Compositional Verification of Cooperating Systems Using Simple Homomorphisms

Peter Ochsenschläger
Institut für Telekooperationstechnik der GMD

1. Introduction

Investigating distributed systems one is faced with the problem, that the interplay of a few simple components may cause a very complex dynamical behaviour of the complete system. In contrast to this for the verification of certain properties of the system often only a few "significant actions" have to be considered in such a complex dynamics. In constructing a complete specification out of components it is therefore obvious to look for more "abstract" versions of the specification, which show an "equivalent behaviour" with respect to "significant actions". By that each component can be investigated in a "simplified enviroment". This is the usual approach in algebraic specification methods, as for example CCS or CSP /BW,Mi/. It has also been applied to petri nets in /Ta,Vo/. Such compositional methods are essentially imprinted by a formal definition of the notion "equivalent behaviour".

Concerning cooperating systems - as we call such systems that are characterized by freedom of decisions and loose coupling of the cooperating partners - it has been shown /Oc5,8,OP/ that equivalence notions as defined in the area of the above mentioned process algebras are too fine. There exist specifications, which are equivalent in an intuitive sense but not in a formal one, as defined there. Moreover this implies that in a compositional approach not in all cases "smallest abstract versions" of a specification can be found. Therefore in /Oc5,8,OP/ equality of the sets of all possible "sequences of actions" (pathes of a labelled transition system) has been used as a formal equivalence notion (language equivalence). This is a very simple equivalence notion, which is of course coarser than the equivalence notions mentioned above. Since with respect to abstractions (described by alphabetic language homomorphisms) it is not sensitive to deadlocks and does not reflect liveness properties /AS/ additionally the notion of simplicity of abstractions has been defined. This additional property of abstractions removes the described defect and makes language equivalence a flexible instrument for verifying cooperating systems.

So for finite state systems minimal automata of homomorphic images form compact, unique and adequate representations of abstractions of system behaviour. They can be used to compare specifications of different levels of abstraction as well as to prove special safety and liveness properties of a specification /Gi,Ne,Oc3,5,6,7,8,9,OP/. This verification approach, which only operates with notions of formal language theory, is the basis of the compositional method developed in this paper. The main goal is to compute minimal automata of homomorphic images efficiently even in case of complex specifications. The fundamental idea is to embed each component of a complex system in a "simplified enviroment", which shows at the interface an "equivalent behaviour" compared to the rest of the system. For each of these "smaller" systems "corresponding" minimal automata are computed and are "composed" to obtain the desired automaton. By that, compact representations of abstractions of system behaviour can be computed without investigating the complete behaviour of a complex system. In case of "well structured" specifications this method causes considerable reductions of the state spaces to be investigated. The smaller systems with "simplified enviroments" avoid a lot of interleavings of actions ("state space explosion"), which are not relevant with respect to the considered abstraction but which are instrumental in the complex dynamics of the system, as mentioned at the beginning.

As a formal basis for specifications the notion of asynchronous product automata (APA) is defined, a very general class of communicating automata. This is not to enlarge the wide area of existing specification languages but to develop the compositional verification method as general as possible. So our approach becomes applicable to a lot of specification techniques as for example different kinds of "simple" and "higher order" petri nets or "extended" automata models as for example SDL or Estelle.

In chapter 2 asynchronous product automata are defined and a formal method is developed to "compose" representations of abstractions of system behaviour and to decide if a "simplified enviroment" shows at the interface an "equivalent behaviour" compared to the rest of the system. In chapter 3 this method is applied to the verification of a connection establishment and release protocol. Thereby an enormous reduction of complexity is obtained.

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

2. Asynchronous Product Automata

As a formal basis for distributed systems specifications we now define the notion of asynchronous product automata (APA), a very general class of communicating automata. APA can be regarded as families of automata (elementary automata, each consisting of one transition), whose sets of states are cartesian products and whose elementary automata are

"glued together" by common components of these products. An asynchronous product automaton (APA) consists of a family of sets of state components $(Z_s)_{s \in S}$, a family of transition triples $(I_t,i_t,\delta_t)_{t \in T}$ and a neighbourhood relation $N : \mathbb{T} \rightarrow (\mathbb{S})$ ((X) denotes the set of all subsets of X). The indices $t \in \mathbb{T}$ are called transitions, and the indices $s \in \mathbb{S}$ are called state components.
For each transition $t \in \mathbb{T}$

- I_t is the set of possible interpretations of t,
- $i_t : \times_{s \in N(t)} (Z_s) \rightarrow (I_t)$ is the interpretation function of t and
- $\delta_t : I_t \times \times_{s \in N(t)} (Z_s) \rightarrow \times_{s \in N(t)} (Z_s)$ is the partial state transition function of t.

To avoid pathological cases we assume $\mathbb{S} = \bigcup_{t \in T} (N(t))$ and $N(t) \neq \emptyset$ for all $t \in \mathbb{T}$. The states of an APA are elements of $\times_{s \in S} (Z_s)$ with the initial state $q_0 = (q_{0s})_{s \in S} \in \times_{s \in S} (Z_s)$. Formally an APA $\mathbb{A}$ is defined by a quadruple $\mathbb{A} = ((Z_s)_{s \in S} , (I_t,i_t,\delta_t)_{t \in T} , N , q_0)$.

Interpretations in the APA-concept permit parametrizations of behaviour patterns and allow "controlable nondeterminism" on the level of transitions. They could also be omitted, because they do not enlarge the expressiveness principally. But as they essentially reduce the description complexity /Ni1/ and as they allow well structured compact specifications, which is very important in practical use of a specification method, they are added to the APA-concept. Apart from these interpretations APA are similar to asynchronous cellular automata introduced in /Zi/.

"Dynamics" of APA are defined by "occurrences" of transitions. A transition $t \in \mathbb{T}$ is activated in a state $p = (p_s)_{s \in S} \in \times_{s \in S} (Z_s)$ referring to an interpretation $v \in I_t$, if $v \in i_t((p_s)_{s \in N(t)})$ and $\delta_t(v,(p_s)_{s \in N(t)})$ is defined. An activated transition t may occur and generates the successor state $q = (q_r)_{r \in S}$ with $q_r = (\delta_t(v,(p_s)_{s \in N(t)}))_r$ for $r \in N(t)$ and $q_r = p_r$ for $r \in \mathbb{S} \setminus N(t)$. In this case (p,t,v,q) and (p,t,q) respectively denotes the corresponding occurrence step with or without mentioning the interpretation. Existence of an occurrence step (p,t,q) means that there exists an interpretation $v \in I_t$, such that t is activated in p referring to v and its occurrence generates the successor state q. For simplicity in the following we only consider occurrence steps without mentioning the interpretation, because our investigations are independent of interpretations.

A sequence of the form $w = (q_1,t_1,q_2)(q_2,t_2,q_3)...(q_k,t_k,q_{k+1})(q_{k+1},t_{k+1},q_{k+2})...(q_n,t_n,q_{n+1})$ with $n \geq 1$ is called an occurrence sequence. If such an occurrence sequence exist, then we say that q_{n+1} is reachable from q_1. Additionally by definition each state is reachable from itself. $\mathbb{Q}$ (state space) denotes the set of all states $q \in \times_{s \in S} (Z_s)$ reachable from the initial state q_0 and Σ denotes the set of all occurrence steps, whose first component is an element of $\mathbb{Q}$. The set $L \subseteq \Sigma^*$ of all occurrence sequences starting with the initial state q_0 and containing the empty sequence ε is called the occurrence language of the corresponding APA. Σ can also be interpreted as the set of arcs of a directed graph, whose set of nodes is $\mathbb{Q}$ and whose arcs are labelled by elements of $\mathbb{T}$. This graph is called the reachability graph of the corresponding APA. By that occurrence sequences are pathes in the reachability graph and the occurrence language is regular (local set) /Pe/, if the reachability graph is finite. The occurrence language as well as the reachability graph is a complete description of the dynamical behaviour of an APA.

APA form an access to a general verification concept which is based on simple homomorphisms and allows different syntactical interfaces. Among these are different kinds of "simple" and "higher order" petri nets as well as communicating automata as for example SDL /SSR/ or Estelle /BD/. The above terminology is based on petri nets. State components of APA correspond to places and states correspond to markings of places. By that the state transition function is realized by the so called occurrence rule of a petri net.

In terms of APA we now formulate our compositional method: A distributed system is an APA and the dynamical behaviour of the system is described by the occurrence language of that APA. A component (subsystem, module) of a system is defined by a subset $A \subseteq \mathbb{T}$ of the set of transitions of the APA. As we often consider the complement of A ("rest of the system" with respect to A) we use the abbreviation A' for $\mathbb{T} \setminus A$. To consider states of an APA restricted to a subset $Y \subseteq \mathbb{S}$ we define $q|Y = (q_s)_{s \in Y} \in \times_{s \in Y} (Z_s)$ for a state $q = (q_s)_{s \in S} \in \times_{s \in S} (Z_s)$. Two special homomorphisms M_A (module homomorphism) and R_A (boundary homomorphism) on the occurrence language $L \subseteq \Sigma^*$ of an APA are used to express the behaviour of a component A of an APA and its behaviour at the interface to A' respectively.

Let $RDA = N(A) \cap N(A')$. By $M_A((p,t,q)) = (p|N(A),t,q|N(A))$ if $t \in A$ and $N(t) \cap RDA \neq \emptyset$, $M_A((p,t,q)) = (p|N(A)\backslash RDA,t,q|N(A)\backslash RDA)$ if $t \in A$ and $N(t) \cap RDA = \emptyset$ and $M_A((p,t,q)) = \varepsilon$ if $t \in A'$ a homomorphism $M_A : \Sigma^* \rightarrow \Sigma_{MA}^*$ is defined with $\Sigma_{MA} = M_A(\Sigma)$. By $R_A((p,t,q)) = (p|RDA,q|RDA)$ if $t \in A$ and $N(t) \cap RDA \neq \emptyset$ and

$R_A((p,t,q)) = \varepsilon$ if $t \in A'$ or $N(t) \cap RDA = \emptyset$ a homomorphism $R_A : \Sigma^* \to \Sigma_{RA}^*$ is defined with $\Sigma_{RA} = R_A(\Sigma)$.

To compare homomorphisms with respect to their "degree of abstraction" we call a homomorphism $\varphi : \Sigma^* \to \Delta^*$ finer than a homomorphism $\psi : \Sigma^* \to \Gamma^*$, if there exists a homomorphism $\mu : \Delta^* \to \Gamma^*$ with $\psi = \mu \circ \varphi$. For this we use the notation $\varphi \geq \psi$. In that case the homomorphic image $\varphi(L)$ contains enough "information" to determine $\psi(L)$. As homomorphisms are used to describe abstractions we assume that they are alphabetic, which means $\varphi(\Sigma) \subseteq \Delta \cup \{\varepsilon\}$ for each homomorphism φ.

Two homomorphisms "acting" on disjoint components of an APA can be "combined" obtaining a new homomorphism: Let $A \subseteq \mathbb{T}$ and let $f : \Sigma^* \to \Phi^*$ as well as $g : \Sigma^* \to \Gamma^*$ be homomorphisms with $M_A \geq f$ as well as $M_{A'} \geq g$, then the homomorphism $f \oplus g : \Sigma^* \to (\Phi \cup \Gamma)^*$ is defined by $(f \oplus g)((p,t,q)) = f((p,t,q))$ if $t \in A$ and $(f \oplus g)((p,t,q)) = g((p,t,q))$ if $t \in A'$. $f \oplus g$ is called the direct sum of f and g. By the additional assumption $\Phi \cap \Gamma = \emptyset$ the direct sum of two homomorphisms is finer than both homomorphisms: There exist homomorphisms (projections) $\varphi : (\Phi \cup \Gamma)^* \to \Phi^*$ and $\gamma : (\Phi \cup \Gamma)^* \to \Gamma^*$ with $f = \varphi \circ (f \oplus g)$ and $g = \gamma \circ (f \oplus g)$.

The homomorphic image $(f \oplus g)(L)$ can be "constructed" using f(L) and g(L) if these two images contain enough information about the boundary behaviour of A and A' respectively, which means that $f \geq R_A$ and $g \geq R_{A'}$. To formulate a corresponding theorem we need some further technical notions: If $f \geq R_A$ and $g \geq R_{A'}$, then there exist homomorphisms $\rho_A : \Phi^* \to \Sigma_{RA}^*$ and $\rho_{A'} : \Gamma^* \to \Sigma_{RA'}^*$ with $R_A = \rho_A \circ f$ and $R_{A'} = \rho_{A'} \circ g$. Let $\Sigma_R = \Sigma_{RA} \cup \Sigma_{RA'}$ and let $\rho : (\Phi \cup \Gamma)^* \to \Sigma_R^*$ be the homomorphism defined by: $\rho(x) = \rho_A(x)$ if $x \in \Phi$ and $\rho(x) = \rho_{A'}(x)$ if $x \in \Gamma$. Let $RC = \{ z \in \Sigma_R^* \mid$ if $z = (p,q)y$ with $(p,q) \in \Sigma_R$ and $y \in \Sigma_R^*$, then $p = q_0|RDA$, and if $z = x(p,q)(p',q')y$ with $(p,q),(p',q') \in \Sigma_R$ and $x,y \in \Sigma_R^*$, then $p' = q \}$. If Σ_R is finite, then RC is a regular set (local set /Pe/). The definition of RC depends on Σ_R. On account of $f \geq R_A$ and $g \geq R_{A'}$ this set can be determined using f(L) and g(L).

Theorem 1: Let $L \subseteq \Sigma^*$ be the occurrence language of an APA and $A \subseteq \mathbb{T}$. If $f : \Sigma^* \to \Phi^*$ and $g : \Sigma^* \to \Gamma^*$ are homomorphisms with $\Phi \cap \Gamma = \emptyset$ and $M_A \geq f \oplus R_A$ as well as $M_{A'} \geq g \oplus R_{A'}$, then $(f \oplus g)(L) = \varphi^{-1}(f(L)) \cap \gamma^{-1}(g(L)) \cap \rho^{-1}(RC)$.

By this representation $(f \oplus g)(L)$ is a regular set if L is regular. It is easy to construct a finite automaton recognizing $(f \oplus g)(L)$ using corresponding automata for f(L) and g(L). Concerning simplicity of $f \oplus g$ we have

Theorem 2: If f and g are simple on L by the same assumptions as in theorem 1, then $f \oplus g$ is simple on L too.

The proofs of these two theorems as well as the proofs of theorem 3, 4 and 5 can be found in /Oc9,10/. Essential to the statements of this chapter is "locality" of occurrence steps, which means that state changes only occur in the neighbourhood of the corresponding transition. Therefore occurrence sequences may be "rearranged" without changing certain homomorphic images. Such "rearrangements" are the main proof techniques for theorems 1 - 5.

Theorem 1 and 2 form one half of our compositional method. They show how abstractions of the behaviour of components of an APA can be "composed". But so far the representation of $(f \oplus g)(L)$ depends on f(L) and g(L). How can f(L) and g(L) be determined without using the (complex) occurrence language L of the complete system? To achieve this we "embed" the components A and A' in "simplified enviroments". Since a component of an APA can be viewed as an APA too we now have to define how two APA can be "composed".

The "glueing together" of elementary automata mentioned in the definition of APA can also be applied to arbitrary APA. Let therefor $\mathbb{A}k = ((Zk_s)_{s \in \mathbb{S}k}, (Ik_t,ik_t,\delta k_t)_{t \in \mathbb{T}k}, Nk, qk_0)$ with $k \in \{1,2\}$ be two APA with $\mathbb{T}1 \cap \mathbb{T}2 = \emptyset$ and $Z1_s = Z2_s$ as well as $q1_{0s} = q2_{0s}$ for all $s \in \mathbb{S}1 \cap \mathbb{S}2$. Now the direct sum $\mathbb{A}1 \oplus \mathbb{A}2$ is defined by $\mathbb{A}1 \oplus \mathbb{A}2 = ((Z_s)_{s \in \mathbb{S}}, (I_t,i_t,\delta_t)_{t \in \mathbb{T}}, N, q_0)$ with $\mathbb{S} = \mathbb{S}1 \cup \mathbb{S}2$, $\mathbb{T} = \mathbb{T}1 \cup \mathbb{T}2$, $Z_s = Zk_s$ and $q_{0s} = qk_{0s}$ for all $s \in \mathbb{S}k$ and $(I_t,i_t,\delta_t) = (Ik_t,ik_t,\delta k_t)$ for all $t \in \mathbb{T}k$ and $N(t) = Nk(t)$, where $k \in \{1,2\}$. We also say that $\mathbb{A}1 \oplus \mathbb{A}2$ is constructed from $\mathbb{A}1$ and $\mathbb{A}2$ by glueing together the common state components $\mathbb{S}1 \cap \mathbb{S}2$.

If A and A' are complementary components of an APA, then this APA is the direct sum of A and A'. In terms of boundary behaviour the following theorem gives a sufficient condition to "embed" a component of an APA in different "enviroments" without changing its behaviour.

Let $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$ and $\mathbb{D}$ be four APA, for which the direct sums $\mathbb{A} \oplus \mathbb{B}$, $\mathbb{C} \oplus \mathbb{D}$ and $\mathbb{A} \oplus \mathbb{D}$ are defined. Let A, B, C and D are the corresponding sets of transitions and SA, SB, SC and SD the sets of their state components. Additionally we assume that $SA \cap SB = SC \cap SD = SA \cap SD$. LAB, LCD as well as LAD may denote the occurrence languages of $\mathbb{A} \oplus \mathbb{B}$, $\mathbb{C} \oplus \mathbb{D}$ and $\mathbb{A} \oplus \mathbb{D}$ respectively.

Theorem 3: If $R_A(LAB) = R_C(LCD)$ and $R_B(LAB) = R_D(LCD)$, then $M_A(LAD) = M_A(LAB)$ and $M_D(LAD) =$

$M_D(LCD)$.

Let $\mathbb{A}$ $\mathbb{D}$ be a representation of the APA considered in theorem 1 and 2, then D = A' and L = LAD . If B and C are "simplified versions" of D and A respectively then LAB and LCD can be "less complex" (with an essentially smaller state space) than L as demonstrated in the next section. Now applying theorem 3 f(L) and g(L) can be determined using LAB and LCD instead of L because M_A f and $M_{A'}$ g . To derive simplicity of homomorphisms on L from investigations on LAB and LCD we need a "cooperating property" of APA/Oc10/:

Let LAB Σ^* be the occurrence language of $\mathbb{A}$ $\mathbb{B}$ and let $f : \Sigma^* \to \Phi^*$ be a homomorphism. $\mathbb{A}$ is called cooperative in $\mathbb{A}$ _ $\mathbb{B}$ with respect to f, if M_A f , $\Phi \cap M_B(\Sigma) =$ and if for each x LAB there exists a finite subset H $(f\ M_B)(x)^{-1}((f\ M_B)(LAB))$ with ε H such that for each u H either $u^{-1}((f\ M_B)(x^{-1}(LAB))) = ((f\ M_B)(x)u)^{-1}((f\ M_B)(LAB))$ or $u^{-1}(H) \cap M_B(\Sigma) = ((f\ M_B)(x)u)^{-1}((f\ M_B)(LAB)) \cap M_B(\Sigma)$ and $u^{-1}(H) \cap \Phi \neq$ if $((f\ M_B)(x)u)^{-1}((f\ M_B)(LAB)) \cap \Phi \neq$.

In combination with theorem 3 the following two theorems /Oc10/ allow to derive simplicity of f and g on L from investigations on LAB and LCD.

Theorem 4: Let r and s be homomorphisms with M_A r R_A , M_D s R_D , $r(LAB) = M_C(LCD)$ and $s(LCD) = M_B(LAB)$. If $\mathbb{A}$ is cooperative in $\mathbb{A}$ $\mathbb{B}$ with respect to r and if $\mathbb{D}$ is cooperative in $\mathbb{C}$ $\mathbb{D}$ with respect to s, then r s is simple on LAD.

Theorem 5: Let r and s be homomorphisms with M_A r R_A , M_D s . If r s is simple on LAD then M_A s is simple on LAD.

3. Verification of a Connect-Disconnect-Protocol

As an application of the definitions and results of chapter 2 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 as depicted in fig. 3.1.

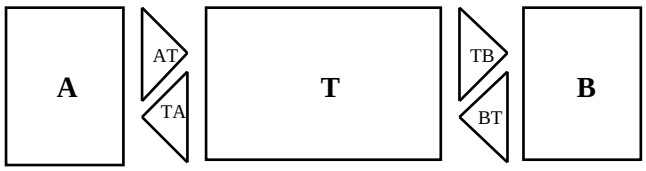

Fig. 3.1

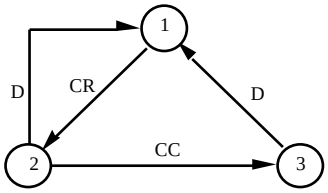

Fig. 3.2

If A wants to establish a connection to B it sends a request (CR) to T. T may refuse (D) this request 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 or confirm (CC) it. In both cases T transmits the answer to A. In case of confirmation B enters its data phase, which means that now B is ready to receive data. Receiving the confirmation A enters its data phase. Then A may send data. Each of A, B and T is free to release the connection (D) independently and at any time i. e. not only in the data phase but also in the connection establishment phase. At each interface of the cooperating partners one can see the pattern of behaviour as described in fig. 3.2.

In /Oc5,9,OP/ this protocol is formally specified by a complex product net. For verification a homomorphism s is defined on the occurrence language L of this product net. s "observes" A's and B's entering (AC, BC) and leaving (AD, BD) of data phases and "ignores" all other occurrence steps by mapping them to ε. Fig. 3.3 shows the minimal automaton of s(L), computed by the product net machine /Oc3/.

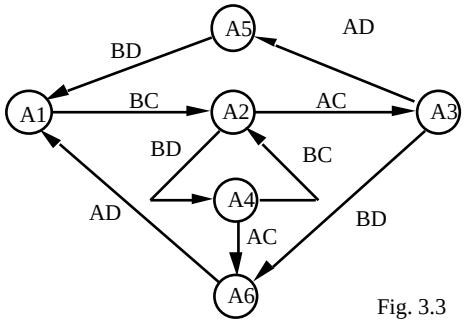

Fig. 3.3

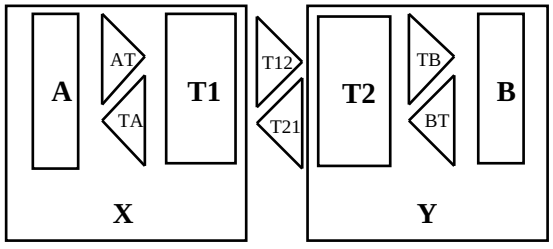

Fig. 3.4

Fig. 3.3 shows that no partner resides in an "old" data phase while the other one enters a "new" one (a safety property), which mean that connections are established and released "consistently" (global separation of data phases). Using

simplicity of s this automaton also proves that again and again A and B may enter a “common” data phase (a liveness property).

These properties deduced from s(L) also hold for other specifications with the same homomorphic image L’ of their occurrence language under a corresponding simple homomorphism. This can be used to verify further refinements of our protocol specification. Let us therefor consider the more realistic specification of fig. 3.4 where the transport system consists of two identical “local” instances coupled by FIFO-queues instead of one “global” transport module.

The reachability graph of this specification consists of 5663 nodes. It is far larger than the reachability graph of the “small” specification (370 nodes). Using the compositional method of the preceding chapter the minimal automaton of L’ can be computed and simplicity of the corresponding homomorphism can be shown without referring to the large reachability graph /Oc9,10/. For this we devide our specification into two parts, X and Y, as depicted in fig. 3.4. By theorem 3 it can be proven that B and A are “simplified versions” of Y and X (see fig. 3.5). To apply theorem 1 and 2 two homomorphisms f and g are defined such that f g s . f and g filter the corresponding information out of A and B respectively and additionally describe the boundary behaviour of X and Y respectively. Now by the results of chapter 2 for f g a corresponding minimal automaton (171 states) can be computed and simplicity can be proven only using the small reachability graph. From this the desired results corresponding to s can be derived (see /Oc5,8,OP/) avoiding the investigation of the large reachability graph (5663 nodes).

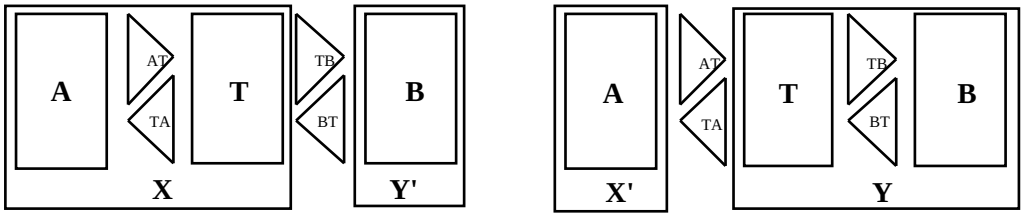

Fig. 3.5

This approach can be developed to an induction proof for a corresponding system with an arbitrary number of identical transport modules /Oc9,10/. Also that induction proof only uses the small reachability graph (370 nodes) to compute the minimal automaton (fig. 3.3) and to show the simplicity of the corresponding homomorphism. To demonstrate the power of our approach it may be mentioned that the reachability graph of a system with three transport modules consists of more than 85000 nodes.

4. Conclusions

Based on the simplicity of language homomorphisms /Oc5,8,OP/ and on asynchronous product automata a compositional method for the verification of cooperating systems is developed. Considering a connection establishment and release protocol it is demonstrated, how the dynamics of a complex specification can be investigated without computing the reachability graph of the complete system. So the well known “state space explosion” is avoided. In earlier papers /Oc1,2,4/ for special classes of homomorphisms so called reduced reachbility graphs were introduced to reduce the complexity of verification investigations. Now the presented method obtains such reductions of state spaces in a far more flexible and efficient manner, which makes corresponding tools usable for the verification of systems of industrial size.

The specification of our example shows some symmetries, which permit an induction proof and facilitate the representation of “simplified components” X’ and Y’ with the same “boundary behaviour”. Such symmetries are in no way necessary requirements for our method. In general the development of “simplified components” of a specification with the same “boundary behaviour” is an essential part of designing well structured systems, since it reflects the information-hiding at interfaces.

We use the terminology of formal language theory to formulate safety and liveness properties of the connection establishment and release protocol. Usually such properties are expressed by temporal logic /CES/. Inspired by the language theoretic approach the translation of properties from the homomorphic image back to the language itself is investigated in the context of temporal logic in /Ni2,Ni3,Ni4,Ni5/. To treat liveness properties in such a manner simplicity of homomorphisms is essential. By these translations our compositional method becomes applicable to temporal logic and by corresponding assumptions it reduces the complexity of model checking comparable to /GW/.

References

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

/BD/ S. Budkowski, P. Dembinski: An Introduction to Estelle
Computer Networks 14 1987

/BW/ J.C.M. Baeten, W.P. Weijland: Process Algebra

Cambridge University Press 1990

/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

/GW/ P. Godefroid, P. Wolper: Partial-Order Methods for Temporal Verification
Concur 1993 LNCS 715 Springer Verlag

/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 813 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
Workshop Algorithmen und Werkzeuge für Petrinetze Berlin 1994

/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:
Verification of Cooperating Systems by Simple Homomorphisms Using the Product Net Machine
Workshop Algorithmen und Werkzeuge für Petrinetze Berlin 1994

/Oc9/ P. Ochsenschläger: Kompositionelle Verifikation kooperierender Systeme
Arbeitspapiere der GMD 885 1994

/Oc10/ P. Ochsenschläger: Kooperationsprodukte formaler Sprachen (to appear 1995/96)

/OP/ P. Ochsenschläger, R. Prinoth: Modellierung verteilter Systeme
Konzeption, formale Spezifikation und Verifikation mit Produktnetzen
Vieweg Verlag 1995

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

/SSR/ R. Saracco, J. R. W. Smith, R. Reed: Telecommunications Systems Engineering using SDL
North Holland 1989

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

/Vo/ W. Vogler: Modular Construction and Partial Order Semantics of Petri Nets
LNCS 625 Springer Verlag 1992

/Zi/ W. Zielonka: Safe Executions of Recognizable Trace Languages by Asynchronous Automata
Logical Foundations of Computer Science 1989 LNCS 363 Springer Verlag

This paper is a contribution to the GI Workshop:

Cambridge University Press 1990

/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

/GW/ P. Godefroid, P. Wolper: Partial-Order Methods for Temporal Verification
Concur 1993 LNCS 715 Springer Verlag

/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 813 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
Workshop Algorithmen und Werkzeuge für Petrinetze Berlin 1994

/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:
Verification of Cooperating Systems by Simple Homomorphisms Using the Product Net Machine
Workshop Algorithmen und Werkzeuge für Petrinetze Berlin 1994

/Oc9/ P. Ochsenschläger: Kompositionelle Verifikation kooperierender Systeme
Arbeitspapiere der GMD 885 1994

/Oc10/ P. Ochsenschläger: Kooperationsprodukte formaler Sprachen (to appear 1995/96)

/OP/ P. Ochsenschläger, R. Prinoth: Modellierung verteilter Systeme
Konzeption, formale Spezifikation und Verifikation mit Produktnetzen
Vieweg Verlag 1995

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

/SSR/ R. Saracco, J. R. W. Smith, R. Reed: Telecommunications Systems Engineering using SDL
North Holland 1989

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

/Vo/ W. Vogler: Modular Construction and Partial Order Semantics of Petri Nets
LNCS 625 Springer Verlag 1992

/Zi/ W. Zielonka: Safe Executions of Recognizable Trace Languages by Asynchronous Automata
Logical Foundations of Computer Science 1989 LNCS 363 Springer Verlag

This paper is a contribution to the GI Workshop: