HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **37**; April 1996
ISSN 0783–5396
ISBN 951-22-3066-6

AN INDUCTION THEOREM FOR RING PROTOCOLS OF PROCESSES DESCRIBED WITH PREDICATE/TRANSITION NETS

TINO PYSSYSALO

Digital Systems Laboratory
Department of Computer Science
Helsinki University of Technology
Otaniemi, Finland

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **37**; April 1996

ISSN 0783–5396
ISBN 951-22-3066-6

An Induction Theorem for Ring Protocols of Processes Described with Predicate/Transition Nets

TINO PYSSYSALO

Abstract: The verification of concurrent systems, consisting of many identical finite-state machines, has gained a lot of research interest recently. One reason is the growth of data communication networks, where millions of computers use the same finite-state protocol to communicate with each other.

In this report we present an induction theorem for ring protocols, consisting of an arbitrary number of identical processes, described using Pr/T-nets. Applying the proposed method temporal properties of protocols, using addresses in communication, can be verified. This has not been possible with induction methods published earlier, because the behaviour of the processes, using addresses in rings of different sizes, has not been considered identical. Using symmetries of the reachability graph of the Pr/T-net of a verified system, we abstract the addresses used by the processes away from the observable behaviour of the system, before the identity of the processes is examined.

As an example we show that a starvation-free solution to the dining philosophers' problem is correct with any number of philosophers. In addition, we show that the arbitration algorithm of the Frame Synchronized Ring is fair, whatever the number of nodes is in the ring, using the assumption that every node is continuously ready to send. The arbitration algorithm of the FSR cannot be successfully verified using methods that have been published earlier.

Keywords: Concurrent and distributed systems, Frame Synchronized Ring, induction theorems, protocol verification, predicate/transition nets.

Printing: TKK Monistamo; Otaniemi 1996

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

Phone: $\frac{90}{+358\text{–}0}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–0–465 077
E-mail: lab@saturn.hut.fi

Contents

1 Introduction

The future trend of data communications is the interconnection of all computers into one global communication network. This huge information super highway provides real-time multimedia services, like video and audio transfer, videoconferencing, homeshopping and hypermedia library. The access to these services takes several different forms. To enable a feasible information exchange between a set of computers, providing and utilizing these services, a set of rules is required to govern the interactions between communicating entities. This set of rules is called a *communication protocol*. Two computers can communicate with each other only if they use the same communication protocol. The number of computers using the same protocol in the global communication network is enormous. For example in Internet there are millions of computers using *Transmission Control Protocol / Internet Protocol* (*TCP/IP*).

Correct design of communication protocols is crucial for a reliable and efficient network operation. For example correctly implemented protocol must not contain any deadlocks and should guarantee a fair access to the resources for all users of the protocol. It is very difficult to verify the correctness of a protocol used in a concurrent communication network without proper analysis methods and automatic verification tools. To ensure the correct behaviour of the protocol, it must be formally specified and then formally verified. Most widely used formal specification and verification methods can be categorized into two classes: those, which are based on state transitions and those, which are based on actions, being observable for an external observer. One of the most straightforward techniques to verify state transition based protocols is the *reachability analysis*, while protocols, specified with observable actions, are usually verified using different kinds of *bisimulations*. One of the most widely used formalisms based on state transitions is *Petri nets* (Reisig 1985). Formal specification languages, based on *process algebras* (Brookes, Hoare and Roscoe 1984, Milner 1989), belong to the latter class.

1.1 The Problem of Verifying Identical Processes

A concurrent system can exhibit extremely complicated behaviour, because of the complex timing of actions of different processes, even if all the processes were identical. The main research problem of this work is the formal analysis of concurrent systems or more specifically *ring protocols*, consisting of a large number of identical finite-state machines or *processes*, i.e., instances of the same (finite) protocol. Given a system $S(n)$, consisting of $n \geq 2$ instances of the same protocol and a property $P(n)$ on $S(n)$, we would like to verify, whether or not $S(n)$ satisfies $P(n)$ for any value of n. The problem is depicted in Figure 1. In (Clarke, Grümberg and Browne 1986b) it is proven that this problem is not semi-decidable in general, even if $P(n)$ is a very simple temporal property. In addition, the problem remains undecidable, even if $S(n)$ is a unidirectional ring of identical finite-state machines, whose configuration is independent of the value of n and only one state transition can occur at a time in $S(n)$. The formal proof is given in (Suzuki 1988).

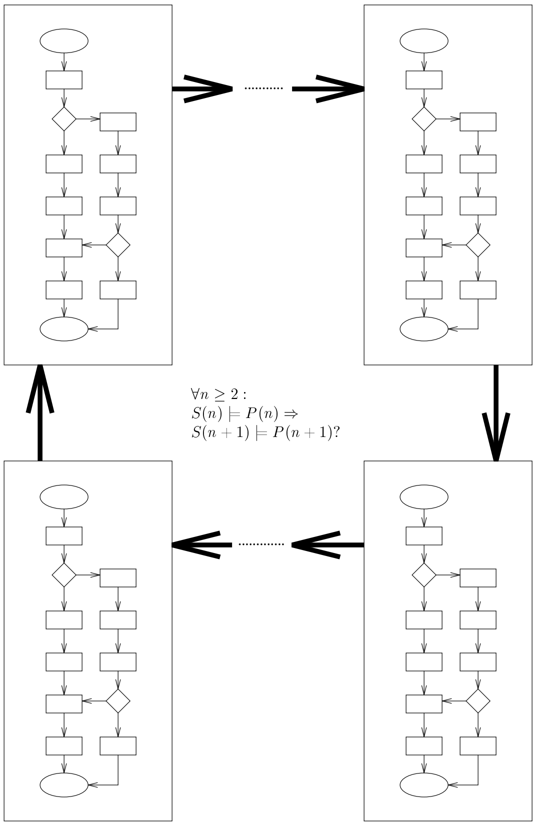


Figure 1: The problem of verifying n identical finite-state machines.

It is clear that conventional methods, based on state transitions, cannot be applied directly to the verification of an arbitrary number of identical processes, because state transition based methods can be applied only to instances having a fixed state space. In addition, the state space of the concurrent system often grows exponentially with respect to some system parameter, e.g., the number of concurrent processes. This phenomenon is called the *combinatory* or the *state space explosion problem.* The state space can be reduced with several reduction methods, based on for example the *stubborn set method* (Valmari 1988) or the *symmetry method* (Jensen 1987). Unfortunately, the reduced state space seldom has the same *temporal properties* as the complete state space.

Because of the state space explosion problem of state transition based methods (and the popularity of process algebras) the first *induction theorems* for identical processes were applicable to the systems, consisting of processes modelled using process algebras. In (Kurshan and McMillian 1989) a structural induction theorem is given for processes, which are specified using *Calculus of Communicating Systems* (*CCS*) (Milner 1989). By applying the theorem it can be shown that a concurrent system, consisting of two identical processes, has similar properties as a system, consisting of any number of identical processes. However, a special invariant process must be provided to carry out the induction. Sometimes it is very difficult to find this invariant process, even if it exists. That is why this method cannot be fully automated and is unfeasible to use in practice.

In (German and Sistla 1992) a fully automatic method is given for verifying temporal properties of concurrent systems, containing an arbitrary number of CCS processes. The idea of this method is that the set of executions, i.e., observable actions, of a single process satisfies the same specification as the set of executions of an arbitrary number of processes satisfies. However, because only a single process is verified, this method has a serious weakness. Global properties such as the mutual exclusion or the absence of deadlock cannot be verified in a straightforward way. In addition, some global properties cannot be verified at all.

Because of the weaknesses of the induction theorems, applicable to the processes, described using process algebras, it would be more reasonable to define an induction theorem for processes modelled using state transition methods, e.g., Petri nets, although it is likely that the state space explosion problem restricts the applicability of the theorem. In (Li, Suzuki and Yamashita 1994a) such an induction theorem is given for processes modelled using fair Petri nets (Suzuki 1985). No invariant processes are needed to carry out the induction, because it is assumed that the invariant process is always a composition of some finite number of identical system processes. The verification of global temporal properties is possible and in fact quite easy too. However, in cases, where the invariant process is not a composition of any finite number of system processes, but the invariant process still exists, this induction theorem is not applicable. Therefore the theorem is not as general as the theorems defined for CCS processes.

The idea of the induction theorem for processes of fair Petri nets is very similar to the idea of theorems applicable to processes, modelled using CCS. The set of transitions of a Petri net is partitioned into internal and interface or observable transitions in a similar way as the actions of the CCS processes are either hidden or observable. If the observable behaviour, i.e., the firing sequence of observable transitions, of the ring consisting of two identical processes is similar to the observable behaviour of a ring, consisting of three processes and some additional conditions are satisfied, we can conclude that the observable behaviour of the ring, consisting of any number of identical processes, is similar to the behaviour of the ring of two processes. Thus, any temporal property, which the ring of two processes satisfies, is satisfied by the ring, containing an arbitrary number of processes too. In (Li, Suzuki and Yamashita 1994b) this induction theorem is extended so that the method can be applied, if any two rings of size $n \geq 5$ and $n-1$ have the similar observable behaviour.

1.2 An Induction Theorem for Verifying Pr/T-Net Processes

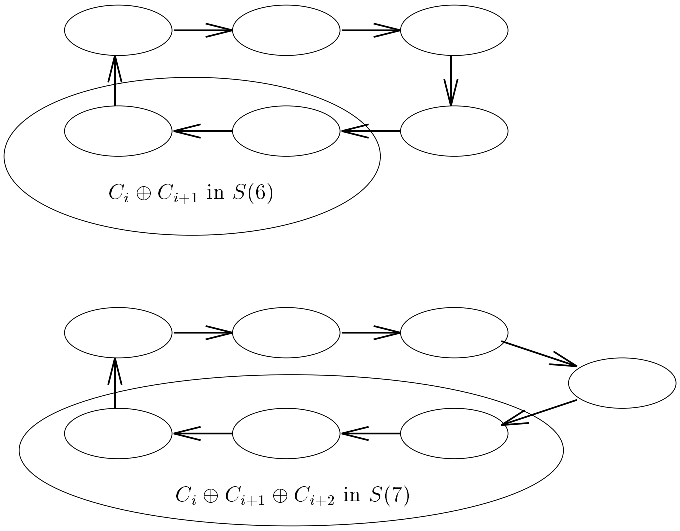
The goal of this report is to develop an induction theorem, which does not suffer from the restrictions of previously published theorems, but which takes the advantage of the useful properties of these theorems. Our induction theorem is greatly inspired by the theorem, given in (Li et al. 1994b). We however model the processes of the ring using *predicate/transition nets* (*Pr/T-nets*) (Genrich 1987), because of their modelling flexibility. Informally a Pr/T-net is an annotated net, which is generated by folding an ordinary low-level Petri net. The Pr/T-net formalism is a considerably more compact way of modelling concurrent systems, containing identical processes, than low-level Petri nets. By using formal transformations (Genrich 1989), i.e., re-writing rules, for Pr/T-nets we can model any system that can be modelled with Pr/T-nets with a net, containing only one place and one transition. An addition of a process does not usually affect the size of the Pr/T-net model of the system. Only the annotation of the net is changed.

The induction theorem, presented in this report, allows us to formally verify several such systems and protocols, the analysis of which has been impossible with earlier methods. This is due to the fact that none of the previously published induction theorems can directly handle rings, in which nodes use source and destination addresses. An addition of a process into this kind of a ring changes the structure of other processes in the ring making rings of different size dissimilar. The structure changes, because processes must have new actions or transitions in order to be able to communicate with the new process. For example in low-level Petri nets a new transition is needed, because all tokens are identical. Tokens of Pr/T-nets are tuples of individuals. One field in the tuple can carry the address information and the structure of the underlying Petri net of a Pr/T-net remains unchanged, when a new process is added into a ring. Only the tokens of the net, i.e., the initial marking of the net changes, when the number of processes changes.

In almost all real ring protocols, e.g., *Fiber Distributed Digital Interface* (*FDDI*) or IBM's *Token ring*, communicating processes use addresses to send data to each other. Our induction theorem allows the usage of addresses in a natural and easy way. We map the real computation of processes into an observable behaviour so that addresses and other identities used in the communication do not affect this observable behaviour of the processes. Therefore, the processes, having a similar observable behaviour, can be considered identical. This mapping must be done anyhow to get rid of the identities, which the processes get in the folding of a low-level Petri net.

When applying our induction theorem a set of identical processes is modelled using the Pr/T-net formalism. The Pr/T-net model of the system contains a set of Pr/T-net components, each describing one identical system process. The observable behaviour of a Pr/T-net component consists of the firings of interface transitions, i.e., transitions that move tokens between net components, and the markings of interface places, which are input places of the interface transitions. The observable behaviour of a net is a subset of the real behaviour or the complete reachability graph of the Pr/T-net. This means that we can hide some irrelevant state transitions in the same way as we hide some actions in process algebras. We prove that in order to show that two Pr/T-net components have similar behaviour, it is sufficient to show that the observable behaviour of the components is similar, if some additional conditions are simultaneously satisfied.

The key idea of our theorem is depicted in Figure 2. The idea is very much inspired by the induction theorem, given in (Li et al. 1994b). Rather than showing that two processes described using the net components have a similar behaviour, which implies that the processes are identical, we show that the two sets of processes or components of size $k \geq 2$ and $k+1$ have a similar behaviour. In many ring protocols two processes may have a different behaviour from the behaviour of three processes, although the processes of the protocol always behave similarly, when the number of processes is larger than three. Consider a well-known dining philosophers' problem. If in the problem of two philosophers, both adjacent philosophers pick up a chopstick, the system deadlocks, because all chopsticks have been reserved. In a system consisting of at least three philosophers, two adjacent philosophers can pick up a chopstick and still the system does not deadlock, because there is still one chopstick available. The verification method, based on the idea of the composition of processes until a fix point is reached, is given in (Valmari and Tienari 1991). The idea of our induction theorem is similar to the composition of processes. Our theorem is applicable, if k components in the ring of n components, where $n \geq 4$ and $2 \leq k \leq n/2$, have similar observable behaviour (we shall later use the term externally visible history) as $k+1$ components in the ring of $n+1$ components. This is depicted in Figure 2. In this case we can change the number of components of the ring without affecting the behaviour of the ring by replacing the set of components of size $k+1$ with the set of size k and vice versa.



If the observable behaviour of $C_i \oplus C_{i+1}$ of $S(6)$ is similar to the observable behaviour of $C_i \oplus C_{i+1} \oplus C_{i+2}$ of $S(7)$, then

$S(6) \models P(6) \Rightarrow$

$S(n) \models P(n)$ for any $n \geq 6$.

Figure 2: An example of the application of the induction theorem.

It is possible to prove any branching time temporal logic property of the ring protocol of a finite (usually small) number of processes automatically (Clarke, Emerson and Sistla 1986a) using a verification tool and then show by using our induction theorem that the protocol of an arbitrary number of processes satisfies (or does not satisfy) the same properties. The automatic verification tool we have used is the predicate/transition net reachability analysis tool, `PROD` (Varpaaniemi, Halme, Hiekkanen and Pyssysalo 1995), developed in the Digital Systems Laboratory of Helsinki University of Technology. With `PROD` we can reason about infinite computation paths of finite automata with three acceptance conditions: finite, looping and repeating (Wolper, Vardi and Sistla 1983).

No invariant process is needed to carry out the induction with our method. Our theorem rather gives a sufficient condition that such an invariant process exists. However, the modelling flexibility of Pr/T-nets allows us to construct the model of the system under verification in such a way that the invariant process can be found easily by the composition of processes, if such an invariant process exists. Thus, our method suffers from the same restriction as the induction theorem proposed in (Li et al. 1994b). In the same paper the induction theorem is applicable, if the behaviour of the processes of a system is similar, when only fair transition sequences are examined. The transitions of a Pr/T-net are not categorized into fair and unfair transitions. Therefore, our induction theorem is applicable only, if the behaviour of the processes is similar, when all possible transition sequences of the net are examined. This corresponds with the case, in which the firing of the transitions of a fair Petri net is not restricted by a temporal formula, i.e., the set of fair transitions is empty.

The most serious restriction of our theorem is the state space explosion. When verifying some real concurrent systems, it may be necessary to examine rings, containing at least ten identical instances of the same protocol. To be able to verify the temporal properties of these systems, the complete global state reachability graph must be generated, which can easily contain more than 500 million states. Efficient analysis of systems of this size is impossible with the most commonly available automatic verification tools. However, we can try to reduce the size of the reachability graph of a system using Pr/T-net transformations, preserving the properties of the original net model, so that the net model contains only one place and one transition. Unfortunately, this kind of a net usually has complex arc expressions and firing conditions, which makes the application of the induction theorem difficult. A feasible application of the theorem demands good modelling skills to find a balance between the size of the reachability graph of the net model of the system and the complexity of the application of the induction theorem.

1.3 Outline of the Report

In Chapter 2 we present the evolution of induction theorems for verifying ring protocols of an arbitrary number of processes. In Chapter 3 we give the preliminaries, needed to understand our induction theorem, presented in Chapter 4. We also prove the correctness of a starvation-free solution to the dining philosophers' problem for an arbitrary number of philosophers in Chapter 3. In Chapter 5 we show that a slightly simplified arbitration algorithm of the *Frame Synchronized Ring* guarantees a fair access to the data bus in a ring of any size. Chapter 6 contains the conclusions and some discussion about the future work.

2 On the Analysis of Identical Processes

In this chapter we shall describe the evolution of the verification methods used in the analysis of an arbitrary number of identical processes and represent the most relevant research results of recently published induction theorems. In the earliest papers the processes were specified and verified using temporal logic and process algebras, while in the most recent ones the properties of temporal logic and process algebras and the modelling flexibility of Petri nets are combined so that the verification of systems consisting of many identical processes could be carried out as automatically and easily as possible.

2.1 Processes Described Using Process Algebras

First induction theorems for verifying many identical finite-state machines were defined for systems, the processes of which were described using process algebras. Two most commonly used process algebras were *Communicating Sequential Processes* (*CSP*) (Brookes et al. 1984) and Milner's *Calculus of Communicating Systems* (Milner 1989). One of the main goals of these papers was the development of an induction theorem, with which the verification could be carried out fully automatically. However, the problem of checking, whether the system $S(n)$ satisfies a property $P(n)$ with any value of n, when a system $S(n)$ satisfies a property $P(n)$ with some n, has been proven to be undecidable in general (Apt and Kozen 1986, Suzuki 1988).

In (Kurshan and McMillian 1989) a structural induction theorem is presented which can be applied for example to processes of a network with an unspecified number of hosts, to processes of a multiprocessor with an unspecified number of CPUs or to a queue with an unspecified number of buffers. The processes are specified using CCS. The induction theorem makes use of a partial ordering of processes. An invariant process must be provided to carry out the induction and therefore the verification is not fully automatic. In fact there are no rules, how the invariant process could be computed in general. In many cases the invariant process could be obtained by the composition of some finite number of system processes, but this does not necessarily work. In Kurshan's and McMillian's opinion this is not a restriction, because the induction always involves a "good guess". However, given the invariant process the checking procedure can be carried out automatically using an algorithm appropriate for the partial order of CCS processes. In the partial order a process P_1 is less than or equal to a process P_2 if and only if P_1 implements P_2 or equivalently P_2 is an approximation of P_1, i.e., is less deterministic than P_2. Two processes are considered identical, if they cannot be distinguished by any finite sequence of observable transitions (Clarke et al. 1986b).

Kurshan and McMillian do the induction verification in three steps. In the first step it is checked that the invariant process is greater in the partial order than the composition of some finite number n of processes. Next in the induction step it is checked, whether the composition of the invariant with the $(n+1)$th system process is less than or equal to the invariant process. The last step is the satisfaction test. The invariant process must satisfy the specification of the concurrent system. In (Wolper and Lovinfosse 1989) a similar structural induction theorem is given, but the processes can be specified using either CCS or CSP.

The same principle, as proposed by Kurshan, McMillian, Wolper and Lovinfosse, to carry out the induction is used in many recently published papers. The main difference is that in these papers it is assumed that the invariant process is always a composition of some small number of system processes. The number of processes needed to obtain the invariant or the fixpoint depends on the problem. For example in the dining philosophers' problem the composition of three processes, describing the philosophers, forms a fixpoint, i.e., the behaviour of the system does not change, when the number of philosophers is greater than two. This is shown in (Valmari and Tienari 1991). It is not known, whether the invariant can always be obtained by the composition, if it exists. In some cases the search for the invariant can be made easier by changing the formal model of the system. This change should not affect the properties of the model. For Petri nets a lot of transformation rules preserving the properties of the original model have been defined. That is one reason, why it might be preferable to use Petri nets in the verification of many identical processes, when the existence of the invariant process is not known.

A method for *automatically* verifying temporal properties of concurrent systems containing an arbitrary number of finite-state processes that communicate using CCS actions is given by the same authors first in (Sistla and German 1987) and later in (German and Sistla 1992). The method can be applied to the systems, consisting of one control process and n identical system processes or to the systems, consisting of n identical system processes only. The temporal properties of the processes are specified using *Propositional Linear Temporal Logic* (*PTL*). All executions, i.e., CCS actions, of a process satisfy a given specification, if the set of execution of a single process is exactly the set of strings accepted by a certain finite-state automaton on infinite strings. PTL specifies the properties of the executions of a single process of the system only, but many important properties such as the mutual exclusion and the absence of deadlock are global properties. In general the verification of global properties is impossible, but for example the absence of deadlocks can be verified by examining, whether there exists a number of processes n such that there is a computation in the system of size n that ends in a deadlock.

In (Valmari and Tienari 1991) an algorithm for reducing the observable behaviour of the process model of a system without affecting its semantics is given. The behaviour of the system is described using the *Labelled Transitions System* (*LTS*), which is a directed graph, whose vertices correspond to the states of the system and edges correspond to state transitions. Any edge is labelled by the name of the action occurring in connection with the state transition. A described system is then reduced such that the reduced LTS is equivalent with the corresponding full ones. For example the reduced LTS, describing the dining philosophers' problem of at least four philosophers, is isomorphic to the reduced LTS, describing three philosophers. Thus, a global state analysis of the behaviour of the system with an arbitrary number of philosophers is possible. The reduction algorithm is implemented into the ARA-tool (Kemppainen 1994). The weakness of this method is that indexed processes, i.e., processes having an identity or an address cannot be verified directly.

With all of these methods only systems consisting of an arbitrary number of finite-state machines can be analyzed. Three additional methods for testing the equivalence of finite CCS processes is given in (de Nicola and Hennessy 1983). In (Darondeau and Gamatie 1990) an observational model for the infinite behaviours of CCS processes is given. With this method the equivalence of processes, having infinite behaviour, can be examined. However, processes must still be finite.

2.2 Induction Theorems for Processes Specified Using Temporal Logic

One possibility to use temporal logic in the verification of many identical processes is given in (Clarke et al. 1986b). The processes of the concurrent system are described using labelled state transition graph or a Kripke structure and *Indexed Computation Tree Logic* (*Indexed CTL**) (Clarke et al. 1986a). Bisimulation is used to study the equivalence between two structures with the same set of indexed propositions but different sets of index values. A closed formula of Indexed CTL* is true in the initial state of one Kripke structure if and only if it is true in the initial state of the other. In (Manna and Pnueli 1991) a method for verifying an arbitrary number of indexed processes specified using temporal logic is given. Processes use different index values, but are otherwise similar. The verification of the described model can be carried out automatically, but the modelling of concurrency is not as easy as with Petri nets.

2.3 Processes Described Using Petri Nets

The induction methods, which are published so far and which are used in the verification of identical processes, described using Petri nets, are mainly based on the structural properties of the system. In (Cai, Sekiguchi, Tanaka, Hikichi and Maruyama 1993) transition invariants are presented for the analysis of structural properties of a Petri net. In (Murata 1989) several place and transition invariants are defined, with which many properties of the net, including for example liveness and structural boundedness, can be proven. A new set of invariants for the analysis of a Petri net is defined in (Couvrer and Paviot-Adet 1994). There are however weaknesses and restrictions in the applicability of these invariant methods. The computation of the invariants becomes complex, when the size and the complexity of a Petri net increases and there are several interesting temporal properties, which are impossible to show using the net invariants.

In the more recent papers (Li et al. 1994a, Li et al. 1994b) useful properties and ideas of process algebras and the temporal logic are combined together with the modelling flexibility of Petri nets to make the induction verification easier. In addition, there has been a lot of research in order to combine the modularity and the structure of process algebras and the faithful description of concurrency of Petri nets (Goltz and Mycroft 1984, Nielsen 1987). Some newer results concerning this combination are presented in (Montanari and Yankelevich 1994, Schreiber 1995). The idea of the induction theorems for processes described using Petri nets is to verify a desired temporal property of a system, consisting of some specific number of processes. This part of the verification can be done automatically with any verification tool. Then the conditions for the induction theorem are checked. If the net model satisfies the conditions, the induction theorem can be applied and it can be concluded that the temporal property, satisfied by the system of some specific size, is satisfied by the system of an arbitrary size.

The conditions for the induction state that if a ring of n ($n \geq 5$) processes is k-mergeable and a ring of $n-1$ processes is $(k-1)$-splittable with some $2 \leq k \leq n/2$ and both rings are uniform and structurally bounded, then any property that is satisfied by the ring of size $n-1$ is satisfied by the ring of any size greater than $n-1$. The *k-mergeability* means that in the ring of n processes k processes can be merged into the ring of $k-1$ processes such that the ring of size k and $k-1$ behaves similarly. The *$(k-1)$-splittability* means that the ring of $k-1$ processes can be split into the ring of k processes such that the behaviour of the ring of size $k-1$ is similar to the behaviour of the ring of size k. The *uniformity* of a ring means that all the processes of the ring are identical. Like all other methods, the induction theorems given in (Li et al. 1994a, Li et al. 1994b) as well as the theorem, proposed in this report, are applicable to finite-state machines only. That is why the rings of size n

and $n-1$ must be *structurally bounded.* In fact, a ring of any finite number of processes is structurally bounded, if the ring of two processes is structurally bounded. The proof is given in (Li et al. 1994a). If a ring of n processes is k-mergeable, uniform and structurally bounded, it means that k processes can be replaced with $k-1$ processes, which implies that the behaviour of the ring of n processes is a subset of the behaviour of the ring of $n-1$ processes. If the ring of $n-1$ processes is $(k-1)$-splittable, uniform and structurally bounded, it means that $k-1$ processes of the ring of size $n-1$ can be replaced with k processes, which implies that the behaviour of the ring of $n-1$ processes is a subset of the behaviour of the ring of n processes. Thus, the splittability and mergeability together implies that the rings of n and $n-1$ processes have a similar behaviour.

Our method is in some sense a combination of all the previously discussed methods. We specify and verify temporal properties of the modelled system using branching time temporal logic and the automatic Pr/T-net reachability analysis tool, `PROD`. Processes of a Pr/T-net model are identified in a similar way as the indexed processes described using temporal logic. Process algebras deal with actions, which correspond to the reachability graph of a Pr/T-net that is given implicitly in the structure and initial marking of a net. We use an invariant process, which is always a combination of some system processes. In (Li et al. 1994a, Li et al. 1994b) this is considered as a restriction, because the induction theorem is not applicable, if the invariant process cannot be obtained by combining system processes. In addition to the properties of the previous methods, using our induction theorem protocols, using addresses and identities in the communication, can be verified too. This is a very useful extension, because it makes the analysis of many real communication protocols possible. The modelling flexibility of Pr/T-nets helps us to model the system so that the reachability graph of the system is small enough so that it can be efficiently verified using automatic verification tools, although in many cases the reachability graphs are still too large. On the other hand the modelling flexibility makes the transformation of the net model easy so that the interactions between different net components are minimal and thus the induction theorem is easily applicable and the invariant process can be easily found.

3 A Structural Induction Theorem for Rings of Fair Petri Nets

The work of this report has been inspired by the structural induction theorem for rings of processes of fair Petri nets, given in (Li et al. 1994a). Our theorem, applicable to ring protocols of identical processes, described using predicate/transition nets, is an extension of this induction theorem, and that is why we shall present the theorem in this chapter.

The structural induction theorem, given in (Li et al. 1994a), can be used to formally infer the correctness of a ring of any size from the correctness of the ring having only a few components, described using fair Petri nets. The induction is based on the similarity of the rings. For $k \geq 2$ let R^k be a ring consisting of k components. The induction theorem states that if R^k is structurally bounded, R^2 and R^3 are similar and if certain additional conditions are satisfied, then for any $k \geq 4$, R^2 and R^k are also similar.

It is possible and in fact quite usual that R^2 and R^j are not similar for some small $j > 2$, even though R^j and R^k are similar for all $k > j$. Recall the dining philosophers' problem, where the problem instance of two philosophers deadlocks, if both philosophers pick up one chopstick, while the instance of three or more philosophers does not deadlock, when two adjacent philosophers have picked up one chopstick each. In (Li et al. 1994b) a new structural induction theorem is given to overcome this restriction. The theorem is applicable, when the rings of sizes $k-1$ and k ($k \geq 5$) have a similar behaviour. We will present this induction theorem here and show that a starvation-free solution to the dining philosophers' problem of five philosophers, given in (Wu and Murata 1983), is correct whatever the number of philosophers is. First we give the definition of fair Petri nets and the similarity between the rings, consisting of components of fair Petri nets.

3.1 Temporal and Fair Petri Nets

A Petri net is a directed graph with two types of nodes, called places and transitions. A node of one type (a place or a transition) is connected to a node of the other type (a transition or a place) by weighted arcs. In (Reisig 1985) an excellent introduction to Petri nets is given. Formally we define a Petri net as a triple of a set of places, a set of transitions and a weight function between places and transitions and between transitions and places.

Definition 3.1 *A* Petri net *is a triple* $N = (P, T; F)$ *where*

> P *is a finite set of* places*,*

T is a finite set of transitions ($P \cap T = \emptyset$) *and*

$F : (P \times T) \cup (T \times P) \to \mathbb{N}$ *is a* weight function. □

A *marking* of a net is a function that assigns to each place a natural number $M : P \to \mathbb{N}$. At a marking M a place p has $M(p)$ tokens. An *initial marking* assigns to each place an initial token count. A place p is an *input place* (or *output place*) of a transition t, if $F(p,t) \geq 1$ (or $F(t,p) \geq 1$).

Definition 3.2 *A transition $t \in T$ is* enabled *at a marking M if and only if $\forall p \in P : M(p) \geq F(p,t)$.* □

An enabled transition t at a marking M can fire and yield to another marking M' such that $\forall p \in P : M'(p) = M(p) - F(p,t) + F(t,p)$. We denote this by $M \to_t M'$. Let T^* be a set of all finite sequences of transitions of T, including the empty set λ and T^ω be a set of all infinite sequences of transitions of T. If $\alpha \in T^*$ and $\beta \in S^* \cup S^\omega$, let $\alpha\beta$ be the concatenation of α and β. $\alpha^i, i \geq 0$ denotes a concatenation of i copies and α^ω denotes the concatenation of infinite copies of a finite *transition sequence* α. We denote the length of a finite transition sequence α by $|\alpha|$ and the length of an infinite transition sequence β by ω. A marking M' is reachable from a marking M, if there is a finite transition sequence α such that $M \to_{t_1} M_1 \to_{t_2} M_2 \cdots \to_{t_{|\alpha|}} M', t_1 \cdots t_{|\alpha|} \in \alpha$. Let $L(N,M)$ denote all finite firing sequences from a marking M, $L^\omega(N,M)$ denote all infinite firing sequences from M and $L^\infty(N,M) = L(N,M) \cup L^\omega(N,M)$. A Petri net is *structurally bounded*, if none of its places contains more than k tokens of some finite k at any marking reachable from an arbitrary finite initial marking. In (Murata 1989) it is shown, how structurally boundedness of the net can be proven using net invariants.

Graphically we present Petri nets by drawing places of a net as circles, transitions as squares and the weight function as an arc with weight $F(p,t)$ (or $F(t,p)$) from p to t (or t to p), if $F(p,t) > 1$ (or $F(t,p) > 1$), respectively. The weight is omitted from the arcs, the weight of which are 1. A marking M is drawn by drawing $M(p)$ dots in the place p.

A *temporal Petri net* is a pair (N, f), where N is a Petri net and f is a temporal logic formula. Definitions and fundamental properties of temporal Petri nets are given in (Suzuki 1985). In (Manna and Pnueli 1991) a good introduction to temporal logic and temporal logic formulas is given. The formula is used to restrict the possible firing sequences of a net N. Let $\mathcal{L}(N,M,f)$ denote the set of firing sequences $\alpha \in L^\infty(N,M)$ from a marking M, such that

1) α is either infinite, or finite and terminating and

2) α satisfies f, i.e, f is true in the initial node of the reachability graph.

A *fair Petri net* is a temporal Petri net, the transitions of which are partitioned into a fair transition set $T' \subseteq T$ and into an unfair transition set $T \backslash T'$. In addition, it is required that the transitions, belonging to the fair set satisfy formulas of the form:

$$f(T') = \bigwedge_{t \in T'} ((\Box\Diamond \uparrow t) \Rightarrow (\Box\Diamond t)) \tag{1}$$

In equation 3.1 we use standard temporal operators $\Box$ (eventually) and $\Diamond$ (always). So $\Box\Diamond$ means always eventually, i.e., infinitely often. We denote en enabled transition t by $\uparrow t$. So a fair Petri net is a temporal Petri net, in which every transition, belonging to the fair transition set and that is enabled infinitely often, is fired infinitely often.

3.2 Similarity of Components of Fair Petri Nets

Next we define a component of a Petri net and the similarity between the components of fair Petri nets. As an example component, we use a Petri net given in Figure 3 (Wu and Murata 1983), modelling one philosopher in the dining philosophers' problem. Places `pt`, `pw` and `pe` describe the thinking, waiting and eating phases of the philosopher, respectively. Tokens, indicating chopsticks or an allowance card are kept in places `pc` and `pa`, respectively. Place `pd` is marked, when the philosopher is not hungry and wants to pass the allowance card to the next philosopher. We will present the problem and its solution in greater detail in the next section.

Definition 3.3 (Li et al. 1) *A* component *is a Petri net $C = (P, T; F)$ in which the set T of transitions can be partitioned as $T = T_L \cup T_I \cup T_R$ such that $|T_L| = |T_R| \geq 1$. The transitions in T_L, T_I and T_R are called* left interface transitions*,* internal transitions *and* right interface transitions*, respectively.* $\Box$

Example 3.4 One possible partition of the transitions of the Petri net in Figure 3 is $T_{\mathrm{L}} = \{\mathrm{te_i}, \mathrm{tt_{i-1}}, \mathrm{tc_{i-1}}\}$, $T_{\mathrm{I}} = \{\mathrm{td_i}, \mathrm{tw_i}\}$ and $T_{\mathrm{R}} = \{\mathrm{te_{i-1}}, \mathrm{tt_i}, \mathrm{tc_i}\}$. $\Box$

We form Petri net chains and rings by merging two or more Petri net components.

Definition 3.5 (Li et al. 2) *Let $C = (P, T; F)$ be a component having places $P = \{p_1, \cdots, p_n\}$, left interface transitions $T_L = \{u_1, \cdots, u_m\}$, internal transitions $T_I = \{v_1, \cdots, v_s\}$ and right interface transitions $T_R = \{w_1, \cdots, w_m\}$, where $T = T_L \cup T_I \cup T_R$. For each $i \geq 0, C_i = (P_i, T_i; F_i)$ denotes the Petri net having the same structure as C, in which:*

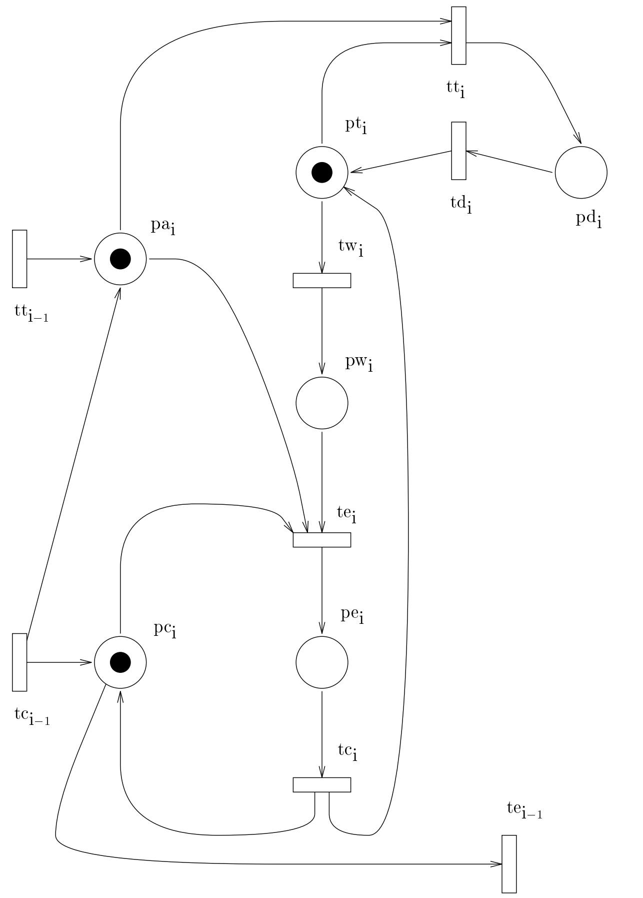


Figure 3: The Petri net representation of the ith philosopher.

1) each p_j is renamed $p_{i,j}, P_i = \{p_{i,1}, \cdots, p_{i,n}\}$,

2) each u_j is renamed $t_{i-1,j}, T_{i,L} = \{t_{i-1,1}, \cdots, t_{i-1,m}\}$,

3) each v_j is renamed $v_{i,j}, T_{i,I} = \{v_{i,1}, \cdots, v_{i,s}\}$,

4) each w_j is renamed $t_{i,j}, T_{i,R} = \{t_{i,1}, \cdots, t_{i,m}\}$ and

5) $T_i = T_{i,L} \cup T_{i,I} \cup T_{i,R}$.

F_i is identical to F under the renaming given above. For $0 \leq i \leq j$,

$$C_i \oplus C_{i+1} \oplus \cdots \oplus \ C_j = (\bigcup_{i \leq l \leq j} P_l, \bigcup_{i \leq l \leq j} T_l, \bigcup_{i \leq l \leq j} F_l)$$

denotes the chain *consisting of the processes or the components $C_i, C_{i+1}, \cdots, C_j$. For each $k \geq 2$:*

$$R^k = C_0 \odot C_1 \oplus \cdots \oplus C_{k-1} = (\bigcup_{0\ \leq l \leq k-1} P_l, \bigcup_{0 \leq l \leq k-1} T_l, \bigcup_{0\ \leq l \leq k-1} F_l)$$

where all subscripts are taken modulo k, denotes the ring *consisting of the components* $C_0, C_1, \cdots, C_{k-1}$. □

The symbol "$\odot$" in the ring $C_0 \odot C_1 \oplus \cdots \oplus C_{k-1}$ is an operator that closes the chain $C_1 \oplus \cdots \oplus C_{k-1}$ into a ring using C_0. A marking of R^k is denoted by M^k and is described as a tuple $(M_0, M_1, \cdots, M_{k-1})$, where each marking M_i is the marking of the corresponding component C_i. It is assumed in (Li et al. 1994a) that all the components of a ring except possibly C_0 have the same initial marking.

Example 3.6 In Figure 4 a ring, consisting of two components describing the philosophers in the dining philosophers' problem, is given. The initial marking of the net is $M^2 = (M_0, M_1)$, where for $1 \leq j \leq 6$ $M_0(p_j) = <1, 0, 0, 1, 1, 0>$ and $M_1(p_j) = <1, 0, 0, 1, 0, 0>$. So initially both chopsticks are available ($p_{0,4}$ and $p_{1,4}$ are marked), both philosophers are thinking ($p_{0,1}$ and $p_{1,1}$ contain a token) and philosopher 0 has the allowance card ($p_{0,5}$ contains a token). □

In R^k the set of fair transitions T^k must satisfy the fairness condition 3.1 and must be generated in such a way that if a transition of a component belongs to T^k then all corresponding transitions of other components must belong to T^k too and if a transition of a component does not belong to T^k then none of the corresponding transitions of other components must belong to T^k either. For a transition t a transition sequence α is *t-legal*, if α is infinite and satisfies 3.1 or is finite and terminating. α is *legal at* C_i, if it is t-legal for all transitions $t \in T^k$ that belong to C_i. An input place of a left (or right)

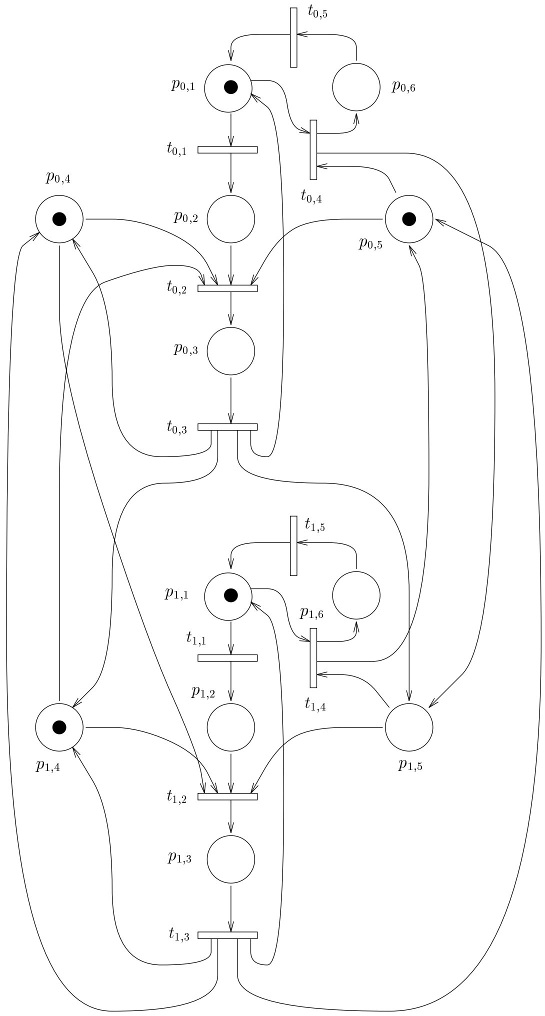


Figure 4: The ring consisting of two philosophers.

interface transition of the component is called a *left* (or *right*) *interface place* of the component. Next we give the definition of a *firability vector.* With the firability vector we define the *history* of a component in a transition sequence. We then define the *similarity* of the components by using the history of the components.

Definition 3.7 (Li et al. 3) *Let* $p_{i_1}, \cdots, p_{i_L}$ *and* $p_{j_1}, \cdots, p_{j_R}$ *be the left and right interface places of* C*, respectively, where* $1 \leq i_1 < \cdots < i_L \leq n$ *and* $1 \leq j_1 < \cdots < j_R \leq n$*. Then for a chain* $C_a \oplus C_{a+1} \oplus \cdots \oplus C_b$ *of length at most* $k-1$ *within* R^k *and a marking* $(M_0, M_1, \cdots, M_{k-1})$*, the* firability vector of $C_a \oplus C_{a+1} \oplus \cdots \oplus C_b$ *at* $(M_0, M_1, \cdots, M_{k-1})$ *is the column vector*

$$\begin{bmatrix} M_{a-1}(p_{j_1}) \\ \vdots \\ M_{a-1}(p_{j_R}) \\ M_a(p_{i_1}) \\ \vdots \\ M_a(p_{i_L}) \\ M_b(p_{j_1}) \\ \vdots \\ M_b(p_{j_R}) \\ M_{b+1}(p_{i_1}) \\ \vdots \\ M_{b+1}(p_{i_L}) \end{bmatrix}$$

□

3.2.1 Strong Similarity

Let $L^\infty(k)$ be an abbreviation of $L^\infty(N^k, M^k)$. We define three kinds of histories for the components of fair Petri nets. The extended local history of the component is the portion of the transition sequence that occurs inside the component and affects the marking of the component. The local history of the component is similar to the extended local history except that it ignores the firability vectors. The externally visible history of the component is the portion of the transition sequence that occurs at the interfaces of the component.

Definition 3.8 *Let* $\alpha \in L^\infty(k)$ *be a transition sequence and for an index* $0 \leq a \leq k-1$, *let* V_i *be the firability vector of* C_a *at* M_i^k. *The* extended local history *of* C_a *in* α, *denoted* $<C_a>_\alpha$, *is the sequence obtained from* $V_0 t_1 V_1 t_2 V_2 \cdots$ *by*

1) *deleting all transitions that do not belong to* C_a,
2) *replacing every remaining* $t_{a-1,j}$, $v_{a,j}$ *and* $t_{a,j}$ *by* u_j, v_j *and* w_j, *respectively and then*
3) *replacing every maximal substring of identical vectors* $V_{i_1} V_{i_2} \cdots V_{i_l}$ *by a single occurrence of* V_{i_1}.

The local history *of* C_a *in* α, *denoted* $\ll C_a \gg_\alpha$, *is the firing sequence obtained from* $<C_a>_\alpha$ *by deleting the firability vectors. The* externally visible history *of* C_a *in* α, *denoted* $[C_a]_\alpha$, *is the sequence obtained from* $<C_a>_\alpha$ *by*

1) *deleting all internal transitions of C and then*

2) *replacing every maximal substring of identical vectors $V_{i_1} V_{i_2} \cdots V_{i_l}$ by a single occurrence of V_{i_1}.* □

We define $h(\alpha) = (<C_0>_\alpha, <C_1>_\alpha, \cdots, <C_{k-1}>_\alpha)$.

Example 3.9 Consider the ring in Figure 4 consisting of two components of dining philosophers. The component C_0 has six interface transitions; three left interface transitions $t_{1,2}$, $t_{1,3}$, $t_{1,4}$ and three right interface transitions $t_{0,2}$, $t_{0,3}$ and $t_{0,4}$. Let the set $T^k = T$, i.e., the set of transitions occurring fairly, be the set of all transitions of the net. Then one legal transition sequence in the ring of two components is

$$\alpha = (t_{0,1} t_{0,2} t_{0,3} t_{1,1} t_{1,2} t_{1,3} t_{0,4} t_{0,5} t_{1,4} t_{1,5})^\omega.$$

The only places that are not input places of any interface transition of C_0 are $p_{0,6}$ and $p_{1,6}$, so the firability vectors of C_0 are vectors indicating the token counts of all other ten places $p_{0,4}$, $p_{1,2}$, $p_{1,4}$, $p_{1,5}$, $p_{1,3}$, $p_{1,1}$, $p_{0,2}$, $p_{0,5}$, $p_{0,3}$, $p_{0,1}$.In the transition sequence α the extended local history of C_0 is an infinite sequence of

$$\left(\begin{bmatrix} 1\\0\\1\\0\\0\\1\\0\\1\\0\\1 \end{bmatrix} v_1 \begin{bmatrix} 1\\0\\1\\0\\0\\1\\1\\1\\0\\0 \end{bmatrix} w_2 \begin{bmatrix} 0\\0\\0\\0\\0\\1\\0\\0\\1\\0 \end{bmatrix} w_3 \begin{bmatrix} 1\\0\\1\\1\\0\\1\\0\\0\\0\\1 \end{bmatrix} \begin{bmatrix} 1\\1\\1\\1\\0\\0\\0\\0\\0\\1 \end{bmatrix} u_2 \begin{bmatrix} 0\\0\\0\\0\\1\\0\\0\\0\\0\\1 \end{bmatrix} u_3 \begin{bmatrix} 1\\0\\1\\0\\0\\1\\0\\1\\0\\1 \end{bmatrix} w_4 \begin{bmatrix} 1\\0\\1\\1\\0\\1\\0\\0\\0\\0 \end{bmatrix} v_5 \begin{bmatrix} 1\\0\\1\\1\\0\\1\\0\\0\\0\\1 \end{bmatrix} u_4 \begin{bmatrix} 1\\0\\1\\0\\0\\0\\0\\1\\0\\1 \end{bmatrix} \begin{bmatrix} 1\\0\\1\\0\\0\\1\\0\\1\\0\\1 \end{bmatrix} \right)^\omega$$

The local history is the extended local history without the firability vectors, i.e., $(v_1 w_2 w_3 u_2 u_3 w_4 v_5 u_4)^\omega$. The externally visible history is obtained from the extended local history by deleting the internal transitions of C_0, i.e., the transitions v_1 and v_5. □

Let us now define the similarity between the components. For each $0 \le i \le k-1$, let $\mathcal{L}_{\neg i}(k)$ denote the set of transition sequences $\alpha \in L^\infty(R^k, M^k)$ that are t-legal for all transitions t of R^k except possibly the internal transitions of C_i.

Definition 3.10 (Li et al. 7) *Two rings R^k and R^l are* similar*, denoted by $R^k \sim R^l$, if*

1) $\{<C_0>_\alpha | \alpha \in \mathcal{L}(R^k, M^k, f^k)\} = \{<C_0>_\alpha | \alpha \in \mathcal{L}(R^l, M^l, f^l)\}$ *and*

2) $\{<C_i>_\alpha | \alpha \in \mathcal{L}(R^k, M^k, f^k)\} = \{<C_j>_\alpha | \alpha \in \mathcal{L}(R^l, M^l, f^l)\}$ *for any* $1 \leq i \leq k-1$ *and* $1 \leq j \leq l-1$.

Two rings are strongly similar*, denoted by $R^k \approx R^l$, if*

1) $\{<C_0>_\alpha | \alpha \in \mathcal{L}_{\neg 0}(k)\} = \{<C_0>_\alpha | \alpha \in \mathcal{L}_{\neg 0}(l)\}$ *and*

2) $\{<C_i>_\alpha | \alpha \in \mathcal{L}_{\neg i}(k)\} = \{<C_j>_\alpha | \alpha \in \mathcal{L}_{\neg j}(l)\}$ *for any* $1 \leq i \leq k-1$ *and* $1 \leq j \leq l-1$. □

The similarity means that none of the copies of C knows in which ring it is in and only the component C_0 knows which copy of C it is as long as the components behave legally. The strong similarity guarantees the same for any component of C that may violate the equation 3.1 for its internal transitions, as long as all other components behave legally.

Theorem 3.11 *If R^2 is structurally bounded and $R^2 \approx R^3$, then $R^2 \sim R^k$ for any $k \geq 3$.*

Proof. The proof is given in (Li et al. 1994a). □

Theorem 3.11 is applicable to finite-state processes only, so we must guarantee that our ring is structurally bounded. In (Li et al. 1994a) it is shown that a ring of any size is structurally bounded, if the ring of two components is structurally bounded. To apply theorem 3.11 to the dining philosophers' problem we must first show that the solution to the problem, consisting of two philosopher components is structurally bounded.

Lemma 3.12 *The ring R^2, given in Figure 4 and describing a solution to the dining philosophers' problem, is structurally bounded.*

Proof. It is shown in (Murata 1989) that a Petri net is structurally bounded, if and only if there exists a $|P|$-vector y of positive integers such that $Cy \leq 0$, where C is the *incidence matrix* of a net. The matrix shows the effect of a transition t_i on the place p_j, when the transition fires. Thus, the incidence matrix is defined as $C = [c_{i,j}]$, where $c_{i,j} = F(t_i, p_j) - F(p_j, t_i)$. It is easy to check that for example the vector $y = [1\ 1\ 4\ 1\ 1\ 1\ 1\ 1\ 1\ 4\ 1\ 1\ 1]^{\mathrm{T}}$ satisfies the requirement of the structural boundedness.

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 4 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 4 \\ 1 \\ 1 \\ 1 \end{bmatrix} \leq 0.$$

□

However, the nets R^2 and R^3 are not strongly similar. Consider the component C_0 of R^3. It has three left interface transitions $t_{2,2}$, $t_{2,3}$ and $t_{2,4}$ and three right interface transitions $t_{0,2}$, $t_{0,3}$ and $t_{0,4}$. These transitions have eleven input places, namely $p_{2,2}$, $p_{2,4}$, $p_{2,5}$, $p_{0,4}$, $p_{2,3}$, $p_{2,1}$, $p_{0,2}$, $p_{0,5}$, $p_{1,4}$, $p_{0,3}$ and $p_{0,1}$. Thus R^2 and R^3 cannot be similar, because the cardinality of their firability vectors is not the same.

3.2.2 Weak Similarity

Using theorem 3.11 we cannot show that the solution to the dining philosophers' problem is starvation-free for any number of philosophers. A similar problem arises in fact quite often in the analysis of identical processes. The ring, consisting of two components, is not similar to the ring, consisting of three components, even though R^3 is often similar with all R^k with $k \geq 4$. Fortunately, in (Li et al. 1994b) the structural induction theorem is extended to overcome this difficulty. Before the theorem we define the weak similarity of components. Let $\mathcal{L}(k)$ be $\mathcal{L}(R^k, M^k, f^k)$.

Definition 3.13 (Li et al. b 7) *Two rings R^k and R^l are* weakly similar*, denoted by $R^k \overset{\bullet}{\sim} R^l$, if*

1) $\{\ll C_0 \gg_\alpha | \alpha \in \mathcal{L}(k)\} = \{\ll C_0 \gg_\alpha | \alpha \in \mathcal{L}(l)\}$ *and*

2) $\{\ll C_i \gg_\alpha | \alpha \in \mathcal{L}(k)\} = \{\ll C_j \gg_\alpha | \alpha \in \mathcal{L}(l)\}$ *for any* $1 \leq i \leq k-1$ *and* $1 \leq j \leq l-1$. □

In spite of using the extended local histories of the components the weak similarity is defined using local histories, which are easier to compute but which can be used in the analysis of the temporal properties of the rings as well as extended local histories. In addition to the weak similarity, we shall define two operations, which assures that in any legal transition sequence of R^{k-1}, either of the two chains of length $r-1$, where $2 \leq r < k/2$, adjacent to C_0 can be replaced by a chain of length r to obtain a legal sequence in R^k and in R^k either of the two legal chains of length r adjacent to C_0 can be replaced by a legal chain of length $r-1$ to obtain a transition sequence in R^{k-1}. Informally, any ring of size k, the number of components of which can be changed using these operations, is weakly similar to the ring of size l for all $l \geq k$.

Definition 3.14 (Li et al. b 8) *For* $2 \leq r \leq k-1$, R^k *is* strongly r-mergeable *if:*

1) *for any* $\alpha \in L^\infty(R^k, M^k)$ *that is legal at* $C_1, C_2, \cdots, C_r$, *there exists* $\beta_0 \in L^\infty(k-1)$ *that is legal at* $C_1, C_2, \cdots, C_{r-1}$, *such that:*

$$h(\beta_0) = (<C_0>_\alpha, \underbrace{<C_1>_{\beta_0}, \cdots, <C_{r-1}>_{\beta_0}}_{[C_1 \oplus C_2 \oplus \cdots \oplus C_r]_\alpha}, <C_{r+1}>_\alpha, \cdots, <C_{k-1}>_\alpha)$$

and

2) *for any* $\alpha \in L^\infty(R^k, M^k)$ *that is legal at* $C_{k-r}, C_{k-r+1}, \cdots, C_{k-1}$, *there exists* $\beta_1 \in L^\infty(R^{k-1}, M^{k-1})$ *that is legal at* $C_{k-r}, C_{k-r+1}, \cdots C_{k-2}$, *such that:*

$$h(\beta_1) = (<C_0>_\alpha, \cdots, <C_{k-r-1}>_\alpha, \underbrace{<C_{k-r}>_{\beta_1}, \cdots, <C_{k-2}>_{\beta_1}}_{[C_{k-r} \oplus C_{k-r+1} \oplus \cdots \oplus C_{k-1}]_\alpha}).$$

□

Definition 3.15 (Li et al. b 9) *For* $2 \leq r \leq k$, R^k *is* (r-1)-splittable *if, for any* $\alpha \in \mathcal{L}(k)$, *there exists* $\beta_0, \beta_1 \in \mathcal{L}(k+1)$ *such that:*

$$h(\beta_0) = (<C_0>_\alpha, \underbrace{<C_1>_{\beta_0}, \cdots, <C_r>_{\beta_0}}_{[C_1 \oplus C_2 \oplus \cdots \oplus C_{r-1}]_\alpha}, <C_r>_\alpha, <C_{r+1}>_\alpha, \cdots, <C_{k-1}>_\alpha)$$

and

$$h(\beta_1) = (<C_0>_\alpha, \cdots, <C_{k-r}>_\alpha, \underbrace{<C_{k-r+1}>_{\beta_1}, <C_{k-r+2}>_{\beta_1}, \cdots, <C_k>_{\beta_1}}_{[C_{k-r+1} \oplus C_{k-r+2} \oplus \cdots \oplus C_{k-1}]_\alpha}).$$

□

To be able to infer about the similarity between two rings using `merge` and `split` operations we must ensure that the original components of the ring obtained using the operations are similar to the new components obtained in the replacement. If all the components of the ring, except perhaps the component C_0, have the same local histories, the ring is called uniform.

Definition 3.16 (Li et al. b 10) *R^k is* uniform *if, for any $1 \leq i, j \leq k-1$ and $\alpha \in \mathcal{L}(k)$, there exists $\beta \in \mathcal{L}(k)$, such that $\ll C_i \gg_\alpha = \ll C_j \gg_\beta$.*

Now we can give theorem, according to which all rings of size $k-1$ or greater exhibit "similar" behaviour for $k \geq 5$, when the rings $k-1$ and k satisfies three conditions given in definitions 3.14–3.16.

Theorem 3.17 *If R^2 is structurally bounded and for some $2 \leq r < k/2$, R^k is strongly r-mergeable, R^{k-1} is $(r-1)$-splittable and R^{k-1} and R^k are both uniform, then $R^{k-1} \stackrel{\bullet}{\sim} R^l$ for all $l \geq k$.*

Proof. The proof is given in (Li et al. 1994b). □

The idea of theorem 3.17 is that if a ring R^{k-1} is $(r-1)$-splittable, the ring of size k obtained using the `split` operation must have a behaviour, which is a subset of the behaviour of the ring of k processes, because R^k is uniform. In the similar way, if R^k is r-mergeable, the ring of size $k-1$ obtained using the `merge` operation must have a behaviour, which is a subset of the behaviour of the original ring of size $k-1$, because R^{k-1} is uniform. Thus, if R^k is r-mergeable and R^{k-1} $(r-1)$-splittable, the rings R^k and R^{k-1} must have a similar behaviour, which is in addition finite, because R^2 is structurally bounded.

3.3 A Starvation-Free Solution to the Problem of Dining Philosophers

In this section we analyze a starvation-free solution to the dining philosophers' problem. The solution is originally given in (Wu and Murata 1983) for five philosophers. We shall now prove using theorem 3.17 that the solution is starvation-free for any number of philosophers.

The dining philosophers' problem represents a *synchronization problem* of a large number of concurrent systems. The problem is stated as follows. A number of philosophers is sitting around a round table. Between every two philosophers there is a chopstick. In order to eat, a philosopher needs to pick up a chopstick on its right and on its left side, which means that two neighbours cannot eat simultaneously (a *mutual exclusion requirement*). This represents the synchronization problem of a number of processes (philosophers) and the same number of resources (chopsticks). Every philosopher can be in one of the three states: thinking, waiting or eating. Three places, named pt_i, pw_i and pe_i represent these states in Figure 3. The thinking state represents a non-critical section of any duration. The waiting state represents philosopher's desire to eat. The critical section is represented by the eating state of a finite duration. The dining philosophers' problem is to control the system in such a way that the duration of the waiting state of each philosopher is finite.

In Figure 3 an allowance card is used to guarantee the starvation-freeness. A philosopher who has the card can proceed from the waiting state to the eating state. The card is given to the neighbour after the eating state or in the thinking state, in case the philosopher is not hungry. The original idea of Wu and Murata was that in the solution as many allowance cards are used as there can be philosophers eating at the same time. We, however, use only one card, because induction theorem 3.17 cannot be applied, if the initial marking of the net changes from ring to ring. Thus, the mutual exclusion requirement holds trivially and we only need to show that the waiting time of no philosopher is infinite.

We start by showing that the rings R^8 and R^9 are uniform. But first we prove a helpful lemma. We have used the predicate/transition net reachability analysis tool `PROD` (Grönberg, Tiusanen and Varpaaniemi 1993, Varpaaniemi et al. 1995) in the proof of the lemma and in all subsequent proofs concerning the properties of reachability graphs. A transition sequence satisfies a formula $t \Rightarrow \Diamond t'$, if every occurrence of a transition t is followed by an occurrence of t' after a finite number of fired transitions.

Lemma 3.18 *Any transition sequence in $\mathcal{L}(8)$, i.e., in $\mathcal{L}(R^8, M^8, f^8)$ (and $\mathcal{L}(9)$) satisfies $v_{i,1} \Rightarrow \Diamond t_{i,2}$ and $t_{i,2} \Rightarrow \Diamond t_{i,3}$.*

Proof. The reachability graph of R^8 (R^9) contains 69,984 (236,196) states and 414,072 (1,554,957) arrows. We can find every state, in which a transition $v_{i,1}$ ($t_{i,2}$) is enabled for any $0 \leq i \leq 7$ ($0 \leq i \leq 8$) with the `PROD` query `enabled`.

```
define enabled(transition,philosopher) query
volatile node not(not(fire(transition(philosopher==p)) true))
```

For example the reachability graph of R^8 contains 22,599 (2,187) states (R^9 contains 76,545 (6,561) states), in which $v_{1,1}$ ($t_{1,2}$) is enabled. These states are found with the query `enabled`$(v_{i,1}, 1)$ (`enabled`$(t_{i,2}, 1)$). From all of the states, found with `enabled`, we have to search for infinite transition sequences, in which $t_{i,2}$ ($t_{i,3}$) is never fired. The `PROD` query `fired`, given below, returns every such infinite transition sequence. Note that the query is in such a form that firing $t_{i,2}$ (or $t_{i,3}$) empties the wait place.

```
define fired(transition,philosopher,wait_place) query node
not(not(fire(transition(philosopher==p)) true)) and dspan
((((wait_place) : (field[0] == philosopher)) != empty), true) false
```

PROD returns an empty set to every query `fired` for the transition $v_{i,1}$ ($t_{i,2}$) and $0 \leq i \leq 7$ (and $0 \leq i \leq 8$). Thus we can conclude that any sequence in $\mathcal{L}(8)$ (and $\mathcal{L}(9)$) satisfies the claim of this lemma. $\square$

Lemma 3.19 *R^8 and R^9 are uniform.*

Proof. We show only that R^8 is uniform. The uniformity of R^9 is proven in a similar way and thus omitted.

Transition sequences $\gamma = v_{0,1}t_{0,2}t_{0,3}$ and $\gamma' = t_{0,4}v_{0,5}$ change the initial marking $(M, M', M', M', M', M', M', M')$ to $(M', M, M', M', M', M', M', M')$ so that the marking of C_0 (M) becomes the marking of C_1 and the marking of all other components remain unchanged (in the marking M'). Thus, for any $1 \leq i \leq 7$ and $\alpha \in \mathcal{L}(8)$, there exists $\gamma\beta, \gamma'\beta \in \mathcal{L}(8)$, such that $\ll C_i \gg_\alpha = \ll C_{i+1} \gg_{\gamma\beta}$ or $\ll C_i \gg_\alpha = \ll C_{i+1} \gg_{\gamma'\beta}$. Therefore, it suffices to show that for any $\alpha \in \mathcal{L}(8)$, there exists $\beta \in \mathcal{L}(8)$, such that $\ll C_1 \gg_\alpha = \ll C_7 \gg_\beta$. Because all the transitions of the net are in T^8, we have to examine transition sequences containing every transition, i.e., α and β must contain every transition that is enabled infinitely often infinitely many times. Thus any local history of C_1 in the transition sequence α is of the form $[(t_4v_5)^a(v_1t_2t_3)^b(t_4v_5)^c]^\omega$, where $a, c \geq 0$ and $b > 0$ are all finite. In addition, α is legal and belongs to $\mathcal{L}(8)$, if $a + c > 0$. Thus the philosopher can pass the allowance card to the next philosopher any finite number of times. When the philosopher becomes hungry, by lemma 3.18 it cannot pass the card any more but begins to eat. After eating the philosopher can again pass the card any finite number of times.

Consider now the component C_7. It is straightforward to see that for example the transition sequence $\beta = [(t_{0,4}v_{0,5}t_{1,4}v_{1,5} \cdots t_{6,4}v_{6,5}t_{7,4}v_{7,5})^a(v_{0,1}t_{0,2}t_{0,3}v_{1,1}t_{1,2}t_{1,3} \cdots v_{6,1} t_{6,2}t_{6,3}v_{7,1}t_{7,2}t_{7,3})^b(t_{0,4}v_{0,5}t_{1,4}v_{1,5} \cdots t_{6,4}v_{6,5}t_{7,4}v_{7,5})^c]^\omega$ belongs to $\mathcal{L}(8)$ and $\ll C_7 \gg_\beta = [(t_4v_5)^a(v_1t_2t_3)^b(t_4v_5)^c]^\omega$, where $a, c \geq 0$ and $b > 0$ are all finite and and $a + c > 0$. $\square$

We have proven the uniformity of R^8, because either R^6 or smaller rings are not 1- or 2-splittable and R^7 or smaller rings are not 2- or 3-mergeable.

Lemma 3.20 *R^8 is 3-splittable.*

Proof. We have to show that in any legal transition sequence $\mathcal{L}(8)$ the externally visible history of the component $C_1 \oplus C_2 \oplus C_3$ ($C_5 \oplus C_6 \oplus C_7$) is the same as the the externally visible history of the component $C_1 \oplus C_2 \oplus C_3 \oplus C_4$ ($C_5 \oplus C_6 \oplus C_7 \oplus C_8$) in some transition sequence $\mathcal{L}(9)$. We only show that such a transition sequence exists for the component $C_1 \oplus \cdots \oplus C_3$. The proof of the existence of the transition sequence for $C_5 \oplus \cdots \oplus C_7$ is similar and thus omitted.

We show that every transition of the component $C_1 \oplus \cdots \oplus C_3$ has the same effect on the firability vector V_1 of the component as have some transitions of the component $C_1 \oplus \cdots \oplus C_4$ on its firability vector V_2. In addition, we show that the components have similar interface transitions and those transitions of $C_1 \oplus \cdots \oplus C_4$ that do not correspond any transition of $C_1 \oplus \cdots \oplus C_3$ can be fired without affecting V_2. All this will ensure the existence of $\beta \in \mathcal{L}(9)$ such that for any transition sequence α in $\mathcal{L}(8)$ the externally visible history of $C_1 \oplus \cdots \oplus C_3$ in α is the same as the externally visible history of $C_1 \oplus \cdots \oplus C_4$ in β.

The firability vectors of the components $C_1 \oplus \cdots \oplus C_3$ and $C_1 \oplus \cdots \oplus C_4$ consist of the places $p_{0,2}$, $p_{0,4}$, $p_{0,5}$, $p_{1,4}$, $p_{0,3}$, $p_{0,1}$, $p_{3,2}$, $p_{3,4}$, $p_{3,5}$, $p_{4,4}$, $p_{3,3}$, $p_{3,1}$ and $p_{0,2}$, $p_{0,4}$, $p_{0,5}$, $p_{1,4}$, $p_{0,3}$, $p_{0,1}$, $p_{4,2}$, $p_{4,4}$, $p_{4,5}$, $p_{5,4}$, $p_{4,3}$, $p_{4,1}$, respectively. We have to show that for every firing of a transition (and its effect on V_1) in any legal transition sequence $\mathcal{L}(8)$ in $C_1 \oplus \cdots \oplus C_3$ there exits a transition, the firing of which has the same effect on V_2 and the transition belongs to some legal transition sequence $\mathcal{L}(9)$.

It is clear that the left interface transitions ($t_{0,2}$, $t_{0,3}$ and $t_{0,4}$) of the components affect the components in a similar way, because they have the same left interface transitions and places. The internal transition of $C_1 \oplus \cdots \oplus C_3$ ($t_{1,1}$, $t_{1,2}$, $t_{1,3}$, $t_{1,4}$, $t_{1,5}$ and $t_{2,1}$) affect only those places, which are common to both components. The rest of the internal transitions of $C_1 \oplus \cdots \oplus C_3$ ($t_{2,2}$, $t_{2,3}$, $t_{2,4}$, $t_{2,5}$, $t_{3,1}$ and $t_{3,5}$) affect only those places of V_1, which do not belong to V_2. These internal transitions have the same effect on V_1 as the internal transitions $t_{3,2}$, $t_{3,3}$, $t_{3,4}$, $t_{3,5}$, $t_{4,1}$ and $t_{4,5}$ of $C_1 \oplus \cdots \oplus C_4$ have on V_2. Both components have similar right interface transitions $t_{3,2}$, $t_{3,3}$, $t_{3,4}$ and $t_{4,2}$, $t_{4,3}$, $t_{4,4}$, respectively, the firing of which have the same effect on both components.

We still have to show, how the firing of internal transitions of C_2 in the component $C_1 \oplus \cdots \oplus C_4$ affects V_2. Because no place of V_2 is connected to any transition of C_2, it is clear that the firing of any transition in C_2 has no effect on V_2. So we can fire the transitions of C_2 in such a way that the fairness requirement is satisfied. $\square$

Lemma 3.21 *R^9 is strongly 4-mergeable.*

Proof. As in lemma 3.20 we have to show that in any legal transition sequence $\mathcal{L}(9)$ the externally visible history of the component $C_1 \oplus C_2 \oplus C_3 \oplus C_4$ ($C_5 \oplus C_6 \oplus C_7 \oplus C_8$) is the same as the the externally visible history of the component $C_1 \oplus C_2 \oplus C_3$ ($C_5 \oplus C_6 \oplus C_7$) in some transition sequence $\mathcal{L}(8)$. We only show that such a transition sequence exists for the component $C_1 \oplus \cdots \oplus C_4$. The proof of the existence of the transition sequence for $C_5 \oplus \cdots \oplus C_8$ is similar and thus omitted.

The firability vectors (V_1 and V_2) of the components $C_1 \oplus \cdots \oplus C_4$ and $C_1 \oplus \cdots \oplus C_3$ consist of the places $p_{0,2}$, $p_{0,4}$, $p_{0,5}$, $p_{1,4}$, $p_{0,3}$, $p_{0,1}$, $p_{4,2}$, $p_{4,4}$, $p_{4,5}$,

$p_{5,4}$, $p_{4,3}$, $p_{4,1}$ and $p_{0,2}$, $p_{0,4}$, $p_{0,5}$, $p_{1,4}$, $p_{0,3}$, $p_{0,1}$, $p_{3,2}$, $p_{3,4}$, $p_{3,5}$, $p_{4,4}$, $p_{3,3}$, $p_{3,1}$, respectively. As in lemma 3.20 the components share the left interface transitions and places, so the effect of the firing of left interfaces transitions is the same on both V_1 and V_2. The only internal transitions of $C_1 \oplus \cdots \oplus C_4$ that affect V_1 are $t_{1,2}$, $t_{1,3}$, $t_{3,2}$, $t_{3,3}$, $t_{3,4}$, $t_{4,1}$ and $t_{4,5}$. The internal transitions $t_{1,2}$, $t_{1,3}$, $t_{2,2}$, $t_{2,3}$, $t_{2,4}$, $t_{3,1}$ and $t_{3,5}$ of $C_1 \oplus \cdots \oplus C_3$ have the same effect on V_2. The right interface transitions $t_{4,2}$, $t_{4,3}$, $t_{4,4}$ and $t_{3,2}$, $t_{3,3}$, $t_{3,4}$ have the same effect on V_1 and V_2, respectively. The internal transitions of C_2 in $C_1 \oplus \cdots \oplus C_4$ has no effect on V_1, so no transition is fired in $\mathcal{L}(8)$, when a transition, belonging to C_2 is fired in $\mathcal{L}(9)$. □

Lemma 3.22 *Every hungry philosopher will eventually eat in the solution to the dining philosophers' problem, consisting of eight philosophers.*

Proof. The proof is similar to lemma 3.18, in which we have shown with `PROD` that any transition sequence in $\mathcal{L}(8)$ satisfies $v_{i,1} \Rightarrow \Diamond t_{i,2}$. □

In (Wu and Murata 1983) the property is proven for R^5 using net invariants. Now we can combine all the lemmas and prove that the solution to the dining philosophers' problem is correct, regardless the number of philosophers.

Theorem 3.23 *For any $k \geq 8$, every hungry philosopher will eventually eat in the ring, consisting of k philosophers.*

Proof. The theorem follows from theorem 3.17 and lemmas 3.12, 3.19, 3.20, 3.21 and 3.22. □

We can use for example `PROD` to show that the solution works, when the number of philosophers is 2,3,4,5,6 or 7 and thus conclude that the solution is starvation-free with any number $k \geq 2$ of philosophers.

4 An Induction Theorem for Ring Protocols of Processes of Pr/T-Nets

It is usually easy to show using the induction theorem described in the preceding chapter that a system, consisting of any number of finite components, has a behaviour satisfying some specification. Small systems consisting of a few components, can be automatically analyzed using any verification tool. For example with `PROD` it can be verified, if a system satisfies any branching time temporal logic specification that can be expressed with CTL (Clarke et al. 1986a). It is also easy to show, whether the system is uniform, splittable and mergeable. However, the proofs can be sometimes quite lengthy, because of the great number of branches in the reachability graph caused by different interleavings of actions.

The induction theorem has a couple of weaknesses and that is why it is not always feasible to apply it to the analysis of a real system. One of the weaknesses is that the theorem is applicable to low-level Petri nets only. Real systems are so complex that modelling them using low-level Petri nets leads to huge net models containing hundreds of places and transitions. In many situations it is more reasonable to use high-level Petri nets, for example predicate/transition nets (Genrich 1987, Genrich 1989) in the modelling of identical processes. Another weakness of the theorem is that protocols, in which addresses are used in communication, cannot be successfully verified using theorem 3.17.

Consider a system of n processes. When a new process is added to the system, each process must be able to communicate with it. Modelling this system using low-level Petri nets, new transitions must be added to the net components modelling the processes, because all tokens are identical undistinguishable black tokens. The components do not know the new component, unless they have transitions with which they can change tokens with the new component. Therefore the structure of net components must be changed, when the number of processes changes, which makes the application of theorem 3.17 impossible.

In this chapter we extend the theorem of the preceding chapter to overcome these weaknesses, i.e., we develop and define an induction theorem for ring protocols of processes of Pr/T-nets. We give an identity to the components so that they are able to communicate with each other. Using Pr/T-nets, components can be identified by using one field in the tuple as an identification of the component at each marking. The structure of the underlying Petri net of a Pr/T-net does not change, when processes are added or removed in the system, because only the tuple tokens of the net are changed. The usage of address information is very desirable in the modelling of many real systems and even necessary, when we want to analyze communication protocols for example. A component, i.e., a node in the communication network, must know the address of the node with which it communicates.

We first give a short introduction to the Pr/T-net formalism. After that we define the concept of similarity between Pr/T-net components. Finally we present the induction theorem for communication ring protocols (or any other ring protocol, in which we want to identify the components) of processes described using Pr/T-nets.

4.1 Predicate/Transition Nets

A predicate/transition net is a Petri net, whose places, transitions and arcs are annotated with a set of variable predicates, well-formed formulas and symbolic sums of tuples of a first-order structure for a first-order language L. A marking of a net consists of tuples of individuals in the places, which denote the interpretation of the relation symbols, i.e., the state of the concurrent system. We first give the definitions for the structure and the language, used in the annotation of a Pr/T-net.

Definition 4.1 *A* structure *is a tuple of objects,* $\mathcal{R} = (D; f_1, ..., f_k; R_1, ..., R_n)$, *where* D *is a non-empty set of* individuals *called the* domain *or the* carrier *of* $\mathcal{R}$, f_i *are* functions *in* D *and* R_j *are* relations *in* D. □

Definition 4.2 *For each* $n \geq 0$, *a finite set of* n*-ary* operators *(function symbols) and a finite set of* n*-ary* predicates *(relation symbols) of the structure* $\mathcal{R}$ *form the* vocabulary *of the* first-order language L *that consists of three kinds of expressions: the set of* individual terms L_t, *the set of* first-order formulae L_f *and the set of* symbolic sums $LC^{(n)}$. *In addition, there is an infinite set of* variable symbols V *in* $\mathcal{R}$, *disjoint from operators and predicates. Expressions are built in the following way.*

L_t:

(a) *A variable is in* L_t.

(b) *If* $f^{(n)}$ *is a* n*-ary operator and* $v_1, ..., v_n$ *are terms, then* $f(v_1, ..., v_n)$ *is a term.*

(c) *No other expression is a term.*

L_f:

(a) *The constants* $\top$ *(true) and* $\perp$ *(false) are* atomic *formulae in* L_f.

(b) *If* v_1 *and* v_2 *are terms, then* $v_1 = v_2$ *is an atomic formula in* L_f.

(c) *If* $P^{(n)}$ *is a* n*-ary predicate and* $v_1, ..., v_n$ *are terms, then* $Pv_1...v_n$ *is an atomic formula in* L_f.

(d) If p_1 and p_2 are in L_f, then $\neg p_1$ and $(p_1 \vee p_2)$ are in L_f.

(e) If x is a variable and p is in L_f, then $(\exists x)p$ is in L_f.

(f) No other expression is in L_f.

$LC^{(n)}$:

(a) The constant 0 is in $LC^{(n)}$.

(b) If $v_1, ..., v_n$ are terms, then the n-tuple $\langle v_1, ..., v_n \rangle$ is in $LC^{(n)}$.

(c) If l_1, l_2 are in $LC^{(n)}$, then $(l_1 + l_2)$ is in $LC^{(n)}$.

(d) If l is in $LC^{(n)}$ and z denotes a non-negative integer, then zl is in $LC^{(n)}$.

(e) If l is in $LC^{(n)}$ and p is a formula, then $[p]l$ is in $LC^{(n)}$.

(f) If x is a variable and l is in $LC^{(n)}$, then $\sum_x l$ is in $LC^{(n)}$.

(g) No other expression is in $LC^{(n)}$. □

Now we can give a formal definition of a Pr/T-net. Let $LC_\Sigma(A)$ denote a set of symbolic sums of a set A and let A^* denote the set of all finite tuples of A.

Definition 4.3 *A* predicate/transition net *is a 3-tuple $N_L = \langle N, A, M_0 \rangle$, where*

1) N is an underlying Petri net, $N = (P, T; F)$,

2) A is the annotation *of N, $A = (A_N, A_P, A_T, A_F)$, where*

- *(a) $A_N = \mathcal{R}$ is a first-order structure for L, called the* support *of N.*
- *(b) A_P is a bijection between the set of places and the set of variable predicates that assigns an* arity *to each place. The arity is the size of tuples in that place.*
- *(c) $A_T : T \rightarrow L_f$ assigns a well-formed formula, called a* firing condition *to each transition. All variables that are outside the scope of quantifiers $\exists$ and $\forall$ must appear in some symbolic sum that A_f assigns to the arc hitting the transition.*
- *(d) A_F assigns to each arc a symbolic sum of $LC^{(n)}$. The arities of all the symbolic sums of tuples in an arc must equal to the arity of a place, which the arc hits.*

3) $M_0 : P \rightarrow LC_\Sigma(A_N^*)$ is the initial marking *of N. It assigns to each place a symbolic sum of tuples of the carrier of A_N. The size of the tuples must equal to the arity of the place.* □

When drawing Pr/T-nets we put firing conditions inside the transition squares, arc expressions beside the arcs and initial markings inside the circles. In the sequel, we shall use the word tuple as an abbreviation of the word tuple token, because all tokens of a Pr/T-net are tuples. In Figure 5 a Pr/T-net model of a simple communication protocol is given, where N (N > 1) processes use a critical section on a mutual exclusion basis. Note that the critical section is not explicitly modelled. The solution relys on the presence of a unique privilege token in the ring. A process, having a privilege token, is guaranteed to have an exclusive access to the critical section. In the initial marking of the net in Figure 5 the place `Idle` contains a pair $\langle 0, 0\rangle$, which denotes that a node 0 has a turn, i.e, the privilege token, the destination address of which is currently 0. The place `Addresses` has N 1-tuples, denoting all possible destination addresses. The system operates in such a way that a node having a turn can send it to any node in the communication network. A node, which does not have a turn, must forward the turn towards the destination node and this forwarding node cannot have access to the critical section. Note that the modelled solution is not fair. In (Martin 1985) three solutions to the distributed mutual exclusion problem are presented. The first solution, in which a privilege token circulates continuously is unfair, while the other two, where the token is moved only on request, are fair. Note also that applying theorem 3.17 it is impossible to examine the properties of the net model given in Figure 5, although the protocol is very simple.

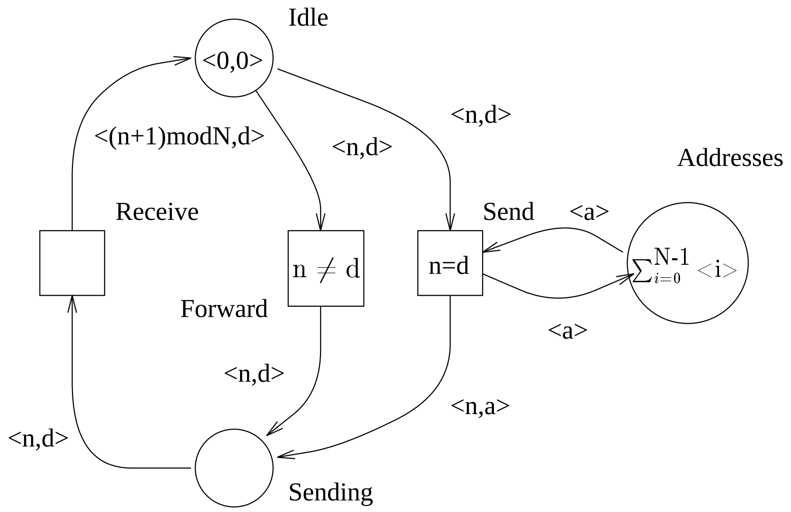


Figure 5: A Pr/T-net model of a simple, unfair communication protocol.

Transitions of a Pr/T-net change the tuples of individuals in the places, i.e., change the state of the concurrent system. The tuples of individuals at different markings form the set of vertices V and the transitions that change a marking of the net to another marking form the set of edges E of the reachability graph of the net. To be able to formally define the reachability graph we must define, how the transitions are fired and how we handle the symbolic sums of the arc expressions.

If a formula F has variables outside the scope of quantifiers $\exists$ and $\forall$, it is called *open*. Otherwise it is called *closed* or a *proposition*. The occurrences of variables in a single term are always outside the scope of quantifiers. Closed terms are *compound names* of individuals. Variables being outside the scope of quantifiers are called *free* and can be replaced by terms in expressions.

Definition 4.4 *Let $\alpha = x_1 \leftarrow v_1, \cdots, x_n \leftarrow v_n$ be a* substitution, *where x_is are variables and v_is are terms. If E is an expression, $E : \alpha = E : \{x_1 \leftarrow v_1, \cdots, x_n \leftarrow v_n\}$ designates the result, when every free occurrence of x_i is substituted by v_i in an expression E. $E : \alpha$ is called an α-*instance *of E.* □

Any closed expression E has an *interpretation* $\mathcal{R}(E)$, which is determined by the interpretation of its subexpressions: terms, operators, predicates and constants. The structure $\mathcal{R}$ assigns to each closed term an individual of $\mathcal{R}$ and to each closed formula the truthvalue ($\top$ or $\bot$).

Definition 4.5 *If every operator $f^{(m)}$ and every predicate $P^{(n)}$ of a first-order language L denotes an m-ary function of $\mathcal{R}$, designated by $f_{\mathcal{R}}$ and n-ary relation of $\mathcal{R}$, designated by $P_{\mathcal{R}}$, respectively, $\mathcal{R}$ is called a* structure for L *or a* L-structure. □

Definition 4.6 *Let ϕ be a formula of L. ϕ is said to be* valid *in $\mathcal{R}$, if and only if each closed instance of ϕ is true in $\mathcal{R}$, which is denoted by $\mathcal{R} \models \phi$.* □

Definition 4.7 *If $\mathcal{R} \models (A_T(t)) : \alpha$ (an expression resulting from the substitution, where all the free variables in the firing condition of the transition t are substituted by α, is true in the L-structure of $\mathcal{R}$) and the closed arc expression $A_F(p,t)$ and $A_F(t,s)$ have an interpretation, the α-instance of t is called* feasible. □

Consider again Figure 5. Let $\alpha = \{\mathrm{n} \leftarrow 5, \mathrm{d} \leftarrow 32\}$. Then the α-instance of the transition `Forward` is feasible, because $\mathbb{Z} \models (\mathrm{n} \neq \mathrm{d}) : (\mathrm{n} \leftarrow 5, \mathrm{d} \leftarrow 32)$ and both arc expressions $\langle \mathrm{n}, \mathrm{d} \rangle$ of these arcs that hit `Forward` have an interpretation $\{\langle 5, 32 \rangle\}$.

Definition 4.8 *Let M and t be a marking and a transition of a Pr/T-net N, respectively and α be a substitution of t. At marking M a feasible α-instance $t : \alpha$ is* enabled, *denoted by $M[t : \alpha\rangle$, if*

$$\forall p \in P[\mathcal{R}(A_F(p,t) : \alpha) \subseteq \mathcal{R}(M(p)) \land \mathcal{R}(A_F(t,p) : \alpha) \cap \mathcal{R}(M(p)) = \emptyset].$$

□

Let $\alpha = \{\mathrm{n} \leftarrow 5, \mathrm{d} \leftarrow 32\}$. Then the feasible α-instance of the transition `Forward` in the net given in Figure 5 is not enabled, because $\{\langle 5, 32\rangle\} \not\subseteq \{\langle 0, 0\rangle\}$. Now we can give the definition of the firing of an enabled transition.

Definition 4.9 *Let M and M' be markings, t a transition and $t : \alpha$ a feasible α-instance of t of the Pr/T-net N. An α-instance $t : \alpha$ enabled at M can* fire *to a marking M', denoted by $M[t : \alpha\rangle M'$, if*

$$\forall p \in P \quad \mathcal{R}(M'(p)) = \mathcal{R}(M(p)) \backslash \mathcal{R}(A_F(p, t) : \alpha) \cup \mathcal{R}(A_F(t, p) : \alpha).$$

□

From now on we shall use the term transition instance, when we mean a feasible α-instance of a transition. Let $\alpha = \{\mathrm{n} \leftarrow 0, \mathrm{d} \leftarrow 0, \mathrm{a} \leftarrow 0\}$. Then the transition instance `Send` $: \alpha$ is enabled in the net of Figure 5. When the transition instance fires, $\langle 0, 0\rangle$ is removed from the place `Idle` and put into the place `Sending`.

Now we can finally give a formal definition for the reachability graph of a Pr/T-net.

Definition 4.10 *The* reachability graph *of a Pr/T-net is a pair $\langle V, E\rangle$, where V is the set of vertices and E is the set of edges. The initial marking $M_0 \in V$. If $M \in V$ and $M[t : \alpha\rangle M'$, then $M' \in V$ and $\langle M, t : \alpha, M'\rangle \in E$. There are no other elements in V and E.* □

Example 4.11 In Figure 6 the reachability graph of the Pr/T-net of Figure 5 is given, when N is 2. The graph contains eight vertices and ten edges. We have used only the first letters of the names of the transitions and the places to make the presentation more readable. □

4.2 Similarity of Pr/T-Net Components

Consider Figure 5 again. If we omit arc expressions and firing conditions from the figure and replace the initial marking, consisting of tuples, with an initial marking, consisting of black tokens, we get a low-level Petri net, representing the behaviour of a ring of one component. Note that the structure of the low-level net of one component is the same as the structure of the high-level net, i.e., the places, transitions and arcs are identical.

We would like to analyze the behaviour of the ring by analyzing only the behaviour of one of its components. This is not possible applying the methods of the preceding chapter, because even in the same ring different components have different tuples at markings and therefore the behaviour of the components is not similar, although the structure of each component is the same,

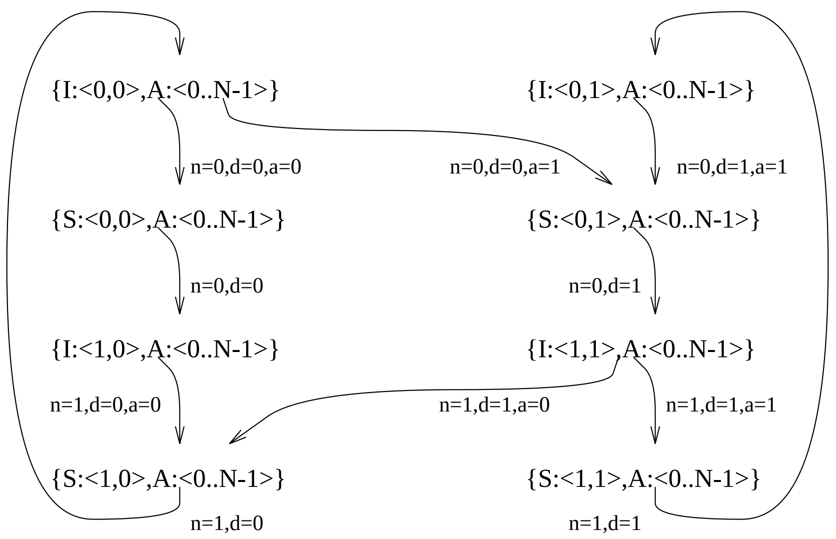


Figure 6: The reachability graph of the Pr/T-net given in Figure 4.1.

i.e., all net components have the same places, transitions and arcs. The similarity of the components could be analyzed using for example the theorem given in (Genrich 1989) which states that two Pr/T-nets are equivalent, if the elementary net systems (low-level Petri nets), generated by unfolding the Pr/T-nets, are equivalent. However, it is not reasonable to unfold a Pr/T-net, because the unfolding can increase the size of the net model enormously. In addition, we would be able to apply the induction theorem in a higher-level of abstraction. Note that the different components of the net of Figure 5 represent similar behaviours, where the only difference is the identity of the involved node or the address it uses. By mapping the identity and address of each node in the observable behaviour of the node using a *cyclic rotation* as in (Jensen 1987) we can abstract the effect of the address information on the behaviour of the components away. We define the rotation or a *symmetry* to be a bijective function from fields of tuples to fields of tuples in such a way that no information is lost, i.e., the enabling of transitions is preserved.

4.2.1 Pr/T-Net Components

All the Pr/T-net components of the system have the same places, transitions, firing conditions and arcs. The components are differentiated only by different tuples at the markings. The arc expression that closes the ring makes the behaviour of rings of different size dissimilar from each other, unless we also map the symbolic sums of tuples in arc expressions in such a way that the information about the size of the ring does not affect the behaviour. Because of the modelling flexibility of Pr/T-nets a component describing a process of the system can be be constructed in a number of ways. The most convenient construction for our purposes is such that makes the application of the induction theorem as easy as possible.

Definition 4.12 *A* component *is a Pr/T-net $C_i = \langle N, A, M_{0_i} \rangle$, where the set of transitions T of an underlying Petri net N, can be partitioned as $T = T_L \cup T_I \cup T_R$, such that $|T_L| = |T_R| \geq 1$. The transitions in T_L, T_I and T_R are called* left interface transitions, internal transitions *and* right interface transitions, *respectively.* □

Note that according to definition 4.12 all the components C_i (of the same net model) are the same Pr/T-net C with different initial marking M_{0_i}. In addition, all the components have the same interface and internal transitions, but because of different initial markings caused by different identities of the components, the transition instances of different components are not identical.

Definition 4.13 *Let C be a Pr/T-net component of a ring of size k, having an underlying net N, an annotation A and an initial marking M_0. For each $i \geq 0, C_i = \langle N, A, M_{0_i} \rangle$ denotes the Pr/T-net having the same structure as C. For $0 \leq i \leq j$:*

$$C_i \oplus C_{i+1} \oplus \cdots \oplus C_j = \langle N, A, \bigcup_{i \leq l \leq j} M_{0_l} \rangle$$

denotes the chain *consisting of the components $C_i, C_{i+1}, \cdots, C_j$, the initial markings of which are $M_{0_i}, M_{0_{i+1}}, \cdots, M_{0_j}$, respectively. For each $k \geq 2$:*

$$C_0 \odot C_1 \oplus \cdots \oplus C_{k-1} = \langle N, A, \bigcup_{0 \leq l < k} M_{0_l} \rangle$$

where all subscripts are taken modulo k, denotes the ring *consisting of the components $C_0, \cdots, C_{k-1}$, where the initial marking of C_0 is M_{0_0} and the initial marking of all other components $1 \leq l < k$ is M_{0_l}. The initial marking M_{0_l} of all the components C_l must be identical for all $1 \leq l < k$, except that the field indicating their identity can be different from component to component.* □

Intuitively the similarity between the components means that no component knows in which ring it is in and in which position in the ring it is. We have defined the chain and the ring in such a way that every component knows its position in the ring, i.e., its identity and every component knows, into which ring it belongs, i.e., the size of the ring. This means that neither the behaviour of the components nor the behaviour of the rings of different sizes are similar, although the structure of the underlying Petri net of the Pr/T-net components is identical. The behaviour of the components is not similar, because the symbolic sums of tuples, representing different components at a marking, are different. In a similar way feasible transition instances, representing rings of different sizes, make the behaviour of these rings different from each other.

To be able to make the observable behaviour of the components and rings identical, we use a set of bijective functions, called symmetries. Using these functions we map tuples of individuals, representing different components, to identical tuples with respect to the identity of the component. We also map arc expressions of feasible transition instances, representing rings of different size, to identical expressions with respect to the size of the ring.

Definition 4.14 *A* symmetry *is a set of bijective functions* $\varphi = \{\varphi_{A_T} \cup \varphi_{A_F} \cup \varphi_M\}$, *where*

1) φ_{A_T} *maps each firing condition, containing the ring size or the component identity, to a constant individual, representing the ring size or the component identity.*

2) φ_{A_F} *maps each arc expression, containing the ring size or the component identity, to a constant individual, representing the ring size or the component identity.*

3) φ_M *maps each tuple of each place at a marking, containing the ring size or the identity of a component, to a tuple, representing a marking and containing constant individuals, representing the ring size and the component identity.* □

Intuitively definition 4.14 means that all net components, the tuples of which have a different identity, but which are otherwise similar, are considered similar in rings of different sizes.

Example 4.15 Consider the net of Figure 5. Every component has one left interface transition `Receive`, which has the firing condition $\top$ and arc expressions $\langle \mathrm{n}, \mathrm{d} \rangle$ and $\langle (\mathrm{n}+1) \mathrm{mod} N, \mathrm{d} \rangle$, two internal transitions `Send` and `Forward` and one right interface transition `Receive`. Let the ring contain two components. By using the symmetries of definition 4.14, individuals of the firing conditions, all fields of the arc expressions and all tuples at markings, containing the ring size $N = 2$, the identity of the components or the address information, i.e., $n = 0$, $n = 1$, $a = 0$, $a = 1$, $d = 0$ or $d = 1$, are mapped to a constant individual, for example to 0. Let C_0 fire the transition `Send` with $a = 0$ and after that let C_0 (C_1) fire `Receive`. Then the place `Idle` contains a tuple $\langle 1, 0 \rangle$, which is identical to the tuple $\langle 0, 0 \rangle$, when we apply a symmetry function $\varphi_M = (i+1) \mathrm{mod} 2$ to get rid of the identity of C_1. □

4.2.2 Firability Vector

Because all our components contain the same places, we have to decide, which tuples in the same place belong to which components at a marking. Otherwise each component has the same tuples, which makes all components trivially identical. We decide that at a marking a tuple in a place belongs to a component i only, if the identity of the tuple is i, i.e., the tuple contains a field indicating an identity and holding the value i. In a similar way we identify the transition instances. A transition instance belongs to the component i only, if the identity of a transition instance is i, i.e., the arc expression contains a field indicating an identity and holding the value i. Informally this means that only a Pr/T-net component itself can change its own tuples by firing its own transition instances.

Definition 4.16 *At a marking M let $\langle tuple_{p,M}(i)\rangle$ denote a tuple in a place p, such that the tuple has the identity i, i.e., one of its fields carries the identity i. Then a transition instance $t:\alpha$ of a transition t belongs to i only, if*

$$\forall p \in P[\mathcal{R}(A_F(p,t):\alpha) \subseteq \langle tuple_{p,M}(i)\rangle].$$

We denote the function, which at a marking M selects only those tuples of a place p, which belongs to the component C_i, by TPL_i. □

As in chapter 3 we call input places of a left (right) interface transition *left interface* (*right interface*) *places* of a transition. A tuple of a component i in a left (right) interface place of a left (right) interface transition instance i is called a *left* (*right*) *interface tuple*.

Definition 4.17 *Let $p_i, \cdots, p_{i+n}$ and $p_j, \cdots, p_{j+m}$ be the the left and right interface places of the component C, respectively, $TPL_{a-1}, \cdots, TPL_{b+1}$ a set of selector functions, which select at each marking only those tuples, which belong to components $C_{a-1}, \cdots, C_{b+1}$, respectively and let φ_M be the set of bijective functions, which maps tuples of different components and of different rings to identical tuples with respect to the component identity and the ring size. Then, for a chain $C_a \oplus C_{a+1} \oplus \cdots \oplus C_b$ of length at most $k-1$ and a marking $(M_0, M_1, \cdots, M_{k-1})$, the* firability vector *of $C_a \oplus C_{a+1} \oplus \cdots \oplus C_b$ at*

$(M_0, M_1, \cdots, M_{k-1})$ *is a column vector:*

$$\begin{bmatrix} \varphi_{a-1}(TPL_{a-1}(M_{a-1}(p_j))) \\ \vdots \\ \varphi_{a-1}(TPL_{a-1}(M_{a-1}(p_{j+m}))) \\ \varphi_a(TPL_a(M_a(p_i))) \\ \vdots \\ \varphi_a(TPL_a(M_a(p_{i+n}))) \\ \varphi_b(TPL_b(M_b(p_j))) \\ \vdots \\ \varphi_b(TPL_b(M_b(p_{j+m}))) \\ \varphi_{b+1}(TPL_{b+1}(M_{b+1}(p_i))) \\ \vdots \\ \varphi_{b+1}(TPL_{b+1}(M_{b+1}(p_{i+n}))) \end{bmatrix}.$$

□

Example 4.18 Consider the Pr/T-net ring of size two in Figure 5. C_0 has one left interface place (`Sending`) and one right interface place (`Sending`), which are initially empty. Note that according to definition 4.12 the set of interface places of different components must be the same set of places, because only the tuples at markings make the components different. After C_0 has fired `Send` with $a = 1$, the place `Sending` contains a tuple $\langle 0, 1\rangle$, which is mapped to a tuple $\langle 0, 0\rangle$, which is the left interface tuple of a transition instance $\langle 1, 0\rangle$ of C_1 (and the right interface tuple of a transition instance $\langle 1, 0\rangle$ of C_0). Thus, at the initial marking the firing of `Send` with $a = 1$ changes the firability vector of C_1 from $\begin{bmatrix} x \\ y \end{bmatrix}$ to $\begin{bmatrix} \langle 0,0\rangle \\ y \end{bmatrix}$, where x and y are symbolic sums of left and right interface tuples of C_1, respectively, in the place `Sending` at the markings $\{I : \langle 0, 0\rangle, A : \langle 0\rangle + \langle 1\rangle\}$ and $\{S : \langle 0, 1\rangle, A : \langle 0\rangle + \langle 1\rangle\}$. (Note that x and y are empty.) □

4.2.3 Histories of Pr/T-Net Components

From now on we denote a transition instance $t : \alpha$ of a transition t by t^α.

Definition 4.19 *Let $\alpha = t_1^{\alpha_1} t_2^{\alpha_2} \cdots t_i^{\alpha_i} \cdots \in L^\infty(k)$ be a firing sequence of transition instances such that for each $0 \le i \le |\alpha|$, $M^k \rightarrow_{t_1^{\alpha_1} t_2^{\alpha_2} \cdots t_i^{\alpha_i}} M_i^k$. For an index $0 \le a \le k-1$, let V_i be the firability vector of C_a in M_i^k. The* extended local history *of C_a in α denoted by $\langle C_a\rangle_\alpha$ is the sequence obtained from $V_0 \varphi_{A_F}(t_1^{\alpha_1}) V_1 \varphi_{A_F}(t_2^{\alpha_2}) \cdots$ by*

1) deleting all transition instances that do not belong to C_a, *2) replacing every maximal substring of identical firability vectors $V_{i_1}V_{i_2}\cdots V_{i_l}$ by a single occurrence of V_{i_1}.* □

The *local history* of the component C_a in α denoted by $\ll C_a \gg_\alpha$ is obtained from $\langle C_a\rangle_\alpha$ by deleting all firability vectors. The *externally visible history* of the component C_a in α denoted by $[C_a]_\alpha$ is the sequence obtained from $\langle C_a\rangle_\alpha$ by deleting all internal transitions of C_a and then replacing every maximal substring of identical firability vectors $V_{i_1}V_{i_2}\cdots V_{i_l}$ by a single occurrence of V_{i_1}. The local history of a component tells what transition instances of the component are fired in a firing sequence. The externally visible history tells, what interface transition instances of the component are fired and how they affect the interface tuples of the component in a firing sequence. Note that in definition 4.19 all mapped transition instances of different components are identical, because all fields in the tuples of the arc expressions contain address information.

Example 4.20 Consider a following sequence of fired transitions of R^3 of the predicate/transition net given in Figure 5: $\alpha = (\text{Send}^{\langle 0,0\rangle,\langle 0\rangle}\ \text{Receive}^{\langle 0,0\rangle}\ \text{Forward}^{\langle 1,0\rangle}\ \text{Receive}^{\langle 1,0\rangle}\ \text{Forward}^{\langle 2,0\rangle}\ \text{Receive}^{\langle 2,0\rangle})^\omega$. It is easy to see that $\langle C_1\rangle_\alpha = \langle C_2\rangle_\alpha =$

$$\left(\begin{bmatrix}\langle 0,0\rangle\\ \\\end{bmatrix}\text{Receive}^{\langle 0,0\rangle,\langle 0\rangle}\begin{bmatrix} \\ \\\end{bmatrix}\text{Forward}^{\langle 0,0\rangle}\begin{bmatrix} \\ \langle 0,0\rangle\end{bmatrix}\text{Receive}^{\langle 0,0\rangle}\begin{bmatrix} \\ \\\end{bmatrix}\right)^\omega.$$

while $\langle C_0\rangle_\alpha =$

$$\left(\begin{bmatrix} \\ \\\end{bmatrix}\text{Send}^{\langle 0,0\rangle,\langle 0\rangle}\begin{bmatrix} \\ \langle 0,0\rangle\end{bmatrix}\text{Receive}^{\langle 0,0\rangle}\begin{bmatrix} \\ \\\end{bmatrix}\begin{bmatrix}\langle 0,0\rangle\\ \\\end{bmatrix}\text{Receive}^{\langle 0,0\rangle}\right)^\omega.$$

□

With the following lemma we show that two chains belonging to two rings are interchangeable, if the chains behave identically at the interfaces, i.e., their externally visible history is the same in all computations.

Lemma 4.21 *Let $\alpha \in L^\infty(k)$ and $\beta \in L^\infty(l)$be firing sequences, such that for some $0 \le a \le b \le k-1$ and $0 \le c \le d \le l-1$:*

$$[C_a \oplus C_{a+1} \oplus \cdots \oplus C_b]_\alpha = [C_c \oplus C_{c+1} \oplus \cdots \oplus C_d]_\beta.$$

Let $J = c + (b-a+1) + (l-1-d) = l + (c-d) + (b-a)$. Then there exists a firing sequence $\gamma \in L^\infty(J)$, such that:

$$h(\gamma) = (\underbrace{\langle C_0\rangle_\beta, \cdots, \langle C_{c-1}\rangle_\beta}_{c}, \underbrace{\langle C_a\rangle_\alpha, \cdots, \langle C_b\rangle_\alpha}_{b-a+1}, \underbrace{\langle C_{d+1}\rangle_\beta, \cdots, \langle C_{l-1}\rangle_\beta}_{l-1-d}).$$

Proof. Let R^J be the resulting ring, when $C_a \oplus C_{a+1} \oplus \cdots \oplus C_b$ of R^k and $C_{d+1} \oplus \cdots \oplus C_{l-1} \oplus C_0 \oplus \cdots \oplus C_{c-1}$ of R^l are connected. We can fire the transition instances in α and β in such a way that

1) the right interface transition instances of C_{c-1} (C_b) are fired simultaneously with the left interface transition instances of C_a (C_{d+1}), because $[C_a \oplus C_{a+1} \oplus \cdots \oplus C_b]_\alpha = [C_c \oplus C_{c+1} \oplus \cdots \oplus C_d]_\beta$,

2) the right interface tuples of C_{c-1} (C_b) and the left interface tuples of C_a (C_{d+1}) are changed in a similar way to the tuples in the input places of the interface transitions of C_{a-1} and C_b in α, because the firability vectors of the components are the same in α and β and the components have the same interface transitions,

3) when only the internal transition instances of α or β are fired, the tuples of $C_a \cdots C_b$ change in a similar way to the tuples of $C_{d+1} \cdots C_{c-1}$, because the effect of internal transitions in α on the firability vector of $C_a \cdots C_b$ is the same as the effect of internal transitions in β on the firability vector of $C_{d+1} \cdots C_{c-1}$.

The resulting firing sequence γ is in $L^\infty(J)$ and γ satisfies the condition above.

□

Definition 4.22 *Two rings* $R^k = C_0 \odot C_1 \oplus \cdots \oplus C_{k-1}$ *and* $R^l = C_0 \odot C_1 \oplus \cdots \oplus C_{l-1}$ *are* similar*, denoted by* $R^k \sim R^l$*, if*

1) $\{\langle C_0\rangle_\alpha | \alpha \in L(k)\} = \{\langle C_0\rangle_\beta | \beta \in L(l)\}$ *and*

2) $\{\langle C_i\rangle_\alpha | \alpha \in L(k)\} = \{\langle C_j\rangle_\beta | \beta \in L(l)\}$ *for any* $1 \leq i \leq k-1$ *and* $1 \leq j \leq l-1$.

□

Note that this definition for similarity between processes of Pr/T-nets is totally different from the similarity of processes of fair Petri nets, given in definition 3.10. In definition 4.22 we demand that components of different rings must behave in a similar way in every possible firing sequence, while in definition 3.10 we only demanded that the components are similar in all fair computations. Thus, we require that our Pr/T-net model is fair, in a sense that a component cannot contain transitions, which are not fired in an infinite firing sequence, if some other component can fire its corresponding transitions in the same sequence. For example the net in Figure 5 is not fair, because some node can always send a turn to itself. Other nodes never gets a turn and thus they can never fire the transition `Send`. This implies that different nodes cannot always operate in a similar way. In ring protocols, e.g., in communication protocols, it is reasonable to assume that every component can fire all of its transitions infinitely often in an infinite firing sequence. On the other hand it is not reasonable to construct net models containing transitions that are never fired.

4.3 Induction Theorem

The idea of the induction theorem is to provide a method with which we can show that a ring R^k can be merged to a ring R^{k-1} such that the set of all possible behaviours of the components of R^k is a subset of all possible behaviours of the components of R^{k-1}. In addition, we would like to be able to show that a ring R^{k-1} can be split to a ring R^k such that the set of all possible behaviours of the components of R^{k-1} is a subset of all possible behaviours of the components of R^k. Thus, when all components of the rings R^k and R^{k-1} operate in a similar way (are *uniform*) and R^k is structurally bounded (R^k is structurally bounded, if R^2 is structurally bounded (Li et al. 1994a)), the behavior of R^l is similar to that of R^{k-1} for any $k \geq 5$ and $l \geq k$. To proceed we first need formal definitions for merging and splitting rings of processes of Pr/T-nets and for the concept of uniformity.

Definition 4.23 *For* $2 \leq r \leq k-1$*,* R^k *is* r-mergeable *if*

1) for any $\alpha \in L^\infty(k)$ *there exists* $\beta_0 \in L^\infty(k-1)$ *such that:*

$$h(\beta_0) = (\langle C_0\rangle_\alpha, \underbrace{\langle C_1\rangle_{\beta_0}, \langle C_2\rangle_{\beta_0}, \cdots, \langle C_{r-1}\rangle_{\beta_0}}_{[C_1 \oplus C_2 \oplus \cdots \oplus C_r]_\alpha}, \langle C_{r+1}\rangle_\alpha, \langle C_{r+2}\rangle_\alpha, \cdots, \langle C_{k-1}\rangle_\alpha),$$

and

2) for any $\alpha \in L^\infty(k)$ *there exists* $\beta_1 \in L^\infty(k-1)$*, such that:*

$$h(\beta_1) = (\langle C_0\rangle_\alpha, \langle C_1\rangle_\alpha, \cdots, \langle C_{k-r-1}\rangle_\alpha, \underbrace{\langle C_{k-r}\rangle_{\beta_1}, \langle C_{k-r+1}\rangle_{\beta_1}, \cdots, \langle C_{k-2}\rangle_{\beta_1}}_{[C_{k-r} \oplus C_{k-r+1} \oplus \cdots \oplus C_{k-1}]_\alpha}).$$

□

Definition 4.24 *For* $2 \leq r \leq k$*,* R^k *is* $r-1$-splittable *if, for any* $\alpha \in L^\infty(k)$*, there exists* $\beta_0, \beta_1 \in L^\infty(k+1)$ *such that:*

$$h(\beta_0) = (\langle C_0\rangle_\alpha, \underbrace{\langle C_1\rangle_{\beta_0}, \langle C_2\rangle_{\beta_0}, \cdots, \langle C_r\rangle_{\beta_0}}_{[C_1 \oplus C_2 \oplus \cdots \oplus C_{r-1}]_\alpha}, \langle C_r\rangle_\alpha, \langle C_{r+1}\rangle_\alpha, \cdots, \langle C_{k-1}\rangle_\alpha),$$

and

$$h(\beta_1) = (\langle C_0\rangle_\alpha, \langle C_1\rangle_\alpha, \cdots, \langle C_{k-r}\rangle_\alpha, \underbrace{\langle C_{k-r+1}\rangle_{\beta_1}, \langle C_{k-r+2}\rangle_{\beta_1}, \cdots, \langle C_k\rangle_{\beta_1}}_{[C_{k-r+1} \oplus C_{k-r+2} \oplus \cdots \oplus C_{k-1}]_\alpha}).$$

□

$(r-1)$-splittability assures that we can replace either of the two chains of length $r-1$ adjacent to C_0 in R^{k-1} by a chain of length r to obtain a firing sequence in R^k. $r-$mergeability assures that we can replace either of the two chains of length r adjacent to C_0 in R^k by a chain of length $r-1$ to obtain a firing sequence in R^{k-1}.

Definition 4.25 *A ring R^k is* uniform *if, for any $1 \leq i, j \leq k-1$ and $\alpha \in L^\infty(k)$, there exists $\beta \in L^\infty(k)$ such that $\ll C_i \gg_\alpha = \ll C_j \gg_\beta$.* □

Before we give the induction theorem, which is the main result of this report, we give a trivial example of a Pr/T-net, which is 2-splittable, 3-mergeable and uniform.

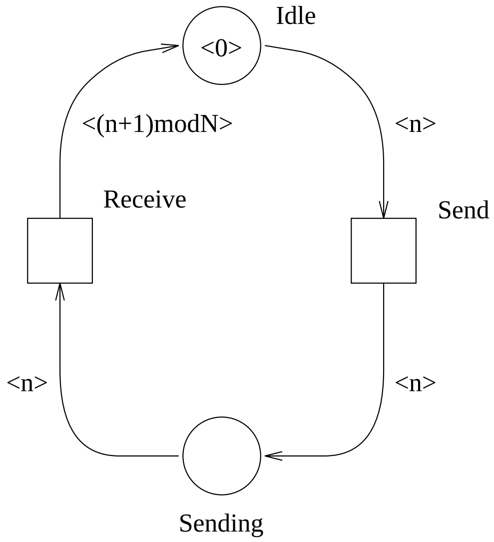


Figure 7: A trivial distributed mutual exclusion.

Example 4.26 Consider a Pr/T-net in Figure 7. The net models a distributed mutual exclusion protocol; only a node, having the token, has access to the critical section (critical section is not modelled). After the critical section, the node forwards the token to the next node. Consider rings R^6 and R^7. The only possible firing sequence of the net is $\alpha = (\texttt{Send Receive})^\omega$, in which `Send` is an internal transition of each component and `Receive` is an interface transition of each component. It is straightforward to see that R^6 and R^7 are uniform, because in the only possible firing sequence $\ll C_i \gg_\alpha = (\text{Receive}^{\langle 0\rangle})^\omega$ for every $1 \leq i \leq 5$ and $1 \leq i \leq 6$, respectively.

The externally visible history of the components $[C_1 \oplus C_2]$ and $[C_4 \oplus C_5]$ of R^6 and $[C_1 \oplus C_2 \oplus C_3]$ and $[C_4 \oplus C_5 \oplus C_6]$ of R^7 in α is

$$\left(\begin{bmatrix}\langle 0\rangle \\ \ \end{bmatrix}\text{Receive}^{\langle 0\rangle}\begin{bmatrix} \ \\ \ \end{bmatrix}\begin{bmatrix} \ \\ \langle 0\rangle\end{bmatrix}\text{Receive}^{\langle 0\rangle}\begin{bmatrix} \ \\ \ \end{bmatrix}\right)^{\omega}.$$

Thus we can exchange $C_1 \oplus C_2$ or $C_4 \oplus C_5$ of R^6 and $C_1 \oplus C_2 \oplus C_3$ or $C_4 \oplus C_5 \oplus C_6$ of R^7 without affecting the other components, as shown in lemma 4.21. This implies that R^6 is 2-splittable and R^7 is 3-mergeable. □

Theorem 4.27 *If R^2 is structurally bounded and for some $2 \leq r < k/2$, R^k is $r-$mergeable, R^{k-1} is $(r-1)-$splittable and R^{k-1} and R^k are both uniform, then $R^{k-1} \sim R^l$ for all $l \geq k$.*

Although we have used the Pr/T-net formalism in the definitions of components, firability vectors and similarity, theorem 4.27 is almost identical to theorem 3.17. However, theorem 4.27 is stronger than theorem 3.17, because in theorem 3.17 we restrict the possible firing sequences of a component to only those, which are fair, while in 4.27 a ring must satisfy the boundedness, splittability, mergeability and uniformity in every possible firing sequence (not only the fair ones). Thus, the only thing we need to show is that theorem 4.27 implies theorem 3.17.

Lemma 4.28 *Let P be a property of a firing sequence and let $\mathcal{L}(k)$ be the set of all fair firing sequences of a ring R^k, let $L^\infty(k)$ be the set of all firing sequences of the ring R^k and let R^k satisfy the conditions of theorem 4.27 for some $k \geq 5$. Then, if the firing sequence L^∞ satisfies P, then $\mathcal{L}(k)$ satisfies P.*

Proof. Assume that the set of fair transitions of a fair Petri net is empty. Then, trivially $\mathcal{L}(k) = L^\infty(k)$, i.e, the set of fair computations is the set of all possible computations. Then, if P is satisfied by all possible firing sequences, it will be satisfied by all fair firing sequences too. □

Note that lemma 4.28 implicitly states that all the transitions of the net must occur fairly in the sense that all transitions that are enabled infinitely often are fired infinitely often. Otherwise, the net model of the ring R^k can contain transitions, which only some components can fire, which implies that the behaviour of the components cannot be similar in all possible firing sequences.

Proof of theorem 4.27. By lemma 4.28 if a ring satisfies theorem 4.27, it will satisfy theorem 3.17. □

The only part of the theorem that can cause problems is the structural boundedness. A Pr/T-net is structurally bounded, if and only if it is possible to assign a positive integer weight to each place so that the weighted sum of tuples of a marking never increases after a firing of any transition and every field of tuples at a marking is bounded. Structural boundedness can be calculated by using linear *place invariants*. In (Genrich 1989) the place invariants are defined for Pr/T-nets. However, to interpret the place invariants it seems necessary to bind the free variables via a substitution, where at least partial knowledge about the transition sequence leading to a marking must be used.

To overcome this problem, *Coloured Petri nets* (Jensen 1987) were defined. Sometimes the easiest way to prove the structural boundedness is just to unfold R^2 and use the place invariants defined for low-level Petri nets given in (Murata 1989) and (Couvrer and Paviot-Adet 1994).

4.4 On the Implementation of the Theorem

It would be tempting to use an automatic verifier to prove that a ring of an arbitrary number of processes satisfies any branching time temporal logic formula. Generally this is impossible for identical finite-state machines (Apt and Kozen 1986) and even for unidirectional rings for n finite-state machines, whose configuration do not depend on n and in which only one state transition can occur at a time (Suzuki 1988). However, it would at least be desirable to determine, if a given Pr/T-net of components were mergeable, splittable or uniform, because this is the most difficult task in the analysis of real protocols.

Below we give an algorithm, which can be used to determine, if our induction theorem is applicable to a ring model of a concurrent system. We start with the reachability graphs of the smallest rings that can satisfy the conditions of theorem 4.27, i.e., R^4 and R^5. We calculate, whether these rings are uniform. If they are not, we increase the size of the rings by one component. If the rings are uniform, we compute, whether they are $r-$mergeable and $r-1-$splittable for some $2 \leq r < k/2$. Note that our algorithm does not halt, if the rings of any size do not satisfy the conditions of the induction theorem.

Algorithm 4.29 A sufficient condition for the induction theorem.

```
k := 5; # initially the size of the ring is 5
u := 1; # a variable, indicating if the rings are uniform
DO WHILE(TRUE)
    generate the reachability graphs of R^{k-1} and R^k;
    FOR (i := 1, i < k, i := i + 1) DO
        FOR (j := 1, j < k, j := j + 1) DO
            IF (∀ infinite firing sequences α of R^{k-1}, ∃β in R^k: ≪C_i≫_α=≪
C_j≫_β)
            DO
            OD
            ELSE u := 0;
        OD
    OD
    IF (u=1) DO
        FOR (r := 2, r < k/2, r := r + 1) DO
            IF (∀ infinite firing sequences α of R^{k-1}, ∃β in R^k: [C_1 ⊕ ··· ⊕
C_{r-1}]_α =
            [C_1 ⊕ ··· ⊕ C_r]_β ∧ [C_{k-r+1} ⊕ ··· C_{k-1}]_α = [C_{k-r} ⊕ ··· ⊕ C_{k-1}]_β) DO
                PRINT (R^{k-1} and R^k are uniform, R^{k-1} is r − 1−splittable
and R^k
                is r−mergeable);
                END;
            OD
        OD
    OD
    k := k + 1;
    u := 1;
OD
```
□

5 Fairness of the Frame Synchronized Ring

The Frame synchronized ring (FSR) (Raatikainen and Zidbeck 1995) is a cost-effective high throughput (beyond 100 Mbps) interconnection network, developed in the Technical Research Centre of Finland during the years 1990-1995. The FSR will be applied to switching applications in *Asynchronous Transfer Mode* switches and in multimedia environments, for example in the distribution of *Video on Demand* services, where high and reliable data rate processing in real time is necessary. That is why the real time and fairness properties of the FSR have been studied extensively recently (Raatikainen, Zidbeck and Flinck 1993).

Due to the distributed *medium access control* (MAC) algorithm, the worst case queuing time of a FSR node, waiting to get an access to the data bus, grows exponentially with respect to the number of nodes connected to the ring. Thus, reachability analysis is not very suitable in the analysis of queuing times of the nodes or the fairness of the FSR (Pyssysalo 1994a), when the number of nodes increases to twenty or fifty. The transfer delay of the FSR is analyzed analytically in (Harju, Raatikainen and Zidbeck 1993). The results are compared to other common ring architectures, which gives implicitly some information about the fairness of the ring. In (Pyssysalo 1994b) the fairness is studied formally using predicate/transition nets and reachability analysis. However, only small rings, consisting of less than six nodes, were successfully analyzed, because of the state space explosion problem. The idea of using an induction theorem in the analysis of the FSR was first presented in (Pyssysalo, Raatikainen and Zidbeck 1995). In this chapter we shall show that the slightly simplified MAC algorithm of the FSR, consisting of any number of nodes, is fair, i.e., every node that is waiting for an access to the data bus, will get it in a finite waiting time. We have had to simplify the algorithm, because of the size of the state space in the reachability analysis. The actual waiting time of the FSR nodes can be determined by using the method for doing a worst case analysis of time critical systems using time Petri nets. The method is given in (Leveson and Stolzy 1987). However, the weakness of this method is that all waiting sequences are considered to have equally high probabilities. In reality the worst case waiting sequence occurs very seldom in the FSR.

5.1 Frame Synchronized Ring

The FSR is a multiprocessor bus concept, based on a ring topology and destination release policy. The FSR consists of a number of nodes, connected together with unidirectional point to point links. Each link contains a parallel data bus and three control lines: a frame synchronization, a bus clock and a request line. The data bus is typically 16, 32 or 64 bits wide, but can be set to any value, depending on the application. The frame synchronization signals indicate the beginning of each frame to the nodes. The bus clock is used to synchronize the data bus and all the control lines to the same clock. The clock frequency is typically $50 - 200 \cdot 10^6$ Hz. The request line carries requests, which the nodes, requesting an access to the data bus, send. All signals, except the request signal, are transferred into the same direction. The nodes contain an interface to the peripheral attached to the node, input and output buffers for the exchange of data between the application and the peripheral and a medium access control algorithm, implemented using Xilinx field programmable gate arrays (Harju et al. 1993). The block diagram of the FSR is presented in Figure 8.

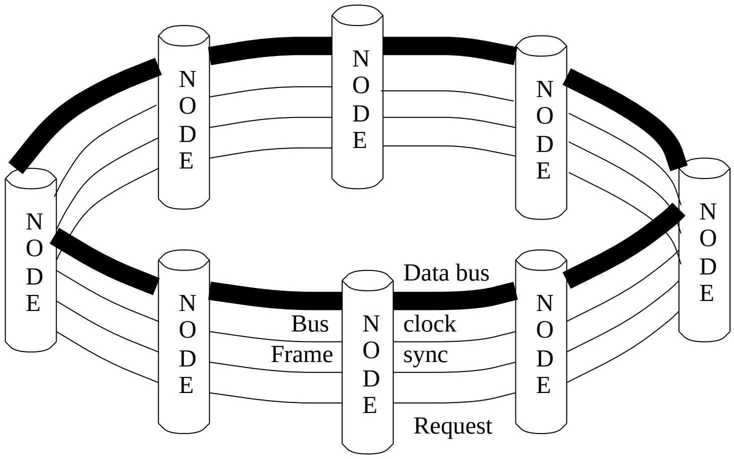


Figure 8: The FSR-bus.

5.1.1 Operation of the FSR

Applications transmit data in packets by setting the packets into the interface buffer of the node. The node monitors the by-passing traffic and control signals and uses a medium access control algorithm to decide, when it can transmit the packet. The node transmits a packet in fixed length frames, consisting of a number of time slots. Each time slot is transmitted from an input of a node to an input of a next node in one bus cycle. The first time slot of each frame is a header, which contains addressing and control information. The subfields of the header indicate, whether the incoming frame is occupied or free, whether it is sent to a group of nodes or to a single node, the destination address of the frame, including the ring address, if the frame is sent from one ring to another, whether the frame is retransmitted, because of a failure of the ring or the destination node and whether the frame is the first or the last frame of a packet. The extension field can contain any application specific control data. The rest of the time slots contain application data. In some applications the last time slot is a trailer, which can contain for example a check sum for the rest of the frame. The structure of the FSR frame and its subfields is illustrated in Figure 9.

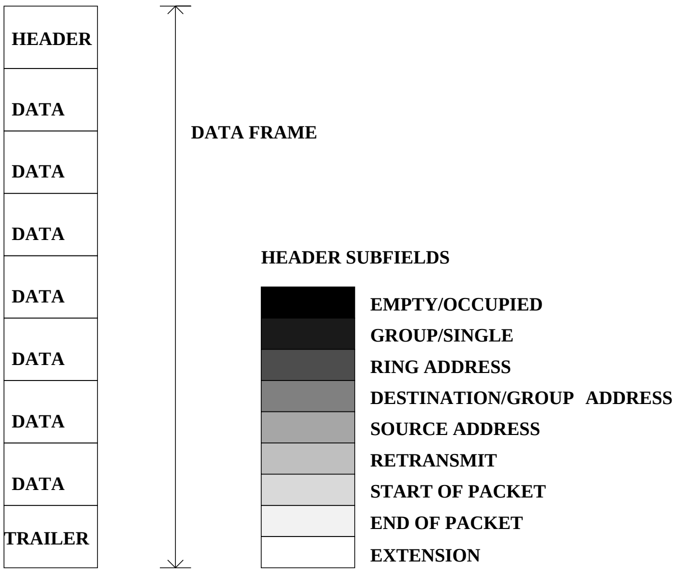


Figure 9: The frame structure of the FSR.

Every node of the FSR executes the MAC algorithm once in every bus clock. The flow diagram of the algorithm is given in Figure 10. Note that initially all variables are inactive or false. After the start-up of the ring, the header starts to circulate automatically and it occurs simultaneously with `frame_sync`.

First the node decides, whether the incoming frame is empty. The incoming frame is empty, if it is not occupied or it is occupied and the destination address of the frame equals to the address of the node. A node becomes ready to send, when the application puts data into the output buffer of the node. A node, which has no data to send, forwards the requests it receives, if the incoming frame is not available. In case the frame is available, the node forwards the requests only, if it has already forwarded a request during the previous bus clock cycle.

A node, having data to send, must get a turn, before it gets an access to the bus. If the incoming frame is occupied, the node can get a turn, if other nodes do not send requests. Otherwise the node must forward requests of other nodes. In case the node has a turn, it can send its own request.

If the incoming frame is available, when a node tries to get an access to the bus, it can immediately start transmitting, if it has a turn and the frame synchronization signal is active, i.e., the beginning of the frame is in the input of the node. If the node does not have a turn, it still can start transmitting, if the frame synchronization signal is active and other nodes are not requesting an access to the bus. In other cases the node gets a turn, if other nodes are not sending requests.

A node that has got the access to the bus blocks incoming requests, until it has sent the whole frame. During each bus clock cycle, the node sends one time slot of a frame. Usually the frame contains as many time slots as the ring contains nodes.

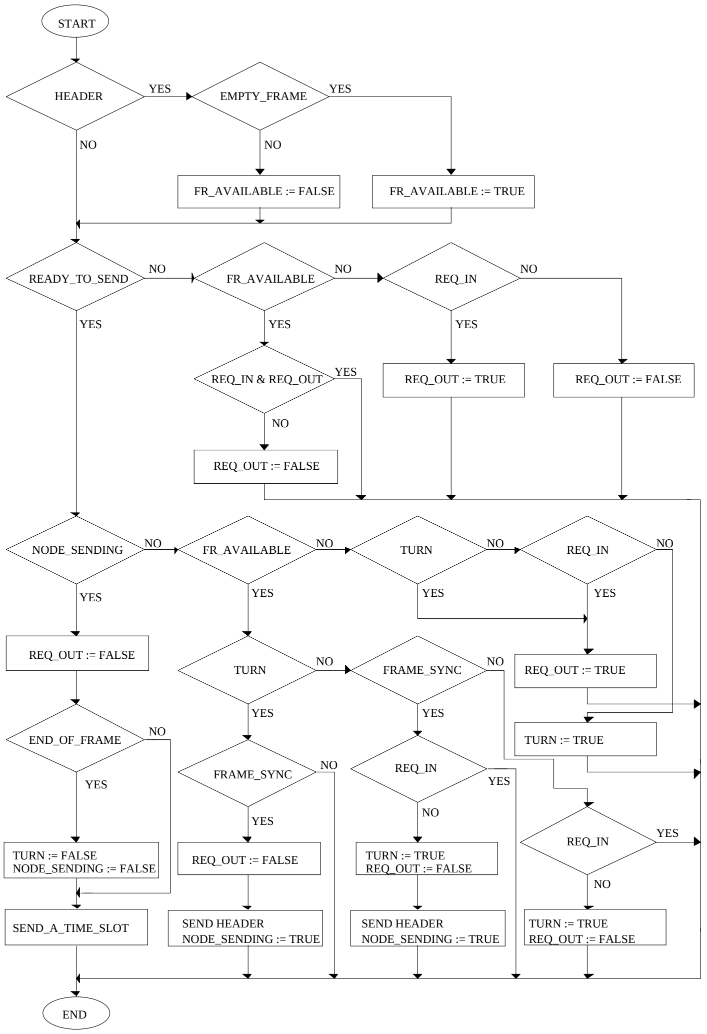


Figure 10: Medium access control algorithm.

5.1.2 Fairness of the FSR

The informal proof of the fairness of the FSR is given in (Raatikainen and Zidbeck 1995). The fairness is proven by showing that the FSR satisfies certain properties that together imply that the FSR is fair. The fairness can be defined in a number of different ways. In (Howell, Rosier and Yen 1991) 24 different versions of fairness are given and in addition their complexity bounds and decidability in general are examined. In addition, it is studied for a certain definitions of fairness, whether there exists an infinite fair computation. A *fair* computation is a computation, which satisfies the fair occurrence rule for a transition, i.e., no transition can be enabled infinitely often without occurring (Valk 1987). The computation, in which from some point of time on a transition cannot be permanently enabled without occurring, is called *just* (Valk 1983). It is not enough that the behaviour of a Petri net, modelling the FSR, is fair or just, because it is still possible that the transition that guarantees a fair access to the ring, is never enabled. Thus, we have to prove that the FSR is fair by proving that the Pr/T-net model of the FSR satisfies a set of properties that together imply the fairness. In (Harju and Raatikainen 1993) an informal proof about the fairness of the FSR is given using the following properties:

Property 5.1 *A node that has issued a service request will be granted to get the access to the bus.*

Property 5.2 *Every node, suspended from transmission, will be granted to issue a service request.*

However, these properties are not reasonable, because a node can get an access to the bus without ever issuing a service request. By examining the MAC algorithm given in Figure 10 it can be seen that before a node can start transmitting, it must have a turn. Note, that it can get the turn and the access to the bus in the same bus clock cycle. Thus, if we want to show that the FSR is fair, i.e., every node, having data to send, will eventually (in a finite time) get an access to the bus, we have to show that the FSR satisfies the following properties:

Property 5.3 *A node, having data to send, will get a turn in a finite waiting time.*

Property 5.4 *A node, having a turn, will get the access to the bus in a finite waiting time.*

5.2 A Pr/T-Net Model of the FSR

In (Pyssysalo et al. 1995) we presented a Pr/T-net model of the FSR, which had only one transition. Each firing of this transition had the same effect on the marking of the net as one bus cycle had on the state of the FSR. Although this kind of the net model had a small reachability graph, only rings, containing less than six nodes, were successfully analyzed using `PROD`. The structure of the net causes more serious problems with respect to the ease of the application of the induction theorem. Because only one transition reads and updates all tuple tokens of the net in every transition instance, the size of the firability vectors depend on the size of the ring, i.e, the number of nodes. Thus, it is impossible to apply the induction theorem to the analysis of this kind of the net model. On the other hand, the size of the reachability graph grows even faster, when one bus clock cycle is modelled with more than one transition.

We have two principles, in the modelling of the FSR using Pr/T-nets. We try to construct our Pr/T-net model in such a way that it is as easy as possible to apply induction theorem 4.27. To keep the size of the reachability graph as small as possible is our second principle. These principles lead to a Pr/T-net net model, presented in Figure 11. The net has two transitions, named `BUS CLOCK` and `MAC`. The former fires as many times as there are nodes in the ring (N). It moves the header tuple from the place indicating the node i to the place indicating the node $(i \mathrm{mod} N) + 1$ and the request tuple from the place indicating the node $(i \mathrm{mod} N) + 1$ to the place indicating the node i. The latter transition updates the variable, request, header and destination tuples of the places. The transition fires only once after N firings of the `BUS CLOCK` transition. This kind of a synchronization, where a set of independently executing processes are synchronized in such a way that every process executes the same operation simultaneously and then continues independent execution again, is called a barrier synchronization (Andrews 1991). The effect of the firing of the `MAC` transition after the synchronization is the same as the effect of the execution of the MAC algorithm in every node of the ring. The arc expressions and the firing conditions of the transitions are explained in the `PROD` net description, given in the appendix.

The structure of the Pr/T-net model of the FSR in Figure 11 is such that the induction theorem can be applied easily, but the number of reachable states is N times larger than the number of reachability states of the net, represented in (Pyssysalo et al. 1995). Thus, using `PROD` we cannot analyze rings, consisting more than four nodes without simplifying the model and restricting the possible transition sequences of the net. In the analysis we have assumed that every node is always ready to send, which means that every node always has data to send. This assumption makes the queuing sequences in the FSR ring considerably shorter and we can analyze the Pr/T-net, consisting of eight nodes, which is necessary, before we can apply the induction theorem. R^4 is not 1-splittable, because in a Pr/T-net component, modelling one node, the firability vector is changed by the interface transitions only, while in a

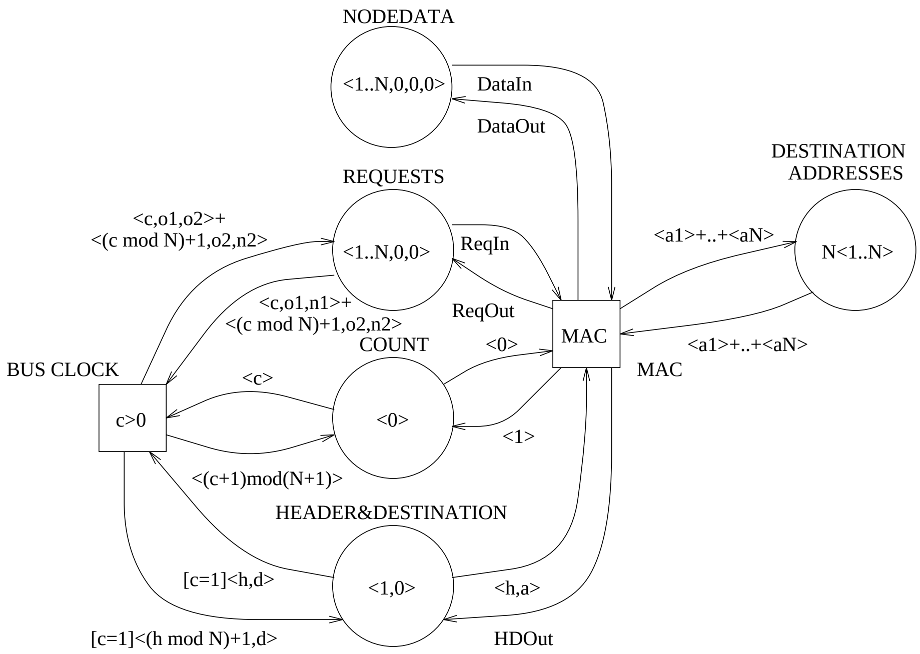


Figure 11: Pr/T-net model of the FSR.

Pr/T-net component, modelling two nodes, the firability vector is changed by one internal transition in addition to the interface transitions. In addition, R^6 is not 2-splittable, because the changes in right interface tuples affect the left interface tuples in R^6, while in R^7 a third component eliminates the visible effects between the right and left interface tuples. This means that the behaviour of a two node component cannot be a subset of a three node component in R^6 and R^7, respectively. Thus, we have to analyze R^8 and R^9 and show that R^8 is 3-splittable and R^9 is 4-mergeable.

Note that we have identified the components in Figure 11 from 1 to N, because we need the identifier 0 for triggering the bus clock. All the components have an own tuple in places `NODEDATA` and `REQUESTS` at all reachable markings, which can be easily verified with `PROD`, by searching for all such markings, which do not contain N different tuples in these places. All transition instances of `MAC` have an identity i, where $1 \leq i \leq N$. However, the transition is an internal transition, because it changes the tuples of each component independently, although every tuple changes simultaneously. Thus, `BUS CLOCK` is the only interface transition of the net.

The input tuple of the transition instance i of the transition `BUS CLOCK` is $\langle i \rangle$, taken from the place `COUNT`. This determines the other input tuples of the transition instance taken from other places without confusion. We have omitted the count, header, and destination tuples from the firability vectors of the rings, because these always map to the same individual and thus do not affect the behaviour of the component. Thus, the only interface place is `REQUESTS` and the firability vectors $C_a \oplus \cdots \oplus C_b$ of the rings, consisting at least eight nodes, have the form $[w\ x\ y\ z\ ]^T$, where w is the right interface tuple of the component C_{a-1} in the place `REQUESTS`, x is the left interface tuple of the component C_a, y is the right interface tuple of C_b and z is the left interface tuple of C_{b+1}.

5.3 Correctness of the FSR

We now apply theorem 4.27 and show that the FSR ring of any size is fair. We shall first show that the FSR ring of eight nodes satisfies properties 5.3 and 5.4, which implies that R^8 is fair. Next we show that R^2 structurally bounded, i.e., its reachability graph has only a finite number of nodes with an arbitrary finite initial marking. Then we proceed by showing that R^8 and R^9 are uniform, R^8 is 3-splittable and R^9 is 4-mergeable. We need an additional assumption to be able to prove the splittability and mergeability, because the length of the transition sequences varies in rings of different sizes. We prove the fairness of R^k $(k \leq 7)$ using `PROD` and finally conclude that the FSR ring of an arbitrary size is fair in the sense that it satisfies properties 5.3 and 5.4.

5.3.1 Fairness of the FSR of Eight Nodes

The FSR ring of eight nodes satisfies properties 5.3 and 5.4, if no transition sequence in the reachability graph of a Pr/T-net in Figure 11 contains a loop, in which any of the eight nodes does not get an access to the data bus.

Lemma 5.5 *Any transition sequence in the FSR ring of eight nodes satisfies property 5.3.*

Proof. We prove this lemma using `PROD`. Because in our simplified model of the FSR given in Figure 11 every node is always ready to send, i.e., has data to send, the property is proven by showing that the reachability graph of the net model of the FSR does not contain any loops, i.e., infinite transition sequences, in which a node does not get a turn. Thus, every node must get a turn in a finite waiting time. The property is shown for all eight nodes in the ring R^8 by the following `PROD` macro

```
define property3(n) query node not(not(dspan((((NODEDATA) :
(field[0] == (n) && field[2] == 0)) != empty), true) false))
```

Because no loops can be found in the reachability graph of the ring R^8, violating property 5.3, R^8 satisfies property 5.3. $\square$

Lemma 5.6 *Any transition sequence in the FSR ring of eight nodes satisfies property 5.4.*

Proof. The proof is similar to the proof of lemma 5.5. We prove with the following PROD macro that the reachability graph of the FSR ring R^8 contains no infinite transition sequence, in which a node, having a turn, will not eventually get an access to the bus.

```
define property4(n) query node not(not((((NODEDATA) :
(field[0] == (n) && field[2] == 0)) != empty) and dspan
((((NODEDATA) : (field[0] == (n) && field[3] == 0)) != empty),
true) false))
```

Because no loops can be found in the reachability graph of the ring R^8, violating property 5.4, R^8 satisfies property 5.4. □

Theorem 5.7 *Any transition sequence in the FSR ring, consisting of eight nodes, satisfies properties 5.3 and 5.4, i.e., every node, having data to send, will get an access to the bus in a finite time.*

Proof. The theorem follows from lemmas 5.5 and 5.6. □

Now we have proven that R^8 satisfies properties 5.3 and 5.4. We proceed by showing that the induction theorem can be applied to the FSR ring containing at least eight nodes.

It is shown in (Li et al. 1994a) that a Petri net, consisting of any number of components is structurally bounded, if the net, consisting of two components is structurally bounded. It is necessary to know that our reachability graph always has a finite number of distinct markings, implying that all infinite transition sequences must be loops. The induction theorem is not applicable, if the graph contains infinitely many distinct markings, because we cannot conclude anything from the infinite transition sequences, consisting of an infinite number of distinct transition instances.

Lemma 5.8 *The FSR ring R^2 is structurally bounded.*

Proof. A Petri net is structurally bounded if and only if it is possible to assign a positive integer weight to each place in such a way that the weighted sum of token counts of a marking never increases after firing of any transition (Murata 1989). We show with PROD that the number of tuples is constant in every reachable marking of the net R^2, i.e., the number of tuples never increases after the firing of any transition of the net. The following PROD macro returns every marking, in which the cardinality of the places is not constant

```
define constant(N) query node ((card(NODEDATA) != (N)) &&
(card(REQUESTS) != (N)) && (card(HEADERandDEST) != 1) &&
(card(ADDRESSES) != (N*N)) && (card(COUNT) != 1))
```

`PROD` returns an empty set to a query, thus the number of tuples in R^2 does not increase after the firing of any transition. In addition to the number of tuples, we have to show that no field of the tuples increases without upper bound. The lower and upper bounds of the fields of the tuples are explicitly given in the `PROD` net description, as seen in the appendix. A transition instance is not enabled, if these bounds are not satisfied. Thus, the unfolded Pr/T-net has only a finite number of places and transitions, because the Pr/T-net contains a finite number of tuples in every reachable marking and the unfolded Pr/T-net has only a finite number of distinct markings, because the fields of the tuples of a Pr/T-net have only a finite number of distinct values for individuals. $\Box$

5.3.2 Uniformity, Splittability and Mergeability of the FSR of Eight Nodes

In order to apply the induction theorem we must first show that the FSR of eight nodes is uniform and 2- or 3-splittable and the FSR of nine nodes is uniform and 3- or 4-mergeable.

Lemma 5.9 *The FSR ring, consisting of eight or nine nodes, is uniform.*

Proof. We only show that R^8 is uniform. The proof for R^9 is similar and thus omitted. The only possible transition sequence in R^8 is $\alpha =$ (`MAC (BUS CLOCK)`$^8)^\omega$. Now consider a component $i \geq 2$. It has one interface transition `BUS CLOCK`, two left interface tuples $i-1$ and i and two right interface tuples i and $(i \text{mod} 8)+1$. Thus, in the only possible firing sequence the externally visible history of the component i, where $2 \leq i \leq 8$ in R^8 is of the form

$$\begin{bmatrix} \langle 1,0,0\rangle \\ \langle 1,0,0\rangle \\ \langle 1,0,0\rangle \end{bmatrix} \left(\begin{bmatrix} \langle 1,a,a\rangle \\ \langle 1,b,b\rangle \\ \langle 1,c,c\rangle \end{bmatrix} \text{BUS CLOCK}^{\langle 1,a,a\rangle+\langle 1,b,b\rangle} \begin{bmatrix} \langle 1,a,b\rangle \\ \langle 1,b,b\rangle \\ \langle 1,c,c\rangle \end{bmatrix} \right.$$
$$\left. \text{BUS CLOCK}^{\langle 1,a,b\rangle+\langle 1,b,b\rangle} \begin{bmatrix} \langle 1,a,b\rangle \\ \langle 1,b,c\rangle \\ \langle 1,c,c\rangle \end{bmatrix} \begin{bmatrix} \langle 1,a,b\rangle \\ \langle 1,b,c\rangle \\ \langle 1,c,d\rangle \end{bmatrix} \right)^\omega$$

Assume that initially none of the nodes $i-1$, i and $(i \text{mod} 8)+1$ is sending. Because we assumed that every node is always ready to send, $a=b=c=1$, because by lemma 5.5 every node gets a turn in R^8 and a node, having a turn, issues requests, if it cannot get an access to the bus. Now consider all the transition sequences in R^8, in which node 2 starts transmitting a frame. By theorem 5.7 this will occur in all possible transition sequences. For all of these sequences there is a transition sequence, in which a node i ($i \geq 3$) will get an access to the bus by lemma 5.6. After sending a frame, a transmission turn will be transferred to the next node in all transition sequences, because otherwise a node never gets an access to the bus, which violates theorem 5.7. By lemma 5.5 a node will eventually get a turn in all transition sequences and it starts issuing requests again. □

Note that the size of the ring affects the number of times the transition `BUS CLOCK` is fired. Thus, no ring of size N cannot be split to a ring of size $N+1$, because in cases, where we send frames to or over an extra node in R^{N+1}, the externally visible history of a component in R^{N+1} contains one extra sequence of two interface transitions `BUS CLOCK`. For the same reason no ring of size $N+1$ is mergeable to a ring of size N. However, by examing the externally visible histories in rings of different size, we can see that this extra sequence is always similar to the next sequence, consisting of two interface transitions. Thus, because this extra sequence has no other effect on the externally visible history of the component except the increase of the length of the history, we merge all similar adjacent sequences $[A]\,\mathrm{BUS\ CLOCK}\,[B]\,[C]\,\mathrm{BUS\ CLOCK}\,[D]\,[E]$ to one sequence (A, B, C, D and E are firability vectors). Because we merge only similar adjacent sequences and by lemma 5.8 all infinite transition sequences in R^k with an arbitrary k contain loops, the merging will not violate the fairness property, i.e., properties which claim that something will eventually happen. The only effect of the merging is that the waiting times in externally visible histories are shorter than in the real FSR ring. Thus, we cannot conclude anything about the waiting times, except that the time increases, when the number of nodes in the ring increases.

Lemma 5.10 *The FSR ring, consisting of eight nodes, is 3-splittable, if similar adjacent transition sequences, containing two similar instances of the transition* `BUS CLOCK`*, are merged to one sequence.*

Proof. We give the outline of the proof only, because the presentation of externally visible histories would require too much space. The transition sequences of R^8 have the same effect on the externally visible history of the component $C_2 \oplus C_3 \oplus C_4$ ($C_6 \oplus C_7 \oplus C_8$) as the transition sequences of R^9 have on the externally visible history of the component $C_2 \oplus C_3 \oplus C_4 \oplus C_5$ ($C_6 \oplus C_7 \oplus C_8 \oplus C_9$), when the behaviour of the components C_1, C_2, C_3, C_4, C_5, C_6, C_7 and C_8 of R^8 is replaced by the behaviour of the components C_1, C_2, C_3, C_5, C_6, C_7, C_8 and C_9 (C_1, C_2, C_3, C_4, C_5, C_6, C_7 and C_9) of R^9. □

Lemma 5.11 *The FSR ring, consisting of nine nodes, is 4-mergeable, if similar adjacent transition sequences, containing two similar instances of the transition* `BUS CLOCK`*, are merged to one sequence.*

Proof. As in lemma 5.10 we give the outline of the proof only. Consider the components $A = C_2 \oplus C_3 \oplus C_4 \oplus C_5$ of R^9 and $B = C_2 \oplus C_3 \oplus C_4$ of R^8. The behaviour of the components $C_6 \oplus C_7 \oplus C_8 \oplus C_9$ of R^9 and $C_6 \oplus C_7 \oplus C_8$ of R^8 is similar to A and B, respectively and is thus omitted.

The transition sequences of R^9 have the same effect on the externally visible history of the component A as the transition sequences of R^8 have on the externally visible history of the component B when the behaviour of the components C_1, C_2, C_3, C_5, C_6, C_7, C_8 and C_9 of R^9 is replaced by the behaviour of the components C_1, C_2, C_3, C_4, C_5, C_6, C_7 and C_8 of R^9. Thus, we need to prove only that there is a transition sequence in R^8, in which the externally visible history of B is the same as in the transition sequences of R^9, in which C_4 is involved. There are three cases:

1) Node 4 waits to get an access to the bus. This behaviour does not affect the externally visible history of A. Thus, there is a transition sequence in $L^\infty(8)$, in which B has a similar history.

2) Node 4 gets an access to the bus and sends a frame to itself. This behaviour does not affect the externally visible history of A either. Thus, there is a transition sequence in $L^\infty(8)$, in which B has a similar history.

3) Node 4 gets an access to the bus and sends a frame to a node i, where $i \neq 4$. This behaviour has the same effect on the externally visible history of A as has the transition sequence in $L^\infty(8)$, where a node of R^8, corresponding the FSR node, which had an access to the bus just before node 4 in R^9, directly sends a frame to the destination of node 4.

□

Theorem 5.12 *Any node, having data to send, will get an access to the bus in a finite time in the FSR ring R^k with $k \geq 8$.*

Proof. The theorem follows from theorems 4.27 and 5.7 and lemmas 5.8, 5.9, 5.10 and 5.11. □

Corollary 5.13 *Any node, having data to send, will get an access to the bus in a finite time in the FSR ring R^k with $k \geq 2$.*

Proof. We have shown the fairness of the FSR ring using `PROD`, when the ring contains less than eight nodes. The proof is similar to the proof of theorem 5.7. Thus, by theorem 5.12 any node, having data to send, will get an access to the bus in a finite time in the FSR ring R^k of size $k \geq 2$. □

5.4 On the Applicability of the Theorem

The verification of the fairness of the FSR of an arbitrary number of nodes has been impossible using the induction methods proposed earlier. This is due to the fact that the nodes of the ring are identified and the nodes use these identities to communicate, i.e., to send data packets to each other. In addition, the modelling of the ring using ordinary Petri nets or other low-level modelling formalism would be much more difficult than using predicate/transition nets. However, as we have seen in this chapter, the application of our induction theorem to the verification of real ring protocols, is by no means a straightforward task. One problem is that it is very time-consuming to prove the preconditions of the induction theorem, because there is no way to implement a theorem prover, with which these conditions could be verified fully automatically. The usage of Pr/T-nets in the modelling of the real system helps us a little, because it is usually easy to modify the Pr/T-net model in such a way that the application of the theorem is easy, compared to other induction theorems. The other problem is the state space explosion. It is difficult to examine temporal properties of a system, consisting more than ten identical processes, which is often necessary, before we can find an invariant or a fixpoint. The state space of such a system can easily contain more than 50 million states, the analysis of which is impossible or at least very time-consuming using the reachability analysis.

In the preconditions of the theorem, it is assumed that the components behave in a similar way in all possible transition sequences in rings of different sizes. This is not usually the case, because the behaviour changes, i.e., the length of the transition sequences of the components grow, when the number of nodes increases in the ring. Thus, in the general case we cannot analyze properties, depending on the size of the ring. However, this cannot be considered as a real restriction, because the verification of such rings is not possible using other induction methods either.

In spite of the high amount of the computation and work, needed during the application of our theorem, the method is still a very powerful analysis method. Once we have shown that a ring of a small number of processes satisfies some temporal logic formula and the preconditions of the induction theorem, we can easily verify, whether the ring of an arbitrary number of processes, modelled using predicate/transition nets, satisfies the same formula.

6 Conclusions

The problem of verifying data communication systems, consisting of many identical finite-state processes, is very important and relevant in the modern computer science. In the future computers all over the world will be connected together into one global data communication network. In this network computers must use the same protocol, i.e., communicating rules, in order to convey information between each other. Thus, the protocol designer must be able to verify communication systems, consisting of an arbitrary number of copies of the same protocol. The basic verification problem is to show, whether a system $S(n)$ of n processes satisfies a property $P(n)$ for any value of n, if a system of some specific number of processes satisfies the property. This problem has been proven to be undecidable in general, but a lot of induction theorems still exist applicable to many interesting concurrent systems, like processes of a network with an unspecified number of hosts or a queue with an unspecified number of buffers.

Most induction methods, published so far, are applicable to system models, the processes of which are described using CCS or CSP. Almost all of these methods are based on the same idea of an explicit invariant process. The invariant process is such that its composition with other system processes does not change the behaviour of the system. From some specific number of processes on the behaviour of the system remains invariant or forms a fixpoint not depending on the number of processes used in the composition. The problem of these induction methods is that it is not known, how the invariant process can be obtained in general. In many cases it is a composition of some system processes, but it many cases the invariant must be guessed. In some methods it is assumed that the invariant process is always a composition of some small number of system processes. This restricts the applicability of these methods, because it is possible that an invariant process still exists, although it cannot be obtained using the composition of a few processes.

In this report we present an induction theorem for ring protocols of identical finite processes, described using predicate/transition nets. We assume that the invariant process is a composition of a small number of system processes. This restricts the applicability of our theorem, but the application of the theorem would not be any easier, if the invariant had to be guessed. We hope that using the transformations of Pr/T-nets a system could be modelled in such a way that the invariant process is obtained easily using the composition, if such an invariant exists. Another assumption we have made is that the Pr/T-net model of a system must be fair in the sense that every transition enabled infinitely often must be fired infinitely often. If the net model is not fair, we cannot be sure that in all possible firing sequences all the processes behave in a similar way.

The most feasible property of our method is that it can be used in the verification of ring systems, in which processes use identities and addresses in communication. Such systems cannot be analyzed using any of the methods, published in the literature, because in these methods the behaviour of processes having different identities is considered dissimilar. This is a serious restriction to the applicability of these induction theorems, because in many systems, e.g., in almost all communication protocols, identifications must be used to make different processes distinguishable. We use symmetries of the reachability graphs, describing the behaviour of different processes, to examine the similarity of processes. It turns out that the behaviour of the processes is symmetrical in the sense that they have the same set of possible firing sequences, in which only the address information is different. Using the symmetries we can omit the effect of addresses on the firing sequences and in this way the behaviour of the processes can be considered similar. As an example of the application of the theorem we show the fairness of the arbitration algorithm of the Frame Synchronized Ring, consisting of any number of nodes. The verification of this algorithm is not possible using the methods, presented earlier, because of the address information used by the nodes of the ring.

It must be stressed out that the usage of our theorem is by no means a straightforward task. Further study is still needed to make the application of the theorem easier and faster. Especially it must be studied, how the preconditions of the theorem could be computed as automatically as possible using effective reachability analysis tools. Unfortunately, the reachability analysis causes another problem. The verification of real-world protocols often requires the generation of global state reachability graphs of systems, containing ten or more processes, before the fixpoint is found. The state space of systems of this size can contain more than one billion states, which makes the usage of most reachability analysis tools unfeasible.

Yet another restriction of our method is that it can be applied to systems, having a finite state space only. However, this cannot be considered as a real restriction, because none of the methods published in the literature can be used in the verification of infinite systems. Further study is still needed to examine the possibilities to analyze the properties of infinite systems using induction theorems and reachability analysis tools.

Acknowledgements

This report is the result of my research work in the Digital Systems Laboratory at Helsinki University of Technology in 1995. So first of all I would like to thank professor Leo Ojala for giving me the opportunity to carry out this work. I would also like to thank all my colleagues in the Digital Systems Laboratory for many fruitful discussions and valuable comments.

But most deeply I am indebted to my girl friend Sari, who have lived in Oulu during my working period in Espoo. I thank you very much for your support and patience and especially for your faithful love.

The financial support received from "Jenny ja Antti Wihurin rahasto" and "Suomen Kulttuurirahasto" is also gratefully acknowledged.

References

Andrews, G. R. (1991). *Concurrent Programming, Principles and Practice*, The Benjamin/Cummings Publishing Company Inc., Redwood City, California.

Apt, K. and Kozen, D. (1986). Limits for Automatic Verification of Finite-State Concurrent Systems, *Information Processing Letters* **30**(6): 307–309.

Brookes, S., Hoare, C. and Roscoe, A. (1984). A Theory of Communicating Sequential Processes, *Journal of the Association for Computing Machinery* **31**(3): 560–599.

Cai, Y., Sekiguchi, T., Tanaka, H., Hikichi, M. and Maruyama, Y. (1993). A Method for Structural Analysis of Petri Net Models, *Proceedings of the IEEE International Conference on Industrial Electronics, Control and Instrumentation (IECON), Maui, Hawaii, November 15–19, 1993*, IEEE, New York, pp. 133–137.

Clarke, E., Emerson, E. and Sistla, A. (1986a). Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications, *ACM Transactions on Programming Languages and Systems* **8**(2): 244–263.

Clarke, E. M., Grümberg, O. and Browne, M. C. (1986b). Reasoning about Networks with Many Identical Finite-State Processes, *Proceedings of the 5th Annual ACM Symposium on Principles of Distributed Computing, Calgary, Canada, August 11–13, 1986*, Association for Computing Machinery, Baltimore MD, pp. 240–248.

Couvrer, J. M. and Paviot-Adet, E. (1994). New Structural Invariants for Petri Nets Analysis, *in* R. Valette (ed.), *15th International Conference on Application and Theory of Petri Nets, Zaragoza, Spain, June 20–24, 1994*, number 815 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 199–218.

Darondeau, P. and Gamatie, B. (1990). Infinitary Behaviours and Infinitary Observations, *Fundamenta Informaticae* **XIII**(3): 353–386.

de Nicola, R. and Hennessy, M. C. B. (1983). Testing Equivalences for Processes, *in* J. Diaz (ed.), *Automata, Languages and Programming, 10th Colloquium, Barcelona, Spain, July 18–22, 1983*, number 154 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

Genrich, H. J. (1987). Predicate/Transition Nets, *in* W. Brauer, W. Reisig and G. Rozenberg (eds), *Petri nets: Central Models and their Properties,* *Advances in Petri Nets 1986, Part I. Proceedings of an Advanced Course, Bad Honnef, September 8–19, 1986*, number 254 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 207–247.

Genrich, H. J. (1989). Equivalence Transformations of PrT-Nets, *in* G. Rozenberg (ed.), *Advances in Petri Nets 1988*, number 424 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 179–208.

German, S. and Sistla, A. (1992). Reasoning about Systems with Many Processes, *Journal of the Association for Computing Machinery* **39**(3): 675–735.

Goltz, U. and Mycroft, A. (1984). On the Relationship of CCS and Petri Nets, *in* J. Paredaens (ed.), *Automata, Languages and Programming, 11th Colloquium, Antwerp, Belgium, July 16–20, 1984*, number 172 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 196–208.

Grönberg, P., Tiusanen, M. and Varpaaniemi, K. (1993). PROD—A Pr/T-Net Reachability Analysis Tool, *Technical Report B11*, Digital Systems Laboratory, Helsinki University of Technology.

Harju, J. and Raatikainen, P. (1993). Performance of the FSR-Bus, *Proceedings of the Second Summer School on Telecommunications, Lappeenranta, Finland, August 9, 1993*, Lappeenranta University of Technology, Lappeenranta, pp. 66–73.

Harju, J., Raatikainen, P. and Zidbeck, J. (1993). Analysis of a Link Layer Protocol for High Speed Data Bus, *IEEE 18th Conference on Local Computer Network, Minneapolis, USA, September 19-22, 1993*, IEEE Computer Society Press, New York, pp. 432–439.

Howell, R. R., Rosier, L. E. and Yen, H. (1991). A Taxonomy of Fairness and Temporal Logic Problems for Petri Nets, *Theoretical Computer Science* **82**: 341–372.

Jensen, K. (1987). Coloured Petri Nets, *in* W. Brauer, W. Reisig and G. Rozenberg (eds), *Petri Nets: Central Models and their Properties, Advances in Petri Nets 1986, Part I. Proceedings of an Advanced Course, Bad Honnef, September 8–19, 1986*, number 254 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 248–299.

Kemppainen, J. (1994). *ARA Tools 2.0—A Set of Tools for Analysing the Behaviour of Concurrent Systems, User's Manual*, AATOS Research Group, Technical Research Centre of Finland, Oulu.

Kurshan, R. P. and McMillian, K. (1989). A Structural Induction Theorem for Processes, *Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14–16, 1989*, ACM Press, Baltimore MD, pp. 239–247.

Leveson, N. G. and Stolzy, J. (1987). Safety Analysis Using Petri Nets, *IEEE Transactions on Software Engineering* **SE-13**(3): 386–397.

Li, J., Suzuki, I. and Yamashita, M. (1994a). Fair Petri Nets and Structural Induction for Rings of Processes, *Theoretical Computer Science* **135**: 377–404.

Li, J., Suzuki, I. and Yamashita, M. (1994b). A New Structural Induction Theorem for Rings of Temporal Petri Nets, *IEEE Transactions on Software Engineering* **20**: 115–126.

Manna, Z. and Pnueli, A. (1991). *The Temporal Logic of Reactive and Concurrent Systems*, Springer-Verlag, New York.

Martin, A. J. (1985). Distributed Mutual Exclusion on a Ring of Processes, *Science of Computer Programming* **5**: 265–276.

Milner, R. (1989). *Communication and Concurrency*, Prentice Hall, London.

Montanari, U. and Yankelevich, D. (1994). Combining CCS and Petri Nets via Structural Axioms, *Fundamenta Informaticae* **20**(1,2,3): 193–229.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE* **77**: 541–580.

Nielsen, M. (1987). CCS—and its Relation to Net Theory, *in* W. Brauer, W. Reisig and G. Rozenberg (eds), *Petri Nets: Central Models and their Properties, Advances in Petri Nets 1986, Part II. Proceedings of an Advanced Course, Bad Honnef, September 8–19, 1986*, number 255 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

Pyssysalo, T. (1994a). The Modelling and Analysis of a High Speed Data Bus with Predicate/Transition Nets, *Research Report A29*, Digital Systems Laboratory, Helsinki University of Technology.

Pyssysalo, T. (1994b). Proving Properties of a New High Speed Data Bus with Predicate/Transition Nets, *Microprocessing and Microprogramming* **40**: 791–794.

Pyssysalo, T., Raatikainen, P. and Zidbeck, J. (1995). FSR—A Fair Switching Architecture, *Seventh Euromicro Workshop on Real-Time Systems, Odense Denmark June 14–16, 1995*, IEEE Computer Society Press, Los Alamitos, pp. 116–123.

Raatikainen, P. and Zidbeck, J. (1995). Frame Synchronized Ring and Its Implementation, *VTT Publications 217*, Technical Research Centre of Finland.

Raatikainen, P., Zidbeck, J. and Flinck, H. (1993). Frame Synchronized Ring: A Bus Concept for High Speed Processing Applications, *Integrated Broadband Communication Networks and Services (IBCN&S), Copenhagen, Denmark, April 19–23, 1993*, Elsevier Science Publicers, Copenhagen, pp. 12.1.1–12.1.12.

Reisig, W. (1985). *Petri nets: An Introduction*, Springer-Verlag, Berlin.

Schreiber, G. (1995). Functional Equivalences of Petri Nets, *in* G. D. Michelis and M. Diaz (eds), *Proceedings of the 16th International Conference on Application and Theory of Petri Nets, Turin, Italy, June 26–30, 1995*, number 935 in *Lecture Notes in Computer Science*, Springer-Verlag, Germany, pp. 432–450.

Sistla, A. P. and German, S. M. (1987). Reasoning with Many Processes, *Proceedings of the 2nd IEEE Symposium on Logic in Computer Science, Ithara, New York, June 22–25, 1987*, IEEE, Washington D.C., pp. 138–152.

Suzuki, I. (1985). Fundamental Properties and Applications of Temporal Petri Nets, *Proceedings of the 9th Annual Conference on Information Sciences and Systems*, John Hopkings University, Baltimore MD, pp. 641–646.

Suzuki, I. (1988). Proving Properties of a Ring of Finite-State Machines, *Information Processing Letters* **28**(4): 213–214.

Valk, R. (1983). Infinite Behaviour of Petri Nets, *Theoretical Computer Science* **25**: 311–341.

Valk, R. (1987). Infinite Behaviour and Fairness, *in* W. Brauer, W. Reisig and G. Rozenberg (eds), *Petri Nets: Central Models and their Properties, Advances in Petri Nets 1986, Part I. Proceedings of an Advanced Course, Bad Honnef, September 8–19, 1986*, number 254 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 377–396.

Valmari, A. (1988). Error Detection by Reduced Reachability Graph Generation, *Proceedings of the 9th European Workshop on Application and Theory of Petri Nets*, IEEE, Venice, pp. 95–112.

Valmari, A. and Tienari, M. (1991). An Improved Failures Equivalence for Finite-State Systems with a Reduction Algorithm, *in* B. Jonson, J. Parrow and B. Pehrson (eds), *11 th IFIP WG6.1 Symposium on Protocol Specification, Testing and Verification, Stockholm, Sweden, June 17-20, 1991*, North-Holland, Amsterdam, pp. 3–18.

Varpaaniemi, K., Halme, J., Hiekkanen, K. and Pyssysalo, T. (1995). PROD Reference Manual, *Technical Report B13*, Digital Systems Laboratory, Helsinki University of Technology.

Wolper, P. and Lovinfosse, V. (1989). Verifying Properties of Large Sets of Processes with Network Invariants, *in* J. Sifakis (ed.), *Automatic Verification Methods for Finite-State Systems*, number 407 in *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 68–80.

Wolper, P., Vardi, M. Y. and Sistla, A. P. (1983). Reasoning about Infinite Computation Paths, *Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science, Tuscan, Arizona, November 7–9, 1983*, IEEE, New York, pp. 185–194.

Wu, Z. and Murata, T. (1983). A Petri Net Model of a Starvation-Free Solution to the Dining Philosophers' Problem, *IEEE Workshop on Languages for Automation, Chigago, November 7–9, 1983*, IEEE, Chigago, pp. 192–195.

Appendix: Pr/T-Net Description of the FSR Ring

```
/*

   The PROD net description of the FSR-ring.

   Tino Pyssysalo
   7/12/1995

*/

#define N 2                /* Number of nodes */

/* Macros corresponding the decision diamonds in the MAC algorithm */

/* A frame is address to the node, if the destination address of the
frame is the address of the node */
#define AddressedToNode(n,add) (((n) == add) ? 1 : 0)

/* A frame ends after N time slots, i.e., when the header next time
reaches the node after the node has started transmitting */
#define EndOfFrame(ns,n,h) ((((ns) == 1) && ((h) == (n))) ? 1 : 0)

/* A frame is empty, if its destination address is 0 or the address of
the node */
#define EmptyFrame(n,add) ((((n) == (add)) || ((add) == 0)) ? 1 : 0)

/* A start of the frame and a frame synchronization signal is in the
position h */
#define Header(n,h) (((n) == (h)) ? 1 : 0)

/* A node is sending, if its node sending bit is set to 1 */
#define NodeSending(ns) (((ns) == 1) ? 1 : 0)

/* A node receives requests, if its request bit is set to 1 */
#define ReqIn(r) (((r) == 1) ? 1 : 0)

/* A node has a turn, if its turn bit is set to 1 */
#define Turn(t) (((t) == 1) ? 1 : 0)

/* A frame is available, if the frame synchronization signal reaches
the node and the frame is empty*/
#define FrameAvailable(n,h,a,f) ((Header((n),(h))) ?
((EmptyFrame((n),(a))) ? 1 : 0) : (f))

/* Places:
```

```
NODEDATA includes N tuples, containg all variables of N nodes in the
         order <node number, frame available, turn, node sending>. 0
         denotes FALSE and 1 denotes TRUE.

REQUESTS contains N triples, denoting the request value of each node
         before the execution of MAC and after the execution of
         MAC. <node number, old request value, new request value>

HEADERandDEST contains a pair, denoting the position of the frame
         synchronization signal in the ring and the destination
         address of the frame. <header position, destination address
         of the frame> 0 denotes an unoccupied frame.

ADDRESSES contains all possible destination addresses.

COUNT denotes the propagation of the bus clock signal in the ring.
      After each firing of the MAC transition, the BUS CLOCK
      transition fires N times.

*/

#place NODEDATA lo(<.1,0,0,0.>) hi(<.N,1,1,1.>) mk(<.1..N,0,0,0.>)

#place REQUESTS lo(<.1,0,0.>) hi(<.N,1,1.>) mk(<.1..N,0,0.>)

#place HEADERandDEST lo(<.1,0.>) hi(<.N,N.>) mk(<.1,0.>)

#place ADDRESSES lo(<.1.>) hi(<.N.>) mk(N<.1..N.>)

#place COUNT lo(<.0.>) hi(<.N.>) mk(<.0.>)

/*

The CLOCK transition moves the request from the node (i mod N) + 1 to
the node i in N steps and the header from the node i to the node (i
mod N) + 1 to the node i.

*/

#trans CLOCK
in{
     REQUESTS: <.c,oldr1,newr1.>+<.(c%N)+1,oldr2,newr2.>;
     HEADERandDEST: (c==1)<.h,d.>;
     COUNT: <.c.>;
}
out{
     REQUESTS: <.c,oldr1,oldr2.>+<.(c%N)+1,oldr2,newr2.>;
     HEADERandDEST: (c==1)<.(h%N)+1,d.>;
```

```
     COUNT: <.(c+1)%(N+1).>;
}
gate (c>0);
#endtr

/*

The transition MAC implements the medium access control algorithm. The
firing of the transition updates all the tuples, denoting the
variables of the nodes, simultaneously, which reduces the size of the
reachability graph.

*/

#trans MAC
in{
     NODEDATA: <.1,f1,t1,ns1.>+  /* the variables of the nodes */
               <.2,f2,t2,ns2.>;  /* DataIn */
     ADDRESSES: <.add1.>+<.add2.>; /* destination addresses of the
                                      nodes, addresses are chosen
                                      randomly from ADDRESSES */
     REQUESTS: <.1,or1,nr1.>+<.2,or2,nr2.>; /* request values of the
                                                nodes */
                                             /* ReqIn */
     HEADERandDEST: <.h,a.>; /* header position and destination
                                address of the frame */
     COUNT: <.0.>; /* denoting the position of the bus clock signal */
}
out{
     NODEDATA: <.1,F1,T1,NS1.>+ /* DataOut */
               <.2,F2,T2,NS2.>;
     ADDRESSES: <.add1.>+<.add2.>;
     REQUESTS: <.1,R1,R1.>+<.2,R2,R2.>; /* ReqOut */
     HEADERandDEST: <.h,A.>; /* HDOut */
     COUNT: <.1.>;
}

/*

With gate expression unnecessary transition instances (instances that
lead to the same marking) are avoided. Instances are avoided by
restricting the destination address of the node i to i, unless the
node starts the transmission of a new frame.

*/

gate (((add1 == 1) || ((!(NodeSending(ns1))) &&
FrameAvailable(1,h,a,f1) && ((Turn(t1) && Header(1,h)) || ((!Turn(t1))
&& Header(1,h) && (!(ReqIn(nr1)))))))
```

```
&& ((add2 == 2) || (((!(NodeSending(ns2))) && FrameAvailable(2,h,a,f2)
&& ((Turn(t2) && Header(2,h)) || ((!Turn(t2)) && Header(2,h) &&
(!(ReqIn(nr2))))))));

/*

The computation of the new variables of the nodes according to the MAC
algorithm

*/

comp {

/* Node 1 */

   F1 = FrameAvailable(1,h,a,f1);
   T1 = (NodeSending(ns1)) ? ((EndOfFrame(ns1,1,h)) ? 0 : t1) : ((t1
== 0 && nr1 == 0) ? 1 : t1);
   NS1 = (NodeSending(ns1)) ? ((EndOfFrame(ns1,1,h)) ? 0 : ns1) :
((F1) ? ((Turn(t1)) ? ((Header(1,h)) ? 1 : ns1) : ((Header(1,h)) ?
((ReqIn(nr1)) ? ns1 : 1) : ns1)) : ns1);
   if (Header(1,h)) {
   A = (!(NodeSending(ns1))) ? ((F1) ? ((Turn(t1)) ? ((Header(1,h)) ?
add1 : ((AddressedToNode(1,a)) ? 0 : a)) : ((Header(1,h)) ?
((ReqIn(nr1)) ? ((AddressedToNode(1,a)) ? 0 : a) : add1) :
((AddressedToNode(1,a)) ? 0 : a))) : ((AddressedToNode(1,a)) ? 0 : a))
: ((AddressedToNode(1,a)) ? 0 : a);
   }
   R1 = (NodeSending(ns1)) ? 0 : ((F1) ? ((Turn(t1)) ? ((Header(1,h))
? 0 : nr1) : ((Header(1,h)) ? ((ReqIn(nr1)) ? nr1 : 0) : ((ReqIn(nr1))
? nr1 : 0))) : ((Turn(t1) || ReqIn(nr1)) ? 1 : nr1));

/* Node 2 */

   F2 = FrameAvailable(2,h,a,f2);
   T2 = (NodeSending(ns2)) ? ((EndOfFrame(ns2,2,h)) ? 0 : t2) : ((t2
== 0 && nr2 == 0) ? 1 : t2);
   NS2 = (NodeSending(ns2)) ? ((EndOfFrame(ns2,2,h)) ? 0 : ns2) :
((F2) ? ((Turn(t2)) ? ((Header(2,h)) ? 1 : ns2) : ((Header(2,h)) ?
((ReqIn(nr2)) ? ns2 : 1) : ns2)) : ns2);
   if (Header(2,h)) {
   A = (!(NodeSending(ns2))) ? ((F2) ? ((Turn(t2)) ? ((Header(2,h)) ?
add2 : ((AddressedToNode(2,a)) ? 0 : a)) : ((Header(2,h)) ?
((ReqIn(nr2)) ? ((AddressedToNode(2,a)) ? 0 : a) : add2) :
((AddressedToNode(2,a)) ? 0 : a))) : ((AddressedToNode(2,a)) ? 0 : a))
: ((AddressedToNode(2,a)) ? 0 : a);
   }
   R2 = (NodeSending(ns2)) ? 0 : ((F2) ? ((Turn(t2)) ? ((Header(2,h))
? 0 : nr2) : ((Header(2,h)) ? ((ReqIn(nr2)) ? nr2 : 0) : ((ReqIn(nr2))
```

```
? nr2 : 0))) : ((Turn(t2) || ReqIn(nr2)) ? 1 : nr1));

Accept ();
}
#endtr
```