Cross-Checking - Enhanced Over-Approximation of the Reachable Global State Space of Component-Based Systems

Mila Majster-Cederbaum and Christoph Minnameier*

Institut für Informatik
Universität Mannheim, Germany
cmm@informatik.uni-mannheim.de

Abstract. State space explosion causes most relevant behavioral questions for component-based systems to be PSPACE-hard. Here, we exploit the structure of component-based systems to obtain a first approximation of the reachable global state space. In order to improve this approximation we introduce a new technique we call cross-checking. The resulting approximation can be used to study global properties of component-based systems, which we demonstrate here for local deadlock-freedom.

1 Introduction

In this paper we deal with reachability in the global state space of a component-based system with n components. We present a technique that builds on the analyses of certain subsystems generated by $d << n$ components, where d is fixed. We explain our approach using the model of *interaction systems* introduced in [GS03] by Gössler and Sifakis as a model for component-based systems. As typical for component-based systems, the description of interaction systems strictly separates the description of the components from the way they are put together, i.e. the *glue code*. I/O-Automata [LT89] and interface automata [dAH01], e.g. can be considered as a subclass of interaction systems, for the latter feature a more general notion of communication. More details about interaction systems and their properties can be found in [Sif04, Sif05, GGM$^+$07b, GGM$^+$07a, GS05, BBS06, MMM07a]. A framework for component-based modeling using interaction systems has been implemented in [BBS06, Goe06]. Please note that the ideas presented in this paper do not rely heavily on the model but can be transferred to other models as long as cooperation of systems forms the top level of system description.

For interaction systems the size of the global state space of a component based system may be exponential in the number n of components and it has been shown that deciding most important behavioral properties is PSPACE-complete [MM08b]. There are various ways to deal with this problem, e.g. partial order reduction or abstraction. Another approach is to establish conditions

* Corresponding author.

O. Bournez and I. Potapov (Eds.): RP 2009, LNCS 5797, pp. 189–202, 2009.

that use compositionality and can be tested in polynomial time to ensure the desired properties, see e.g. [AC05, MMM07a, MMM07b]. Moreover one may impose architectural constraints concerning the communication structure of the component system [AG97, BCD02, MM08a].

In this paper we first exploit the structure of the component system to obtain (in polynomial time) a first over-approximation of the global state space. For this, we consider subsystems built from a fixed number d of components (which can be considered a parameter) and perform reachability analyses there. Restricting our view to sets of subsystems can be considered a form of locality-based abstraction. Different related techniques have been studied in [BPR01, ASCN99, Kov]. A general and abstract treatment of locality-based abstraction can be found in [EGS05].

The contribution of this paper basically consists of the following two steps. First the straight-forwardly computed subsystem approximations are enhanced by a technique called *cross-checking*. Second, the resulting approximations can be used to derive conditions on the subsystems that guarantee global properties.

We demonstrate this for local deadlock-freedom. Deadlock-freedom is an important property in itself and moreover, establishing safety properties can be reduced to establishing deadlock-freedom.

The paper is structured as follows. Section 2 presents the model and an example that will be used throughout the paper. In Section 3 we explain the general approach of investigating subsytems in order to prove properties on the reachable global state space. Section 4 introduces and analyzes cross-checking. As an application we establish in Section 5 a polynomial time checkable condition for deadlock-freedom that is tested in subsystems and makes use of our approximation. Section 6 discusses related work. Section 7 depicts our (partially still in progress) implementations. Finally, we give a short conclusion in Section 8.

2 Interaction Systems

We consider here interaction systems, a model for component-based systems that was proposed and discussed in [GS03].

2.1 Syntax and Semantics

Definition 1. *Interaction Systems*
An ***interaction system*** *is a tuple* $Sys = (K, \{A_i\}_{i\in K}, Int, \{T_i\}_{i\in K})$*, where* K *is the set of* ***components****. W.l.o.g. we assume* $K = \{1, \ldots, n\}$*. Each component* $i \in K$ *offers a finite set of* ***ports*** *(resp.* ***actions****)* A_i *for cooperation with other components. The* ***port sets*** A_i *are pairwise disjoint. Cooperation is described by the* ***interaction set*** *Int. Each component* i *is provided with a* ***local behavior*** T_i*.*

An ***interaction*** *is a finite, non-empty set of actions* $\alpha \subseteq \bigcup_{i\in K} A_i$*. An interaction* $\alpha = \{a_{i_1}, \ldots, a_{i_k}\}$ *with* $a_{i_j} \in A_{i_j}$ *describes that the components* $i_1, \ldots, i_j$ *cooperate via these ports. Interactions* α *are subject to the constraint that for each component* i *at most one action* $a_i \in A_i$ *is in* α*. An* ***interaction set*** *Int*

is a finite set of interactions, s.t. every action of every component occurs in at least one interaction of Int.

The ***local behavior*** *of each component i is described by a labeled transition system $T_i = (Q_i, A_i, \rightarrow_i, q_i^0)$, where Q_i is the finite set of local states, $\rightarrow_i \subseteq Q_i \times A_i \times Q_i$ is the local transition relation and $q_i^0 \in Q_i$ is the local starting state.*

Definition 2. *Participation and Enabled Actions*
Given an interaction $\alpha \in Int$ and a component $i \in K$ we denote by $i(\alpha) := A_i \cap \alpha$ the ***participation*** *of i in α.*
For $q_i \in Q_i$ we define the set of ***enabled actions*** *$ea(q_i) := \{a_i \in A_i \mid \exists q_i' \in Q_i,$ s.t. $q_i \xrightarrow{a_i}_i q_i'\}$. We assume that the T_i's are non-terminating, i.e. $\forall i \in K\ \forall q_i \in Q_i\ ea(q_i) \neq \emptyset$.*

Definition 3. *Semantics*
The ***global behavior*** *$T_{Sys} = (Q, Int, \rightarrow_{Sys}, q^0)$ of Sys (henceforth also referred to as global transition system) is obtained from the behaviors of the individual components, given by the transition systems T_i, and the interaction set Int in a straightforward manner:*

- *$Q = \prod_{i \in K} Q_i$, the Cartesian product of the Q_i, which we consider to be order independent. We denote states by tuples $(q_1, \ldots, q_n)$ and call them* ***global states****.*
- *The relation $\rightarrow_{Sys} \subseteq Q \times Int \times Q$, defined by*
  $\forall \alpha \in Int\, \forall q, q' \in Q \quad q = (q_1, \ldots, q_n) \xrightarrow{\alpha}_{Sys} q' = (q_1', \ldots, q_n') \quad$ iff
  $\forall i \in K \quad (q_i \xrightarrow{i(\alpha)}_i q_i'$ if $i(\alpha) \neq \emptyset$ and $q_i' = q_i$ otherwise).
- *$q^0 = (q_1^0, \ldots, q_n^0)$ is the global starting state for Sys.*

Less formally, a transition labeled by α may take place in the global transition system when each component i participating in α is ready to perform its part $i(\alpha)$.

Example 1. In the following we consider an interaction system that models Tanenbaum's solution [Tan08] to Dijkstra's Dining Philosophers problem. Tanenbaum suggests that each of the philosophers is provided with a separate semaphore that she has to set in order to leave her thinking state. A semaphore however can only be set if its "neighbour" semaphores are unset. Once a philosopher has eaten, she puts back the forks and resets her semaphore. This can be considered an elegant solution as it is symmetric and allows for maximum efficiency (meaning that it still allows for a global state where every second philosopher is in her eating state). On the other hand this is a deadlock-free system with a natural interaction structure whose reachable global state space is exponential in p. This solution can be modeled as an interaction system as follows, where p is the number of philosophers:
$DP(p) = (K(p), \{A_i\}_{i \in K(p)}, Int(p), \{T_i\}_{i \in K(p)})$, where
$K(p) = \{Phil_0, \ldots, Phil_{p-1}, Fork_0, \ldots, Fork_{p-1}, Sem_0, \ldots, Sem_{p-1},\}$,
$Int(p) = \bigcup_{0 \leq i \leq p-1} \{\{pickleft_{Phil_i},\ occupy_{Fork_i}\}, \{pickright_{Phil_i},\ occupy_{Fork_{i-1}}\},$
$\{priority_{Phil_i},\ down_{Sem_i},\ allow_{Sem_{i-1}},\ allow_{Sem_{i+1}}\},$
$\{drop_{Phil_i},\ up_{Sem_i},\ vacate_{Fork_{i-1}},\ vacate_{Fork_i}\}\},$

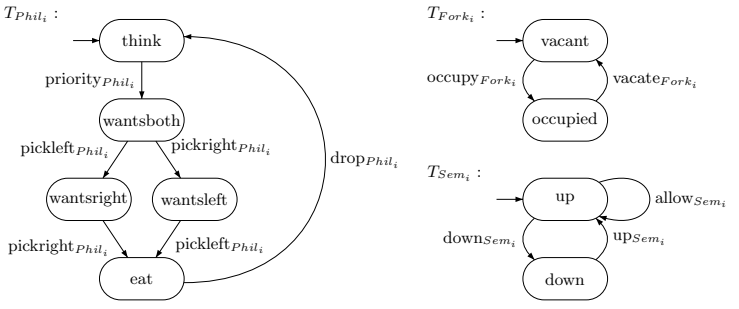

Fig. 1. Tanenbaum's Dining Philosophers - Local Transition Systems

where calculation is modulo p, and the local behaviors T_i and (implicitly the) port sets A_i are given in Figure 1.

2.2 Reachability

For most properties of interaction systems we must determine which states in Q are reachable from the global starting state. Here we propose to first investigate reachability in subsystems which is defined as follows.

Definition 4. ***Reach*(*Sys*)** $:= \{q \in Q \mid q^0 \rightarrow^*_{Sys} q\}$, *where* $\rightarrow^*_{Sys}$ *denotes the reflexive and transitive closure of* $\rightarrow_{Sys}$.

Definition 5. *Substates*
Let $K' \subseteq K$ *and* q *be a global state. Then* $q \downarrow K'$ *denotes the projection of* q *to the components in* K' *and we call* $q' = q \downarrow K'$ *a* ***substate****. We refer to the components* K' *that occur in* q' *by* $K(q')$*. We also use the* $\downarrow$*-operator to denote projections of substates. Finally, let* $Q_{K'} = \prod_{i \in K'} Q_i$ *and* $Subs(K) = \bigcup_{K' \subseteq K} Q_{K'}$.

Definition 6. *Subsystems*
Let $K' \subseteq K$*. The* ***subsystem*** $Sys_{K'}$ *is given by* $(K', \{A_i\}_{i \in K'}, Int_{K'}, \{T_i\}_{i \in K'})$*, where* $Int_{K'} := \{\alpha_{K'} = \alpha \cap (\bigcup_{i \in K'} A_i) \mid \alpha \in Int\} \setminus \{\emptyset\}$*. Note that* $Sys_{K'}$ *accords to our definition of an interaction system, so all definitions for interaction systems apply.*

Definition 7. *Extensions*
Let q' *be a substate. Then* $Ext(q', K')$ *for* $K' \subseteq K$ *denotes the set of* ***extensions*** *of* q' *in* K' *and is defined by* $Ext(q', K') = q' \times \prod_{i \in K' \setminus K(q')} Q_i$*. If* $K' = K(q')$ *let* $Ext(q', K') = \{q'\}$*. We say that a substate* $\hat{q}'$ *is an extension of a substate* q' *if* $K(q') \subseteq K(\hat{q}')$ *and* $\hat{q}' \downarrow K(q') = q'$.

The definition of a subsystem implies that if a state q is reachable in the global transition system, then for every $K' \subseteq K$ the state $q \downarrow K'$ is reachable in the corresponding subsystem. We formalize this observation in the following lemma.
Lemma 1. *Let* $Sys = (K, \{A_i\}_{i \in K}, Int, \{T_i\}_{i \in K})$
$q \in Reach(Sys) \Rightarrow \forall K' \subseteq K,\ (q \downarrow K') \in Reach(Sys_{K'})$.

Corollary 1. *Let* $f(Reach(Sys_{K'})) := \bigcup_{q' \in Reach(Sys_{K'})} Ext(q', K)$. *Then* $Reach(Sys) \subseteq \bigcap_{K' \subseteq K, \text{ with } |K'|=d} f(Reach(Sys_{K'}))$

Remark 1. Each $Reach(Sys_{K'})$ is a compact representation of $f(Reach(Sys_{K'}))$ which is in turn a very coarse over-approxmation of $Reach(Sys)$.

3 Proving Properties on Overapproximations

Let us assume we want to check a property P on each state of the reachable global state space of *Sys*. In a first approach we might proceed as follows.

- We choose a parameter $d << n$ and calculate the reachable states for each subsystem with d components. Each reachable substate $q' = (q_{i_1}, \ldots, q_{i_d})$ is a compact representation of $Ext(q', K)$.
- We formulate a predicate P' such that the validity of P on a global state q is implied by the validity of P' on the projections of q,
  $[\forall K' \subseteq K, |K'| = d\ P'(q \downarrow K')] \Rightarrow P(q)$, hence
  $[\forall K' \subseteq K, |K'| = d\ \forall q \in Reach(Sys)\ P'(q \downarrow K')] \Rightarrow P(Reach(Sys))$

Clearly, we do not want to handle explicitly global states at all. Instead we propose to use the implication
$[\forall K' \subseteq K, |K'| = d \forall q' \in Reach(Sys_{K'})\ P'(q')] \Rightarrow P(Reach(Sys))$

The advantage of this approach is immense: Instead of a complexity that is exponential in $|K| = n$, we have a complexity that is polynomial (with degree d) in n and m. This is because for $K' \subseteq K$ with $|K'| = d$, $Reach(Sys_{K'})$ can be computed in $O(m^d)$, where $m = \max_{i \in K'} |Q_i|$ thus we may compute resp. store the reachable state spaces of all subsystems with d components in time resp. space $O(\binom{n}{d} \cdot m^d)$.

Example 2. For the dining philosophers example with $p = 6$ (i.e. $|K| = 18$) and $d = 4$ the sum of the sizes of the investigated substate spaces is 229.095 compared to 64.000.000 global states in the original system. Obviously the advantage is much greater for a larger parameter p.

However, there is an obvious drawback to our present approach:

Considering subsystems with $d << n$ components neglects much information. Indeed there will be many reachable substates that do not originate from a projection of a global reachable state but are "artefacts". If we check condition P' on many such artefacts we run the risk that P' is violated and we can not conclude P.

Example 3. For the dining philosophers example with $p = 6$ (i.e. $|K| = 18$) and $d = 4$, only 43212 of the 229.095 states in the state spaces of the subsystems are unreachable. This corresponds to 18,85%.

4 Cross-Checking

In this section, we introduce cross-checking as a technique to eliminate artefacts. In a first step, we consider the unreachable states.

Lemma 2. *Let* $\overline{Reach}(Sys_{K'}) = Q_{K'} \setminus Reach(Sys_{K'})$ *be the set of states that can not be reached in* $Sys_{K'}$, $Refuted = \bigcup_{K'' \subseteq K, |K''|=d} f(\overline{Reach}(Sys_{K''}))$ *and* $X_{K'} := f(Reach(Sys_{K'})) \setminus Refuted$.
Then i) $Refuted \subseteq \overline{Reach}(Sys)$ *and*
ii) $Reach(Sys) \subseteq X_{K'}$

As we noted before, the sets $Reach(Sys_{K'})$ may contain many artefacts. We want to use *Refuted* to reduce the number of artefacts. However we can not use Lemma 2 directly as it involves the evaluation of the function f. Therefore we define $Ref(Sys_{K'}) := \{q' \in Subs(Q_{K'}) \mid Ext(q', K') \cap Reach(Sys_{K'}) = \emptyset\}$ and compute (in polynomial time) a $Reach'(Sys_{K'})$ with $f(Reach(Sys_{K'})) \supseteq f(Reach'(Sys_{K'})) \supseteq X_{K'}$ as follows.

Theorem 1. *Let*
$Reach'(Sys_{K'}) \coloneqq Reach(Sys_{K'}) \setminus Ext(\bigcup_{K'' \subseteq K, |K''|=d} Ref(Sys_{K''}) \cap Subs(K'), K')$.
Then $X_{K'} \subseteq f(Reach'(Sys_{K'}))$.

By *cross-checking* we refer to the computation of the various sets $Reach'(Sys_{K'})$ by removing the elements in $Ext(\bigcup_{K'' \subseteq K, |K''|=d} Ref(Sys_{K''}) \cap Subs(K'), K')$ from $Reach(Sys_{K'})$ as described in Algorithm 1. For reasons of efficiency Algorithm 1 represents each set $Reach(Sys_{K'})$ by an array $reach(Sys_{K'})$ (that we refer to as "table") of booleans for all states in $Q_{K'}$. In such a table we have "$reach(Sys_{K'})[q'] = true$" iff $q' \in Reach(Sys_{K'})$. Also for reasons of efficieny Algorithm 1 does not loop over the various sets $Reach(Sys_{K'})$ and the therein reachable substates but rather over all states in $\{q'' \in Subs(K) \mid |q''| < d\}$. For a state q'' we decide (by looking up the reachability flags of its extensions in the various tables $reach(Sys_{K'})$) whether it belongs to $\bigcup_{K' \subseteq K, |K'|=d} Ref(Sys_{K'})$. If this is the case, we set all reachability flags of all extensions of q'' to false. If this is not the case, we add q'' to an initially empty list "list-of-possible-substates" that will be needed in Section 5.1.

Example 4. Let $K_1' = \{Phil_1, Phil_2, Fork_1, Fork_2\}$. In $Sys_{K_1'}$ we are able (by performing the connectors $\{priority_1\}$ and $\{priority_2\}$) to reach the substate $q' = (priority_{Phil_1}, priority_{Phil_2}, vacant_{Fork_1}, vacant_{Fork_2})$. However if we consider the substate $q'' = (priority_{Phil_1}, priority_{Phil_2})$ of q' and its occurrence in the subsystem that is implied by $K_2' = \{Phil_1, Phil_2, Sem_1, Sem_2\}$ we learn that no extension of q'' is in $Reach(Sys_{K''})$. Thus $q'' \in Ref(Sys''_K) \cap Subs(K')$, so we have $q' \notin Reach'(Sys_{K'})$.

After the first application of cross-checking for the subsystem reachabilites, we will have marked 147561 of the 229095 substates unreachable. This corresponds to 64,41%.

Lemma 3. *The sets* $Reach'(Sys_{K'})$ *for all subsystems* $Sys_{K'}$ *with* d *components can be computed in an overall amount of time that is in* $O(d \cdot n^d \cdot m^d)$.

Proof: In the following we present Algorithm 1 *Cross-Checking* that computes the sets $Reach'(Sys_{K'})$ within the specified time bounds.

Remark 2. Apart from the factor d (which can be considered a constant), our cross-checking algorithm remains within the asymptotic time bounds already given by the first step of performing the reachability analyses of the subsystems. We consider this to be an **important** property, as any refinement approach that attempts to increase the number d of considered components would instead result in a complexity in $\Omega(n^{d+1})$.

Remark 3. Note that Algorithm 1 *may* be applied iteratively to the result of the previous application thus further reducing the number of states that are marked reachable until we reach a fixpoint. It is an open question how many iterations will be needed.

Algorithm 1 Cross-Checking

```
 1: PROCEDURE Cross-Checking
 2: for x := 1 to (d − 1) do
 3:     for all subsets K'' = {i_1, ..., i_x} of K do
 4:         for all q'' = (q_{i_1}, ..., q_{i_x}) ∈ Q_{K''} do
 5:             reachable := true;
 6:             for all subsystems Sys_{K'} with K'' ⊆ K' (and |K'| = d) do
 7:                 occurrence := false;
 8:                 for all q' ∈ Ext(q'', K') do
 9:                     occurrence := occurrence OR reach(Sys_{K'})[q'];
10:                 end for
11:                 reachable := reachable AND occurrence;
12:             end for
13:             if reachable = false then
14:                 for all subsystems Sys_{K'} with K'' ⊆ K' (and |K'| = d) do
15:                     for all q' ∈ Ext(q'', K') do
16:                         reach(Sys_{K'})[q')] := false;
17:                     end for
18:                 end for
19:             else add q'' to list-of-possible-substates;
20:             end if
21:         end for
22:     end for
23: end for
24: END Cross-Checking
```

5 Detecting Deadlocks

Deadlock-freedom is an important property in itself and in addition establishing safety properties can be reduced to establishing deadlock-freedom. In this section, we present a definition of some locally checkable predicate P' that implies (in the sense that was described in Section 3) local deadlock-freedom for interaction systems.

Definition 8. *(Minimal) local Deadlock*
Given an interaction system Sys and a global state q we say that a set of components $D \subseteq K$ is a ***local deadlock*** *in q if every interaction in which any of the components in D could (in its present local state) participate is blocked by another component in D. More formally:*
$\forall i \in D\ \forall \alpha \in Int : (ea(q_i) \cap \alpha \neq \emptyset) \Rightarrow (\exists j \in D\ j(\alpha) \not\subseteq ea(q_j))$.
Obviously if we reach a global state q such that some set $D \subseteq K$ is a local deadlock in q no component in D can ever again participate in any interaction.
$D \subseteq K$ is a minimal local deadlock in q if no proper subset of D is a local deadlock in q. A system Sys is ***locally deadlock-free*** *if no state q is reachable such that there is a local deadlock in q.*

Example 5. Let us consider a global state q, where
$q \downarrow \{Phil_1, Fork_1, Phil_2, Fork_2, Phil_3\} =$
$(wantsleft_{Phil_1}, occupied_{Fork_1}, wantsboth_{Phil_2}, occupied_{Fork_2}, wantsright_{Phil_3})$. In this case, $D = \{Phil_1, Fork_1, Phil_2, Fork_2, Phil_3\}$ is a minimal local deadlock in q. (However, no such q is reachable in any of the systems $DP(p)$.)

Definition 9. *Small vs. Large Deadlocks*
When we compute the subsystem reachabilities as described in Section 3 we choose a value for the parameter d. Henceforth we will call local deadlocks D with $|D| \leq d$ ***small*** *local deadlocks and local deadlocks D with $|D| > d$* ***large*** *local deadlocks.*

In order to prove for a system *Sys* the predicate $P =$ "Local Deadlock-Freedom" it is sufficient to prove for some fixed $d << n$ that there are neither small nor minimal large local deadlocks reachable in *Sys*. When we traverse the reachable substates in the various $Reach'(Sys_{K'})$ we will be able to identify deadlocks of size $|D| \leq d$ directly, whereas the existence of deadlocks of size $|D| > d$ will have to be excluded by a sufficient condition. In the following subsections, we describe a locally checkable (i.e. checkable in the subsystems) P' that - when true on all substates in all $Reach'(Sys_{K'})$ - ensures the validity of P.

5.1 Defining and Checking a Condition for Small Deadlocks

To deal with the question of small local deadlocks it is sufficient to prove that there are no deadlocks of size $\leq d$ in the substates that are marked reachable in the reachability tables of the investigated subsystems. Again, it is infeasible to check all 2^d subsets of every substate of every subsystem, because this would yield up to $2^d \cdot \binom{n}{d} \cdot m^d$ loop cycles in the first place. Instead, we will directly check the substates of size 1 to $d-1$ that have been added to *list-of-possible-substates* in Algorithm 1.

5.2 Defining a Condition for Large Deadlocks

Obviously, we can not directly identify a large local deadlock in a subsystem with d components. Instead we are going to check a condition which is sufficient

for deadlock-freedom. In order to formulate this condition we first introduce the following relation between the local states of the components' transition systems.

Definition 10. *"q_i waits for q_j", ($q_i \in Q_i, q_j \in Q_j$)*
We say q_i waits for q_j if $\exists \alpha \in Int$, s.t. $\alpha \cap ea(q_i) \neq \emptyset \ \wedge j(\alpha) \not\subseteq ea(q_j)$.

I.e. q_i waits for q_j if j might prevent i from participating in an interaction in a corresponding global state q. If components prevent each other from participating in interactions then they might be in deadlock. The following definition assigns to a global state q (resp. a substate q') a directed graph based on the relation introduced in Definition 10.

Definition 11. *Wait-for-graph*
For a system Sys and a global state q we define the wait-for-graph $WFG(q) = (V, E)$ by:
$V = \{q_i \mid 1 \leq i \leq n\}$ and $E = \{(q_i, q_j) \in V \times V \mid q_i$ waits for $q_j\}$.
For $K' \subseteq K$ and a corresponding substate $q' = q \downarrow K'$ we denote by $WFG(q')$ the subgraph of $WFG(q)$ generated by $V' = \{q_i \in V \mid i \in K'\}$.

Given a large deadlock $D \subseteq K$ in a reachable global state q we will be able to detect (in at least one subsystem) the following pattern.

Theorem 2. *If Sys has a large minimal deadlock D in a global state q, then there is a subset $K' \subseteq D$ with $|K'| = d$ and a linear order $(i_1, \ldots, i_d)$ of the components in K' such that*
$k < l \Rightarrow q_{i_l}$ is reachable from q_{i_k} in $WFG(q \downarrow K')$.

Example 6. When we apply P' (for $DP(6)$ with $d = 4$) to the reachable state spaces $Reach(Sys_{K'})$ that we computed in the first place, we will detect 1584 (of reachable 185883) substates for which P' is not valid.

Applying P' (for $DP(6)$ with $d = 4$) to the reachable state spaces $Reach'(Sys_{K'})$ that we gain after applying cross-checking, the number of substates for which P' is not valid decreases to 432 (of reachable 81534).

These numbers induce that among the substates whose reachability was refuted via cross-checking there are indeed critical ones. Even more the percentage of reachable substates that are critical has decreased. This is due to a tendency in our approach to leave uncritical substates marked reachable.

5.3 Complexity of Checking Our Condition for Large Deadlocks

According to Section 3 and Theorem 2 we may prove that a system *Sys* does not contain any reachable minimal large deadlocks by proving that for neither of the subsystems $Sys_{K'}$ (with $K' \subseteq K$ and $|K'| = d$) and their substates q' in $Reach'(Sys_{K'})$ there is a linear order as described above for the nodes in $WFG(q')$.

To do so, we first construct, for every subsystem with d components and every therein reachable substate q', the graph $WFG(q')$. Then we could apply the following procedure **Order** which finds a linear order as described in Theorem 2, if there is any.

Preprocessing: Constructing the wait-for-relations
As a preprocessing we can (in $O((n{\cdot}m)^2)$) compute a $(n{\cdot}m{\times}n{\cdot}m)$-matrix W with $W(q_i, q_j) = 1$ if q_i waits for q_j, $W(q_i, q_j) = 0$ otherwise. This matrix includes for every substate q' the information about $WFG(q')$. Thus for a substate q', we simply create the $d \times d$-matrix for $WFG(q')$ and fill it by copying the relevant information from our matrix W. This can be done in $O(d^2)$.
Procedure Order:
Perform breadth-first search for every local state in $WFG(q')$. If there is one state q_j from which all other states can be reached make $i_1 := j$. Now find a state from which all remaining (not yet ordered) states are reachable and so on. Whenever such a state cannot be found, abort. Return the order when all components are ordered.
Proof of Correctness:
It is obvious that if the Procedure Order is not aborted then a returned order suffices our requirements. We show that if there is a linear order as described in Theorem 2, then Procedure Order will find one.

Note that if there is a linear order of the components in K' as described in Theorem 2 then this also holds for every subset of K' (w.r.t. the graph $WFG(q')$). This means in every step of Procedure Order we can choose the next component for the linear order and it is always guaranteed that the linear order so far can be enhanced (by a linear order of the remaining components) to a correct linear order for all d components.
Actual Implementation and Complexity:
The description of Procedure Order above will not be implemented directly but rather acts as a makeshift for ease of understanding and for our proof of correctness. For our implementation, we first compute the transitive closure of $WFG(q')$. This is possible in $O(d^3)$. Thus, instead of performing up to d breadth-first-searches we can simply determine the next component in our linear order (d times) by examining for each of the d components, if the remaining (not yet ordered) components are reachable from it (a comparison which can be done in $O(d)$ using the transitive closure).

So the Procedure Order can be performed in an overall time in $O(d^3)$.

The overall complexity of our check for large deadlocks is thus bounded by $O(m^d \cdot n^d \cdot d^3)$.

5.4 Connected Subsystems

Definition 12. *Interaction Graph & Connected Systems*
For an interaction system Sys we define the ***interaction graph*** $IG = (V, E)$ *by:* $V = K$ *and* $E = \{\{i, j\} \mid \exists \alpha \in Int\,(\alpha \cap A_i \neq \emptyset \wedge \alpha \cap A_j \neq \emptyset)\}$
We call an interaction system ***connected*** *if its interaction graph is connected.*

Note that for the purpose of deadlock detection we may restrict our attention to connected systems *Sys* (as for an unconnected system it is equivalent to prove its interaction graph's connected components deadlock-free). However if the original interaction system is connected, we may restrict all our observations

(i.e. reachability analyses, cross-checking and the checks for small resp. large deadlocks) - without loss of correctness or even loss of information - to connected subsystems.

Example 7. For Tanenbaum's dining philosphers as modelled here, the maximum degree of a node in the interaction graph is 9. This makes it easy to derive that the maximum number of *connected* subsystems is bounded by $n \cdot 9^{d-1} = O(n)$ for a fixed choice of d.

6 Related Work

Many important approaches have been developed in the past to tackle the problem of state space explosion. A wide spectrum of methods for approximation and/or reduction of the state space, ranging from partial order reduction, exploiting equivalences to abstraction/refinement techniques have been investigated and e.g. incorporated in model checking tools and abstract interpretation approaches. Our approach to establish properties of component based systems is in a certain sense complementary but can nevertheless be put into comparison with some existing techniques. The basic principles of our approach can be summarized as follows.

1) We exploit the knowledge about the interaction structure of the system, i.e. the interaction graph. For this we determine the connected subgraphs with d nodes in this graph ($d << n$ a constant).
2) Then we calculate the reachable state spaces for the subsystems belonging to these subgraphs.
3) We apply cross-checking to delete "artefacts" within these subsystems.
4) We establish a condition that is to be checked on the subsystems and when satisfied guarantees a global property.
5) All steps can be performed in polynomial time (bounded by a polynomial with degree d).

Step 2) can be seen as a locality based abstraction in the sense of [EGS05] which is the most general paper on locality-based abstraction we know of. When we consider a subsystem with d components then this corresponds to an observer that has access to these d components of the system. (If each observer has access to exactly one component, the special case of Cartesian abstraction [BPR01, Arn94] arises.) However a closer look to the notion of partial transition relation in locality based abstractions of [EGS05] and hence the notion of local reachability, shows that our approach has to be distinguished from theirs. In [EGS05] one condition for a partial state p_1 to evolve to state p_2, i.e. $\underline{t}(p_1, p_2)$, is that p_1 matches some state p with respect to the kernel of the transition t.(Please note that each transition relation t of [EGS05] corresponds to an interaction α of our model.)

In one of our local systems given by the components $\{i_1, ...i_d\}$ a local transition can take place in a sub-state if all partners of an interaction α that are part of the subsystem offer their part of the interaction, i.e. we then perform $\alpha \cap \bigcup_{1 \leq i \leq d} A_{i_d}$.

As we may obtain artefacts by proceeding in such a way, called loss of information in [EGS05], we then apply cross-checking in step 3) to eliminate as many artefacts as possible by comparing the subsystems. To the best of our knowledge this technique has not been applied in the context of state space investigations. Cartesian abstraction [BPR01, Arn94] which involves finding the smallest Cartesian product that contains a given set has a similar purpose for the case $d = 1$. Cross-checking is however closely related to techniques employed in the relational model of data bases. Speaking in data base terminology we decompose an initial data base scheme into sub-schemes (corresponding to our subsystems). Applying the join operation $\bowtie$ to all these subsystems, i.e. $Reach(Sys_{K'_1}) \bowtie Reach(Sys_{K'_2}) \bowtie ... \bowtie Reach(Sys_{K'_k})$ (where $k = \binom{n}{d}$) would yield the best over-approximation one can get when using locality based abstraction if no further knowledge is available. However calculating these joins leaves us with the same complexity issue as calculating the reachable global state space. So we avoid evaluating this sequence of joins and perform instead the comparison of pairs of subsystems.

Concerning step 4), the conditions on subsystems that guarantee global properties, the closest work to ours is by [AC05] on deadlock-freedom who base their work on concurrent programs but employ a similar notion of subsystems and local transition relation. They consider subsystems of size 3 but apply no technique comparable no cross checking.

7 Implementation

All presented techniques have been implemented in our tool "PrInSESSA" [MS08] where in addition we also apply a variant of cross-checking to the detection of minimal large deadlocks.

In order to allow for a quantitative comparison with other tools, a BDD-based framework is presently being implemented. First benchmarks show that the BDD-based variant can prove instances as large as $DP(500)$ deadlock-free within minutes.

8 Conclusion and Further Work

We presented a method to obtain an enhanced over-approximation of the global state space of a component-based system with n components in polynomial time. The method consists of choosing a value for d, investigating subsystems consisting of d components in a first step (in $O(n^d \cdot m^d)$) and then improving this approximation by cross-checking (in $O(d \cdot n^d \cdot m^d)$). The computation of the first step can be improved in various respects. Firstly, for the purpose of proving deadlock-freedom we do not have to consider all $\binom{n}{d}$ subsystems but only such that are connected. Secondly, the computation of the various sets $Reach(Sys_{K'})$ can be performed in parallel. Our approximation can be used to investigate global properties by considering subsystems and checking conditions on them which requires only polynomial cost. We showed how this can be achieved for

the property of local deadlock-freedom. Interesting open theoretical questions are e.g. how many iterations are needed at most until the iterative application of cross-checking reaches a fixpoint and in which complexity class the exact computation of the set $X_{K'}$ defined in Section 4 lies.

References

[AC05] Attie, P., Chockler, H.: Efficiently Verifiable Conditions for Deadlock-Freedom of Large Concurrent Programs. In: Cousot, R. (ed.) VMCAI 2005. LNCS, vol. 3385, pp. 465–481. Springer, Heidelberg (2005)

[AG97] Allen, R., Garlan, D.: A Formal Basis for Architectural Connection. ACM Trans. Softw. Eng. Methodol. 6(3), 213–249 (1997)

[Arn94] Arnold, A.: Finite transition systems: semantics of communicating systems (Translator-John Plaice). Prentice Hall International (UK) Ltd., Hertfordshire (1994), Translator-John Plaice

[ASCN99] George, A.S., Clarke, L.A., Naumovich, G.: Data flow analysis for checking properties of concurrent java programs. In: ICSE 1999: Proceedings of the 21st international conference on Software engineering, pp. 399–410. ACM, New York (1999)

[BBS06] Basu, A., Bozga, M., Sifakis, J.: Modeling Heterogeneous Real-time Components in BIP. In: Proceedings of SEFM 2006, pp. 3–12. IEEE Computer Society, Los Alamitos (2006)

[BCD02] Bernardo, M., Ciancarini, P., Donatiello, L.: Architecting Families of Software Systems with Process Algebras. ACM Trans. on Software Engineering and Methodology 11, 386–426 (2002)

[BPR01] Ball, T., Podelski, A., Rajamani, S.K.: Boolean and cartesian abstraction for model checking c programs. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 268–283. Springer, Heidelberg (2001)

[dAH01] de Alfaro, L., Henzinger, T.: Interface Automata. In: FSE 2001, pp. 109–120 (2001)

[EGS05] Esparza, J., Ganty, P., Schwoon, S.: Locality-based abstractions. In: Hankin, C., Siveroni, I. (eds.) SAS 2005. LNCS, vol. 3672, pp. 118–134. Springer, Heidelberg (2005)

[GGM+07a] Goessler, G., Graf, S., Majster-Cederbaum, M., Martens, M., Sifakis, J.: An Approach to Modelling and Verification of Component Based Systems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 295–308. Springer, Heidelberg (2007)

[GGM+07b] Goessler, G., Graf, S., Majster-Cederbaum, M., Martens, M., Sifakis, J.: Ensuring Properties of Interaction Systems by Construction. In: Reps, T., Sagiv, M., Bauer, J. (eds.) Wilhelm Festschrift. LNCS, vol. 4444, pp. 201–224. Springer, Heidelberg (2007)

[Goe06] Goessler, G.: Component-based Design of Heterogeneous Reactive Systems in Prometheus. Technical Report 6057 INRIA (2006)

[GS03] Goessler, G., Sifakis, J.: Component-based Construction of Deadlock-free Systems. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 420–433. Springer, Heidelberg (2003)

[GS05] Goessler, G., Sifakis, J.: Composition for Component-based Modeling. Sci. Comput. Program. 55(1-3), 161–183 (2005)

[Kov] Kovalyov, A.V.:

[LT89] Lynch, N.A., Tuttle, M.R.: An Introduction to Input/Output Automata. In: CWI-Quarterly, pp. 219–246 (1989)

[MM08a] Majster-Cederbaum, M., Martens, M.: Compositional Analysis of Tree-Like Component Architectures. In: Proceedings of EMSOFT 2008 (2008)

[MM08b] Majster-Cederbaum, M., Minnameier, C.: Everything is PSPACE-Complete in Interaction Systems. In: Fitzgerald, J.S., Haxthausen, A.E., Yenigun, H. (eds.) ICTAC 2008. LNCS, vol. 5160, pp. 216–227. Springer, Heidelberg (2008)

[MMM07a] Majster-Cederbaum, M., Martens, M., Minnameier, C.: A Polynomial-time Checkable Sufficient Condition for Deadlock-Freedom of Component-based Systems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 888–899. Springer, Heidelberg (2007)

[MMM07b] Majster-Cederbaum, M., Martens, M., Minnameier, C.: Liveness in Interaction Systems. In: Proceedings of FACS 2007. ENTCS (2007)

[MS08] Minnameier, C., Schaube, R.: PrInSESSA - Proving Properties of Interaction Systems by Enhanced State Space Approximation (2008), `http://134.155.88.3/main/chair_de/03/cmm_cross_checking/index_de.html`

[Sif04] Sifakis, J.: Modeling Real-time Systems. In: Keynote talk RTSS 2004 (2004)

[Sif05] Sifakis, J.: A Framework for Component-based Construction. In: Proceedings of the Third IEEE International Conference on Software Engineering and Formal Methods, pp. 293–300. IEEE Computer Society, Los Alamitos (2005)

[Tan08] Tanenbaum, A.: Modern Operating Systems, 3rd edn (2008)