MUTUAL EXCLUSION VERIFICATION OF PARAMETERIZED READER-WRITER ALGORITHM: A CASE STUDY

Cecilia E. Nugraheni
Computer Sciece Department, Parahyangan Catholic University
Jl. Ciumbuleuit 94 Bandung 40141
E-mail:cheni@home.unpar.ac.id

ABSTRACT

This paper presents the verification of mutual-exclusion properties of parameterized reader-writer algorithm. A class of diagram called Predicate diagrams [1] is used for representing the abstractions of parameterized systems described by specifications written in TLA. The verification is done by integrating deductive verification and algorithmic techniques. The correspondence between the original specification and the diagram is established by non-temporal proof obligations. Whereas model checker SPIN [3] is used to verify properties over finite-state abstractions.

Keywords: *formal method, verification, parameterized system, temporal logic.*

1. INTRODUCTION

Verification consists of establishing whether a system satisfies some property, that is, whether all possible behaviors of the system are included in the property specified. There are basically two approaches to formal verification, which are the deductive approach and the algorithmic approach. The deductive approach is based on *verification rules*, which reduces the system validity of a temporal property to the general validity of a set of first-order *verification conditions*. The most popular algorithmic verification method is *model checking*. Although this method is fully automatic for finite-state systems, it suffers from the so-called *state-explosion* problem.

A parameterized parallel system, or a parameterized system for short, consists of several similar processes whose number is determined by an input parameter. Many interesting systems are of this form, for example, mutual exclusion algorithms for an arbitrary number of processes wanting to use a common resource. To provide methods for the uniform verification of such systems is a challenging problem. The key to such a uniform treatment is parameterization, i.e. presenting a single syntactic object that actually represents a family of objects.

In the context of parameterized systems, the ability to conduct a uniform verification of a parameterized system is one of the striking advantages of the deductive method for temporal verification over algorithmic techniques such as model-checking techniques. In general, nothing can be concluded about the property holding for any value of n from the fact that it holds for some finite set of values. In comparison, the deductive method establishes in one fell swoop the validity of the property for any value of n [5,2].

The need for a more intuitive approach to verification leads to the use of *diagram-based formalisms*. Usually, these diagrams are graphs whose vertices are labeled with first-order formulas, representing sets of system states, and whose edges represent possible system transitions. This approach combines some of the advantages of deductive and algorithmic verification: the process is goal-directed, incremental and can handle infinite-state systems.

This paper presents the verification of mutual exclusion properties of parameterized reader-writer algorithm. This algorithm can be considered as a parameterized system, since it can handle arbitrary numbers of readers and writers.

We used the diagram-based method. In particular, we adopted the approach proposed by Cansell et.al. In [1] they presented a class of diagrams called predicate diagrams and showed how the using of these in formal verification. We made a little modification of the definition of the original predicate diagrams, in particular the definition related to the actions. Instead of actions, we concentrate only on parameterized actions which are actions of the form $A(k)$. This form of actions is usually used in modeling actions of a particular process in the system. We use TLA* [6] to formalize our approach and use TLA$^+$ style [4] for writing specifications.

This paper is structured as follows. We begin with the informal description of reader-writer algorithm. Section 2 explains briefly the formal specification of parameterized systems in TLA*. Section 3 describes the definition and the use of predicate diagrams in the verification of parameterized systems. The solution of the case study is presented in Section 4. Finally, conclusion and future work will be given in Section 5.

2. PARAMETERIZED READER-WRITER ALGORITHM

The protocol is expressed in terms of four sets that contain (the identities of) currently active readers, waiting readers, active writers and waiting writers. All the sets are initially empty. Assume there are M processes running in the systems. Every process in the system can perform the following actions:

- *ImmRd*(k): Process k signals its desire to read. If there are no active or waiting writers, then it is immediately proceed to read the data.
- *ReqRd*(k): Process k requests reading access, but there are active or waiting writers. It is added to the set of waiting readers.
- *GrRd*(k): Waiting reader k is allowed to read data when there are no active writers.
- *EndRd*(k): Active reader k signals that it has finished reading and exists the protocol.
- *ImmWr*(k): Process k signals its desire to write. If there are neither active or waiting writers nor active or waiting readers, it may immediately proceed to write the data.
- *ReqWr*(k): Process k signals that it wants to write data, but the preconditions of action *ImmWr*(k) are not met. It is added to the set of waiting writers.
- *GrWr*(k): Waiting writer k is allowed to write when there are neither active or waiting readers nor active writers.
- *EndWr*(k): Active writer k signals that it has finished writing and exits the protocol.

The fairness conditions are chosen such that no process that has entered the system is persistently neglected. In particular, there is a strong fairness condition on the action *DoWr*(k), for all $k \in M$, presumably implemented using a queue rather than a set of waiting readers.

Our objective is to prove that the algorithm satisfies the following properties:

1. Mutual-exclusion among writers, i.e. every time there is only maximal one writing process.
2. Mutual-exclusion among readers and writers, i.e. every time whenever there are active readers then there are no active writers and vice versa.

3. PARAMETERIZED SYSTEMS SPECIFICATION

In this work we have restricted to a class of parameterized systems that are interleaving and consist of a finitely, but arbitrarily, discrete components.

Let M denotes a finite and non-empty set of processes running in the system being considered. A parameterized system can be described as a formula of the form:

$$parSpec \equiv Init \wedge \Box\, [\exists k \in M{:}Next(k)]_v \wedge \forall\, k \in M : L(k)$$

where

- *Init* is a state predicate that describes the global initial condition,
- *Next*(k) is an action characterizes the next-state relation of a process k,
- v is a state function representing the variables of the system and
- $L(k)$ is a formula stating the liveness conditions expected from the process or subsystem k.

Actions of the form $A(k)$ are called parameterized actions.

4. PREDICATE DIAGRAMS

Now we present a class of diagrams that will be used for the verification of parameterized systems.

4.1 Definition of predicate diagrams

The underlying assertion language, by assumption, contains a finite set $\mathbf{O}$ of binary relation symbols $\subset$ that are interpreted by well-founded orderings. For $\subset \in \mathbf{O}$, its reflexive closure is denoted by $\subseteq$. We write $\mathbf{O}^{=}$ to denote the set of relation symbols $\subset$ and $\subseteq$ for $\subset \in \mathbf{O}$.

A predicate diagram is a finite graph whose nodes are labeled with sets of (possibly negated) predicates, and whose edges are labeled with actions (more precisely, action names) as well as optional annotations that assert certain expressions to decrease with respect to an ordering in $\mathbf{O}^{=}$. Intuitively, a node of a predicate diagram represents the set of system states that satisfy the formulas contained in the node. (We indifferently write n for the set and the conjunction of its elements.) An edge (n,m) is labeled with action A if A can cause a transition from a state represented by n to a state represented by m. An action A may have an associated fairness condition; fairness conditions apply to all transitions labeled by the action rather than to individual edges.

Formally, the definition of predicate diagrams is relative to finite sets $\wp$ and α that contain the state predicates and the (names of) parameterized actions of interest; we will later use $\tau \notin \alpha$ to denote a special stuttering action. We write $Cl(\wp)$ to denote the set of literals formed by the predicates in $\wp$, that is, the union of $\wp$ and the negations of the predicates in $\wp$.

Definition 1. Assume given two finite sets $\wp$ and α of state predicates and parameterized action names. A predicate diagram $G = (N,I,\delta,o,\zeta)$ over $\wp$ and α consists of:

- a finite set $N \subseteq 2^{Cl(\wp)}$ of nodes,
- a finite set $I \subseteq N$ of *initial nodes*,
- a family of δ_A where $A \in \alpha$ of relations $\delta_{A \subseteq N \times N}$; we also denote by δ the union of the relations δ_A, for $A \in \alpha$ and write $\delta_=$ to denote the reflexive closure of the union of these relations,
- an edge labeling o that associates a finite set $\{(t_1,<_1), \ldots, (t_k, <_k)\}$, of terms t_i paired with a relation $<_i \in \mathbf{O}^{=}$ with every edge $(n,m) \in \delta$, and

- a mapping $\zeta:\alpha \rightarrow \{\text{NF},\text{WF},\text{SF}\}$ that associates a fairness condition with every parameterized action in α; the possible values represent no fairness, weak fairness, and strong fairness.

We say that the parameterized action $A \in \alpha$ can be taken at node $n \in N$ iff $(n,m) \in \delta_A$ holds for some $m \in N$, and denote by $En(A) \subseteq \text{N}$ the set of nodes where A can be taken. We say that the parameterized action $A \in \alpha$ can be taken along an edge (n,m) iff $(n,m) \in \delta_A$.

We now define *runs* and *traces* through a diagram as the set of those behaviors that correspond to fair runs satisfying the node and edge labels. To evaluate the fairness conditions we identify the enabling condition of a parameterized action $A \in \alpha$ with the existence of A-labeled edges at a given node.

Definition 2. Let $G = (N,I,\delta,\text{o},\zeta)$ be a predicate diagram over sets $\wp$ and α. A run of G is an ω-sequence $\sigma = (s_0,n_0,A_0)\ (s_1,n_1,A_1)\ \ldots$ of triples where s_i is a state, $n_i \in N$ is a node and $A_i \in \alpha \cup \{\tau\}$ is a parameterized action such that all of the following conditions hold:

- $n_0 \in I$ is an initial node.
- $s_i\ \|[n_i]\|$ holds for all $i \in \mathbf{N}$.
- For all $i \in \mathbf{N}$, either $A_i = \tau$ and $n_i = n_{i+1}$ or $A_i \in \alpha$ and $(n_i,n_{i+1}) \in \delta_{A_i}$. $A_i \in \alpha$ and
- $(t,<) \in o(n_i,n_{i+1})$, then $s_{i+1}\ \|[t]\| < s_i\ \|[t]\|$.
- If $A_i = \tau$ then $s_{i+1}\ \|[t]\| \leq s_i\ \|[t]\|$ holds whenever $(t, <) \in o(n_i,m)$ for some $m \in N$.
- For every parameterized action $A \in \alpha$ such that $\zeta(A) = \text{WF}$ there are infinitely many $i \in \mathbf{N}$ such that either $A_i = A$ or $n_i \notin En(A)$.
- For every parameterized action $A \in \alpha$ such that $\zeta(A) = \text{SF}$, either $A_i = A$ holds for infinitely many $i \in \mathbf{N}$ or $n_i \in En(A)$ holds for only finitely many $i \in \mathbf{N}$.

We write *runs*(G) to denote the set of runs of G.
The set $tr(G)$ of traces through G consists of all behaviors $\sigma = s_0 s_1 \ldots$ such that there exists a run $\sigma = (s_0,n_0,A_0)\ (s_1,n_1,A_1)\ \ldots$ of G based on the states in σ.

4.2 Verification using predicate diagrams

The verification process using predicate diagrams is done in two steps. The first step is to find a predicate diagram that can be proven to be the correct representation of the system being verified, i.e. the diagram conforms to the system specification. We say that a predicate diagram G *conforms* to a parameterized system *parSpec* if every behavior that satisfies *parSpec* is a trace through G.

Theorem 1
Let $G = (N,I,\delta,o,\zeta)$ be a predicate diagram over $\wp$ and α and let $parSpec = Init \wedge \Box\ [\exists k \in M{:}Next(k)]_v \wedge \forall\ k \in M : L(k)$ be a parameterized system. If all the following conditions hold then G conforms to *parSpec*:

1. For all $n \in I$, $\models Init \rightarrow n$.
2. $|\!\approx n \wedge [\exists\ k \in M : Next(k)]_v \rightarrow n' \vee \bigwedge_{(m,A(k)):(n,m)\in\delta_{A(k)}} \langle \exists\ k \in M : A(k) \rangle_v$
3. For all $n,m \in N$ and all $(t,<) \in o(n,m)$:
   a. $|\!\approx n \wedge m' \wedge \bigwedge_{A(k):(n,m)\in\delta_{A(k)}} \langle \exists k \in M : A(k) \rangle v \rightarrow t' < t$.
   b. $|\!\approx n \wedge [\exists k \in M{:}Next(k)]_v \wedge n' \rightarrow t' \leq t$.
4. For every parameterized action $A(k) \in \alpha$ such that $\zeta(A(k) \neq \text{NF}$:
   a. If $\zeta(A(k) = \text{WF}$ then $\models parSpec \rightarrow \text{WF}_v(\exists\ k \in M : A(k))$.
   b. If $\zeta(A(k) = \text{SF}$ then $\models parSpec \rightarrow \text{SF}_v(\exists\ k \in M : A(k))$.
   c. $|\!\approx n \rightarrow \text{ENABLED}\ \langle \exists k \in M{:}\ \text{A}(k) \rangle_v$ holds whenever $n \in En(A(k))$.
   d. $|\!\approx n \wedge \langle \exists k \in M{:}\text{A}(k) \rangle_v \rightarrow \neg\ m'$ holds for all $n,m \in N$ such that $(n,m) \notin \delta_{A(k)}$.

The second step of the verification process is to prove that that all traces through a predicate diagram satisfy some property F we view the diagram as a finite transition system that is amenable to model checking. All predicates and actions that appear as labels of nodes or edges are then viewed as atomic propositions.

Regarding predicate diagrams as finite labeled transition systems, their runs can be encoded in the input language of standard model checkers such as SPIN [3]. Two variables indicate the current node and the last action taken. The predicates in $\wp$ are represented by boolean variables, which are updated according to the label of the current node, non-deterministically, if that label contains neither P nor $\neg P$. We also add variables $b_{(t,<)}$, for every term t and relation $< \in \mathbf{O}$ such that $(t,<)$ appears in some ordering annotation $o(n,m)$. These variables are set to 2 if the last transition taken is labeled by $(t,<)$, to 1 if it is labeled by $(t,\leq)$ or is stuttering transition and to 0 otherwise. Whereas the fairness conditions associated with the actions of a diagram are easily expressed as LTL assumptions for SPIN [3].

5. SOLUTION

The specification of parameterized reader-writer algorithm is given in Fig. 2. There are four variables, which are *ar*, *wr*, *aw* and *ww*, representing the set of active readers, waiting readers, active writers and waiting writers.

Fig. 2 shows a suitable predicate diagram for the parameterized reader-writer algorithm. We set $\text{Cl}(\wp)$ to contain the eight predicates, which are: $|ar| = 0$, $|ar| > 0$, $|wr| = 0$, $|wr| > 0$, $|aw| = 0$, $|aw| = 1$, $|ww| = 0$ and $|ww| > 0$. It is assumed that the following conditions hold: $\neg(|ar| = 0) \leftrightarrow |ar| > 0$, $\neg(|wr| = 0) \leftrightarrow |wr| > 0$, $\neg(|aw| = 0) \leftrightarrow |aw| = 1$ and $\neg(|aw| = 0) \leftrightarrow |aw| > 0$. It can be argued that $\neg(|aw| = 0) \leftrightarrow |aw| = 1$ holds, since the number of the active writers is never greater than 1. *IR*, *RR*, *GR*, *ER*, *IW*, *RW*, *GW* and *EW* stands for *ImmRd*(*k*), *ReqRd*(*k*), *GrRd*(*k*), *EndRd*(*k*), *ImmWr*(*k*), *ReqWr*(*k*), *GrWr*(*k*) and *EndWr*(*k*), respectively.

Using Theorem 1 it can be shown that the predicate diagram in Fig. 2 conform to the specification in Fig. 1. For example, we have:

- $Init \rightarrow |ar| = 0 \wedge |wr| = 0 \wedge |aw| = 0 \wedge |ww| = 0$
- $|ar| = 0 \wedge |wr| = 0 \wedge |\mathrm{aw}| = 0 \wedge |\mathrm{ww}| = 0 \wedge [\exists\, k \in M : Next(k)]_v \rightarrow$
  $|ar| = 0 \wedge |wr| = 0 \wedge |aw| = 0 \wedge |ww| = 0 \vee \langle \exists\, k \in M : ImmRd(k)\rangle_v \wedge |ar| > 0 \wedge |wr| = 0 \wedge |aw| = 0 \wedge |ww| = 0 \vee$
  $\langle \exists\, k \in M : ImmWr(k)\rangle_v \wedge |ar| = 0 \wedge |wr| = 0 \wedge |aw| = 1 \wedge |ww| = 0$

The next step is to encode the predicate diagram in Promela, the input language of SPIN. To do this, six variables are used which are *action*, *node*, *ar*, *wr*, *aw*, and *ww*. *action* and *node* are used to indicate the last action taken and the current node; whereas *ar*, *wr*, *aw* and *ww* are used to represent the predicates that hold on every node, for example, if *ar* = 1 then the predicate $|ar| > 0$ holds and if *ar* = 0 then the predicate $|ar| = 0$ holds. *action* = 1 if *ImmRd*(*k*) is taken, *action*=2 is *ReqRd*(*k*) is taken and so on.

The theorems are now can be written as $\Box(aw = 0 \vee aw = 1)$, $\Box(aw = 1 \rightarrow ar = 0)$ and $\Box(ar = 1 \rightarrow aw = 0)$. Last, by using SPIN we model-checked the resulted transition system. As result, we concluded that the algorithm satisfies all the properties we want to prove.

6. CONCLUSION

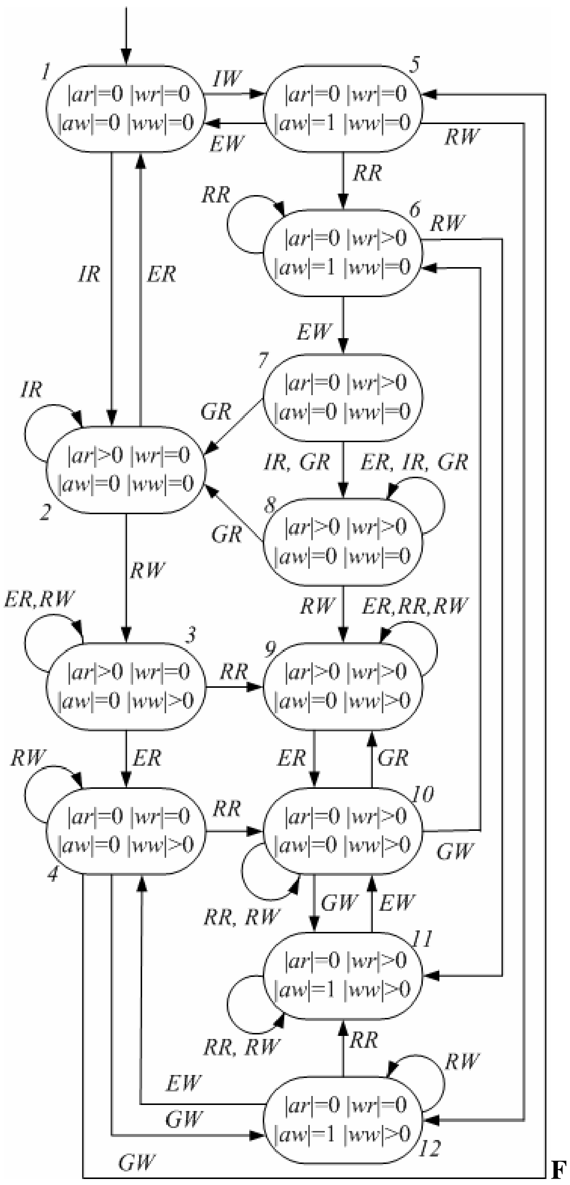
We have presented a diagram-based verification of parameterized systems. The parameterized systems are represented as parameterized TLA specifications. The verification is done deductively and algorithmically by means of predicate diagrams. We have modified the definition of predicate diagrams and their runs in order to work with parameterized systems.

By using the parameterized reader-writer algorithm as a case study, we have shown that the modified predicate diagrams can be used to prove the mutual exclusion properties. Note that we didn't consider the liveness conditions on this case. We are allowed to do this since the mutual-exclusion properties are safety properties. The situation is different if the properties we want to prove are liveness properties, such as *every time a process requests to read, it will be eventually allowed to read.*

$$\begin{aligned}
&Init \equiv ar = \emptyset \wedge wr = \emptyset \wedge aw = \emptyset \wedge \mathrm{ww} = \emptyset\\
&ImmRd(k) \equiv \wedge\, k \notin ar \cup wr \cup aw \cup ww \wedge aw = \emptyset\\
&\qquad \wedge\, ww = \emptyset \wedge ar' = ar \cup k\}\\
&\qquad \wedge\, \mathrm{UNCHANGED}\langle wr,aw,ww\rangle\\
&ReqRd(k) \equiv \wedge\, k \notin ar \cup wr \cup aw \cup ww\\
&\qquad \wedge\, \neg(aw{=}\emptyset \wedge ww{=}\emptyset) \wedge wr' = wr \cup \{k\}\\
&\qquad \wedge\, \mathrm{UNCHANGED}\,\langle ar,aw,ww\rangle\\
&GrRd(k) \equiv \wedge\, k \in wr \wedge aw = \emptyset \wedge ar' = ar \cup \{k\}\\
&\qquad \wedge\, wr' = wr / \{k\}\\
&\qquad \wedge\, \mathrm{UNCHANGED}\langle \mathrm{aw},\ \mathrm{ww}\rangle\\
&EndRd(k) \equiv \wedge\, k \in ar \wedge ar' = ar \setminus \{k\}\\
&\qquad \wedge\, \mathrm{UNCHANGED}\,\langle wr,aw,ww\rangle\\
&ImmWr(k) \equiv \wedge\, k \notin ar \cup wr \cup aw \cup ww\\
&\qquad \wedge\, ar \cup wr \cup aw \cup ww = \emptyset \quad \wedge\\
&\qquad aw' = aw \cup \{k\}\\
&\qquad \wedge\, \mathrm{UNCHANGED}\,\langle ar,wr,ww\rangle\\
&ReqWr(k) \equiv \wedge\, k \notin ar \cup wr \cup aw \cup ww \wedge ww' = ww \cup\\
&\qquad \{k\}\\
&\qquad \wedge\, a\ r \cup wr \cup aw \cup ww \neq \emptyset\\
&\qquad \wedge \mathrm{UNCHANGED}\langle ar,wr,aw\rangle\\
&GrWr(k) \equiv \quad \wedge\, k \in ww \wedge a \cup w \cup aw = \emptyset \wedge aw' = aw\\
&\qquad \cup\{k\}\\
&\qquad \wedge\, ww' = ww \setminus \{k\} \wedge \mathrm{UNCHANGED}\\
&\qquad \langle ar,wr\rangle\\
&EndWr(k) \equiv \qquad \wedge\, k \in aw \wedge aw' = aw \setminus \{k\}\\
&\qquad \wedge\, \mathrm{UNCHANGED}\,\langle ar,wr,ww\rangle\\
&Next(k) \equiv \quad \vee\, ImmRd(k) \vee ReqRd(k) \vee GrRd(\mathrm{k})\\
&\qquad \vee\, EndRd(\mathrm{k}) \vee ImmWr(k) \vee ReqWr(\mathrm{k})\\
&\qquad \vee\, GrWr(k) \vee EndWr(k)\\
&v \equiv \langle ar,wr,\mathrm{aw},ww\rangle\\
&L(k) \equiv \mathrm{WF}_v(GrRd(k)) \wedge \mathrm{WF}_v(EndRd(k)) \wedge\\
&\qquad \mathrm{SF}_v(GrWr(k)) \wedge \mathrm{WF}v(EndWr(k))\\
&ParRW \equiv \forall\, k \in 1..M{:}Init \wedge \Box\, [Next(k)]_v \wedge L(k)\\
&\mathrm{THEOREM}\ ParRW \rightarrow \Box\, (|aw| \leq 1)\\
&\mathrm{THEOREM\ ParRW} \rightarrow \Box\, (|aw| = 1 \rightarrow |ar| = 0)\\
&\mathrm{THEOREM\ ParRW} \rightarrow \Box\, (|ar| > 0 \rightarrow |aw| = 0)
\end{aligned}$$

Fig. 1. Specification of parameterized reader-writer.

It is planned to continue this work with verification of universal properties of parameterized reader-writer algorithm. Universal properties are properties expressed as formulas of the form $\forall k \in 1..n{:}P(k)$ for *P* is some property. On this case, another class of diagram will be used, which is Parameterized Predicate Diagram [7,8].



F

ig. 2. Suitable predicate diagram for reader-writer algorithm.

REFERENCES

[1] Dominique Cansell, Dominique Méry and Stephan Merz. Predicate diagrams for the verification of reactive systems. In 2^{nd} *Intl. Conf. on Integrated Formal Methods IFM 2000*, vol. 1945 of Lectures Notes in Computer Science, Dagstuhl, Germany, November 2000. Springer-Verlag.

[2] E.A. Emerson and K.S. Namjoshi. Verification of a parameterized bus arbitration protocol. *Volume 1427 of Lecture Notes in Computer Science*, pp. 452--463. Springer,1998.

[3] G. Holzmann. The SPIN model checker. *IEEE Trans*. On software engineering, 16(5):1512-1542. May 1997.

[4] Leslie Lamport. The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, 16(3): 872-923, May 1994.

[5] Zohar Manna and Amir Pnueli. Verification of parameterized programs. *In Specification and Validation Methods* (E. Borger, ed.), Oxford University Press, pp. 167-230, 1994.

[6] Stephan Merz. Logic-based analysis of reactive systems: hiding, composition and abstraction. *Habilitationsschrift*. Institut für Informatik. Ludwig-Maximillians-Universität, Munich Germany. December 2001.

[7] Cecilia E. Nugraheni. Predicate diagrams as basis for the verification of reactive systems. *PhD Thesis*. Institut für Informatik. Ludwig-Maximillians-Universität, Munich Germany. February 2004.

[8] Cecilia E. Nugraheni. Universal properties verification of parame-terized parallel systems. In *Proceeding of the International Confe-rence on Computational Scince and its Applications (ICCSA 2005)*, Volume 3482 of Lecture Notes in Computer Science, pages 453-462. Springer, 2005.