theoretical Computer Science Vol. B Elsevier 1990

/Ne/ M. Nebel: Ein Produktnetz zur Verifikation von SmartCard-Anwendungen in der STARCOS-Umgebung
GMD-Studien 234 1994

/Ni1/ U. Nitsche: Erreichbarkeitsanalyse von Produktnetzen
Arbeitspapiere der GMD 521 1991

/Ni2/ U. Nitsche: Propositional Linear Temporal Logic and Language Homomorphisms
Logical Foundations of Computer Science St. Petersburg 1994
Lecture Notes in Computer Science Springer Verlag

/Ni3/ U. Nitsche: AVerification Method Based on Homomorphic Model Abstraction
ACM Symposium on Principles of Distributed Computing Los Angeles 1994 ACM Press

/Ni4/ U. Nitsche: Verifying Temporal Logic Formulas in Abstractions of Large Reachability Graphs
this Workshop

/Ni5/ U. Nitsche: Simple Homomorphisms and Linear Temporal Logic
Arbeitspapiere der GMD 1994

/Oc1/ P. Ochsenschläger: Projektionen und reduzierte Erreichbarkeitsgraphen
Arbeitspapiere der GMD 349 1988

/Oc2/ P. Ochsenschläger: Modulhomomorphismen
Arbeitspapiere der GMD 494 1990

/Oc3/ P. Ochsenschläger: Die Produktnetzmaschine
Petri Net Newsletter 39 1991

/Oc4/ P. Ochsenschläger: Modulhomomorphismen II
Arbeitspapiere der GMD 597 1991

/Oc5/ P. Ochsenschläger: Verifikation kooperierender Systeme mittels schlichter Homomorphismen
Arbeitspapiere der GMD 688 1992

/Oc6/ P. Ochsenschläger: Verifikation verteilter Systeme mit Produktnetzen
PIK 16 (1993) 42 - 43

/Oc7/ P. Ochsenschläger: Verifikation von SmartCard-Anwendungen mit Produktnetzen
4. GMD-SmartCard Workshop Darmstadt 1994

/Oc8/ P. Ochsenschläger: Kompositionelle Verifikation kooperierender Systeme
Arbeitspapiere der GMD 1994

Peh/ B. Pehrson: Protocol Verification for OSI
Computer Networks and ISDN Systems 18 1989/90

/Per/ D. Perrin: Finite Automata
Handbook of Theoretical Computer Science Vol. B Elsevier 1990

/Ta/ D. Taubner: Finite Representations of CCS and TCSP Programs by Automata and Petri Nets
LNCS 369 1989

This paper is a contribution to the GI Workshop:

Algorithmen und Werkzeuge für Petrinetze Humboldt-Universität Berlin 1994

Dr. Peter Ochsenschläger
Institut für Telekooperationstechnik der GMD
Rheinstr. 75
D-64295 Darmstadt

Tel. +49 6151 869-283
Fax +49 6151 869-224
e-mail ochsenschlaeger@darmstadt.gmd.de