Complexity Results for Reachability in Cooperating Systems and Approximated Reachability by Abstract Over-Approximations

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Nils Semmelrock
aus Hamburg

Mannheim, 2013

Dekan:	Professor Dr. Heinz Jürgen Müller, Universität Mannheim
Referent:	Professor Dr. Mila Majster-Cederbaum, Universität Mannheim
Korreferent:	Professor Dr. Matthias Krause, Universität Mannheim

Tag der mündlichen Prüfung: 11. Juni 2013

Abstract

This work deals with theoretic aspects of cooperating systems, i.e., systems that consists of cooperating subsystems. Our main focus lies on the complexity theoretic classification of deciding the reachability problem and on efficiently establishing deadlock-freedom in models of cooperating systems. The formal verification of system properties is an active field of research, first attempts of which go back to the late 60's. The behavior of cooperating systems suffers from the state space explosion problem and can become very large. This is, techniques that are based on an analysis of the reachable state space have a runtime exponential in the number of subsystems. The consequence is that even modern techniques that decide whether or not a system property holds in a system can become unfeasible.

We use interaction systems, introduced by Sifakis et al. in 2003 [GS03], as a formalism to model cooperating systems. The reachability problem and deciding deadlock-freedom in interaction systems was proved to be PSPACE-complete [MCM08c]. An approach to deal with this issue is to investigate subclasses of systems in which these problems can be treated efficiently. We show here that the reachability problem remains PSPACE-complete in subclasses of interaction systems with a restricted communication structure. We consider structures that from trees, stars and linear arrangements of subsystems. Our result motivates the research of techniques that treat the reachability problem in these subclasses based on sufficient conditions which exploit characteristics of the structural restrictions [Mar09, Hoa85, BR91, BHH$^+$06, BCD02, MCM08a].

In a second part of this work we investigate an approach to efficiently establish the reachability of states and deadlock-freedom in general interaction systems. We introduce abstract over-approximations – a concept of compact representations of over-approximations of the reachable behavior of interaction systems. Families of abstract over-approximations are the basis for our approach to establish deadlock-freedom in interaction systems in polynomial time in the size of the underlying interaction system. We introduce an

operator called Edge-Match for refining abstract over-approximations. The strength of our approach is illustrated on various parametrized instances of interaction systems. Furthermore, we establish a link between our refinement approach and the field of relational database theory and use this link in order to make a preciseness statement about our refinement approach.

Zusammenfassung

Diese Arbeit beschäftigt sich mit theoretischen Aspekten von kooperierenden Systemen, d.h. Systemen, die aus kooperierenden Subsystemen bestehen. Unser Augenmerk liegt hauptsächlich auf der komplexitätstheoretischen Klassifizierung des Erreichbarkeitsproblem und dem effizienten Nachweis von Verklemmungsfreiheit in Modellen von kooperierenden Systemen. Die formale Verifikation von Systemeigenschaften ist ein aktives Forschungsfeld dessen Anfänge in die späten sechziger Jahre zurückreichen. Kooperierende Systeme leiden unter dem Problem der Zustandsraumexplosion und können ein sehr komplexes Verhalten besitzen. Techniken, die auf der Analyse des erreichbaren Zustandsraumes basieren weisen hier eine Laufzeit auf, die exponentiell in der Anzahl der Subsysteme ist. Die Konsequenz ist, dass selbst aktuelle Techniken, die Systemeigenschaften entscheiden, an ihre Grenzen geraten.

Wir benutzen den von Sifakis et al. 2003 [GS03] eingeführten Formalismus der Interaktionssysteme um kooperierende Systeme zu modellieren. Das Erreichbarkeitsprobelm und das Problem der Verklemmungsfreiheit in Interaktionssystemen ist PSPACE-vollständig [MCM08c]. Ein Ansatz dieses Problem anzugehen ist die Betrachtung von Teilklassen, in denen diese Probleme effizient behandelt werden können. Wir zeigen hier, dass das Erreichbarkeitsproblem auch in Teilklassen mit eingeschränkter Kommunikationsstruktur PSPACE-vollständig ist. Wir betrachten Strukturen, die Bäume, Sterne und lineare Anordnungen aus Subsystemen darstellen. Unsere Ergebnisse motivieren die Untersuchung von Techniken die das Erreichbarkeitsproblem in diesen Teilklassen, basierend auf hinreichenden Bedingungen welche die strukturellen Charakteristiken ausnutzen, behandeln [Mar09, Hoa85, BR91, BHH+06, BCD02, MCM08a].

In einem zweiten Teil dieser Arbeit stellen wir einen Ansatz vor, der es ermöglicht die Erreichbarkeit von Zuständen und Verklemmungsfreiheit in Interaktionssystemen festzustellen. Dafür führen wir abstrakte Überapproximationen ein. Dies sind kompakte Repräsentationen von Überapproximationen des erreichbaren Verhaltens von Interaktionssystemen. Familien von

abstrakten Überapproximationen sind die Basis für unseren Ansatz in polynomieller Zeit Verklemmungsfreiheit in Interaktionssystemen festzustellen. Wir benutzen einen Operator, den wir Edge-Match nennen, um abstrakte Überapproximationen zu verfeinern. Die Stärke unserer Ansätze demonstrieren wir anhand von verschiedenen parametrisierten Modellen von Interaktionssystemen. Darüber hinaus ziehen wir eine Verbindung zwischen unserem Ansatz der Verfeinerung von abstrakten Überapproximationen und dem Gebiet der relationalen Datenbanktheorie. Wir benutzen diese Verbindung um eine Aussage über die Güte unseres Verfeinerungsansatzes zu machen.

Contents

Acknowledgements

First, I am grateful to Professor Mila Majster-Cederbaum for introducing me to this new area of research and for being a great supervisor. Apart from her helpful technical guidance, I thank her for her kind and supportive nature. She was always approachable, provided helpful comments and ideas for further research and still allowed for unsupervised phases whenever I felt I needed to work things out on my own.

I also thank the second referee Professor Matthias Krause and the third examiner Professor Frederik Armknecht for taking time to review my thesis.

Thanks to my former and current fellow workers Christoph, Moritz and Christian for making my time very enjoyable. The same goes to all other fellow scientists I met at the faculty.

Finally, I thank my family for just being there for me.

Glossary of Symbols

Interaction Systems

A_i The set of ports of a component i. 23

α An interaction. 23

a_i A port of a component i. 23

$\mathbb{C}$ A domain. 82

$en(q)$ The operator that yields all labels that are enabled in the state q of a transition system. 26

$E(R)$ The global extension of the transition system R. 75

$i(\alpha)$ The port of component i that participates in the interaction α. 23

IM An interaction model. 23

Int A set of interactions. 23

$\mathrm{IS}_{\mathrm{general}}$ The class of general interaction systems. 35

$\mathrm{IS}_{\mathrm{line}}$ The subclass of linear interaction systems. 35

$\mathrm{IS}_{\mathrm{star}}$ The subclass of star-like interaction systems. 35

$\mathrm{IS}_{\mathrm{tree}}$ The subclass of tree-like interaction systems. 35

K A set of components. 23

π A path in a transition system. 101

Q_C Cartesian product of the local state spaces of the components in C. 74

$q{\downarrow_C}$ The projection of the global state q on the components in C. 74

q_C^0 Tuple that consists of the local initial states of the components in C. 74

$q_C{\downarrow_D}$ The projection of the state q_C on the components in D. 81

Q_i The state space of the local behavior of a component i. 27

q_i^0 The initial state of the local behavior of a component i. 27

S_C An abstract over-approximation that is constructed as in Lemma 3.2. 77

Sys An interaction system. 27

T_i Transition system that models the local behavior of a component i. 27

$\rightarrow_i$ The transition relation of the local behavior of a component i. 27

Relational Algebra

A An attribute. 165

D_i The domain of the attribute A_i. 165

$\mathrm{dom}(\mathsf{A}_i)$ Yields the domain D_i of the attribute A_i. 165

$\mathsf{D}(\mathsf{R})$ Union of the domains of all attributes of the relational scheme R. 165

R A relational scheme. 165

$\mathsf{r}(\mathsf{R})$ A relation on the relational scheme R. 165

t A tuple. 165

Misc

2^K The powerset of the set K. 82

AP A set of atomic propositions. 102

b The blanc symbol. 55

δ The transition function of a Turing machine. 55

Γ A set of tape symbols. 55

∞ The operator that, applied on a set, yields all finite and infinite concatenations of the elements in the set. 102

L_S A labeling functions that assigns atomic propositions to the states of the transition system S. 102

M A Turing machine. 55

$\models$ A binary relation on transition systems and system properties that states that a transition system satisfies a property. 102

P A system property. 100

$\mathbb{P}$ The state space of a Turing machine. 55

p^0 The initial state of a Turing machine. 55

$Path(S)$ The set of all finite and infinite paths of the transition system S. 102

$Path_{fin}(S)$ The set of all finite paths of the transition system S. 102

p^N A halting state of a Turing machine. 55

p^Y A halting state of a Turing machine. 55

$RIST$ The set of instances of the reachability problem in tree-like interaction systems. 42

RT Operator that yields all reachable transitions of a transition system. 74

Σ A set of input symbols. 55

$\mathbb{T}$ A variable for the movement of the tape head of a Turing machine ($\mathbb{T} \in \{-1, 1\}$). 57

$TQBF$ The set of true QBF instances. 40

$trace(\pi)$ The trace of the path π. 102

$Traces(S)$ The set of all infinite and maximal finite traces of the transition system S. 102

$Traces_{fin}(S)$ The set of all finite traces of the transition system S. 102

$TRIST$ The set of all tuples consisting of a tree-like interaction systems and a reachable global state. 42

Chapter 1

Introduction

1.1 Motivation

This work deals with a complexity theoretic classification of deciding certain system properties in subclasses of cooperating systems and introduces an approach for establishing deadlock-freedom in cooperating systems. In the following we put this work into context and motivate the relevance of our results. [WRS+08]

1.1.1 Formal Verification

> "Complete formal verification is the only known way to guarantee that a system is free of programming errors."[1]

Formal verification of systems refers in the widest sense to techniques that show or refute desired behavior of systems by formal methods. From today's perspective, this means a formal verification technique shows or refutes that

[1] Opening of the paper [KEH+09] from Klein et al. in which they discuss the formal verification of the *slE4 microkernel*.

a formal model of a system meets desired properties by using formal methods of mathematics.

System properties describe that the behavior of a system should satisfy certain requirements, e.g., a certain situation must or must not occur or that it is always possible to evoke a certain situation. Particularly systems that operate in environments where a system failure can cause harm to people or result in huge financial losses are crucially required to satisfy certain properties. For example, an error in the control system of a nuclear power plant or in a guidance system on a plane can become fatal, a flawed central processing unit in large-scale production can result in an expensive recall campaign. An, in general, unwanted situation in a system is a situation where the system gets stuck and is unable to continue its desired behavior. For example, an operation system that crashes is at least unpleasant and a control system of a nuclear power plant that is not able to react to a critical incident can lead to a dangerous situation. A system state like that is called a deadlock and a system that can not get into a deadlock state is called deadlock-free. The system property of deadlock-freedom is given a particular significance as the problem of verifying safety properties (an important class of system properties that we discuss in detail in Chapter 4) in cooperating systems can be reduced to the detection of deadlocks [GW92]. The authors of [GW92] introduce a technique to detect deadlocks in cooperating systems and show how the technique can be used to verify an arbitrary safety property.

One way to describe system properties is based on temporal logic such as LTL [Pnu77] and CTL [CE81] or the more general modal logic μ-calculus [Koz82]. Another approach that differs from a description by a modal logic is based on equivalence relations on the behavior of systems like bisimulation or observational equivalence as discussed in [Mil89]. The idea is to describe a system and a system property in the same formalism. If the descriptions are equivalent then the system fulfills the property. Techniques that automatically check equivalences are for example published in [KS83, PT87, CS01]. In the following we take a closer look at the temporal logics LTL and CTL

and techniques that automatically check whether a system fulfills a property described in these logics.

Prior [Pri57] introduced in 1957 a modal logic with the operators F ("eventually") and P ("previously") that states that a proposition eventually becomes true in the future respectively a proposition was true in the past. Thus, this first attempt assumes that time is linear. In correspondence with Kripke, who mentioned that a linear view of time might not always be enough, Prior developed two branching time logics that are suited to express that a proposition becomes true in all or at least one possible future. For example "only one process of the system will enter it's critical section at one point in time", "a customer can only withdraw money from the ATM if the pin was entered correctly", "if the reactor becomes to hot, it will shut down eventually" or "if a user is logged into the ATM, the user has always the possibility to logout".

In 1977 Pnueli introduced linear temporal logic (LTL) a temporal logic for specifying system properties [Pnu77, Pnu79]. The idea is that a system that is executed runs through a (possible infinite) sequence of states where in each state atomic propositions hold, i.e., an execution induces a sequence of sets of atomic propositions. An LTL formula specifies a set of permitted sequences of sets of atomic propositions and a system models an LTL formula if the induced sequence of atomic system properties of each possible execution is permitted. "a customer can only withdraw money from the ATM if the pin was entered correctly" is an example for a property that can be specified by LTL.

In 1981 Clarke and Emerson introduced computational tree logic (CTL) a branching time logic [CE81]. In contrast to LTL, a CTL formula does not deal with sequences of sets of atomic propositions but with trees the nodes of which are sets of atomic propositions. A system induces a (possible infinite) computational tree, i.e., a tree that describes all possible executions of the system. A system fulfills a CTL formula if the induced computational tree is a tree that is described by the formula. "If a user is logged into the ATM,

the user has always the possibility to logout" is an example for a system property that can be described by CTL as this property specifies that for each system state where a user is logged into the ATM there is a sequence of actions that leads to a state where the user can logout.

LTL and CTL are suited to express important system properties. Especially the system property of deadlock-freedom can be described by LTL and CTL.

Among the first attempts of formal verification is the deductive program verification which started with the *Floyd-Hoare logic* where pre- and postconditions are assigned to commands in computer programs. A set of inference rules is used in order to deduce pre- and postconditions of larger code fragments. This work was published in [Hoa69] and was influenced by [Flo67] where a similar approach is described for flowcharts. The introduced approach is suited to prove partial correctness of programs, i.e., for a given precondition this approach can be used to show that a program meets desired postconditions but this approach does not prove termination of the program.

Techniques that automatically check whether a system models a CTL respectively LTL formula are subsumed under the term CTL respectively LTL *model checking*. The input of a CTL or LTL model checking technique is the description of a system and an LTL respectively CTL formula. The costs of a CTL or LTL model checking technique are given in relation to the input size, i.e., if the system is given by a labeled transition systems then the size of the system corresponds to the number of contained states and transitions and the size of an LTL or CTL formula corresponds to the number of contained subformulas. In addition to introducing CTL, Clarke and Emerson provided in [CE81] a technique that automatically checks whether or not a system, given as a transition system, models a CTL formula. Quielle and Sifakis described the same (for a temporal logic similar to CTL) [QS82]. The technique in [CE81] is polynomial in the size of the transition system and the size of the CTL formula (the number of subformulas). This bound

is improved in [CES86] by an algorithm that decides whether or not a CTL formula in a transition system holds in time linear in the size of the system and the formula, i.e., for a transition system S and a CTL formula Φ the technique runs in $O(|S| \cdot |\Phi|)$ where $|S|$ denotes the size of S and $|\Phi|$ the size of Φ. In [VW86] Vardi and Wolper introduced a technique for automatic checking whether an LTL formula is modeled by a system given as a transition system. The technique by Vardi and Wolper is linear in the size of the transition system but exponential in the size of the LTL formula, i.e., for a transition system S and an LTL formula φ the technique runs in $O(|S| \cdot 2^{|\varphi|})$. See [Sch02] for an overview of the complexity of temporal logic model checking. Algorithms that implement LTL and CTL model checking exhibit these runtime and are in use in popular model checking tools – see for example *Spin* for LTL model checking [Hol97]. Thus, algorithms in use today are linear in the number of states and transitions of the transition systems in consideration.

Burch, Clarke, et al. described in [BCM+92] how a symbolic representation of labeled transition systems by binary decision diagrams (BDDs) [Bry86] allows for model checking large systems in a reasonable time. As the title of [BCM+92] states, this approach permits the verification of system properties in systems with 10^{20} states and beyond. This number was improved in the subsequent years, e.g., in [BCL91] Burch et al. reported about a technique based on symbolic model checking that manages systems with 10^{120} states. Note that these numbers refer to computer performance at respective times. Although, symbolic model checking is superior to model checking on systems that are represented explicitly, this technique does not improve the asymptotic bounds of CTL and LTL model checking algorithms. In [BCCZ99] Clarke et al. introduced bounded model checking, a technique where LTL model checking is reduced symbolically to the satisfiability problem of propositional logic formulas.

1.1.2 Model Checking Cooperating Systems

A cooperating system is a system that consists of subsystems which work together, i.e., a cooperating system is specified by its subsystems and by a description of how these subsystems interact among each other – this description is called the "glue-code" of the system. Compared to the size of the subsystems and the glue-code, the global behavior that results from such a specification can become quite complex and thus hard to analyze. This problem is called the *state space explosion problem* [BK08], i.e., the phenomenon that the state space of a system grows exponentially in the number of subsystems that work in parallel. The subsystems of a cooperating system operate in parallel where each subsystem features its own state space, i.e., the state space of a cooperating system consists of the Cartesian product of the state spaces of the subsystems. The glue-code connects the subsystem by specifying a dependency between the state transitions of the subsystems, i.e., the state transition that is offered by a certain subsystem is depending on other subsystems being in particular states. Thus, the glue-code restricts the reachable state space of a cooperating system. Nevertheless, the size of the state space is exponential in the number of subsystems and the size of the reachable state space might be as well.

There are several formalisms that model cooperating systems including Petri nets [Pet67] or UML [RJB99] (graphical languages), CSP [Hoa85] or CCS [Mil82] (process algebras), Linda [ACG86] or Reo [Arb04] (coordination languages) or even actual programming language as Java or C/C++ and the formalism of interaction systems [GS03] that we introduce in Section 1.4 of this chapter and use in the remainder to model cooperating systems. Further models, that are similar to the formalism of interaction systems, are described in Section 1.4.1.

A cooperating system satisfies a system property that is given by an LTL or CTL formula if and only if the global behavior of the system satisfies the formula. LTL and CTL model checking algorithms are linear in the

size of the system in consideration, i.e., model checking a cooperating system by those algorithms requires time exponential in the number of subsystems. This, observation is backed up by [BVW94] (extended abstract of [KVW00]) and [Kup95] where it was proven that LTL and CTL model checking for *concurrent programs* [Pnu79], a formalism for modeling cooperating systems, is PSPACE-complete in the size of the input system and the respective temporal logic formula (see [VU98] for a summary of complexity theoretic classifications regarding LTL and CTL model checking). The authors even strengthened this statement by showing that the same complexity theoretic classification holds if the temporal logic formula is considered to be arbitrary but fixed, i.e., the PSPACE-completeness result does not depend on the LTL or CTL formula. Note that these results do not imply the PSPACE-completeness of the problem to model check a particular system property described in LTL or CTL in a cooperating system. However, the complexity theoretic classification of the problem to check whether or not a particular system property holds in a system has been researched for various formalisms that model cooperating systems and various important system properties. For example, the PSPACE-completeness of deciding reachability, deadlock-freedom and liveness in 1-safe Petri nets was shown in [CEP95]. The PSPACE-completeness of deciding reachability, deadlock-freedom, progress and availability in interaction systems was shown in [MCM08c]. See [AKY99] for results in *communicating finite state machines* and [CEP95, EN94] for results in various subclasses of Petri nets.

Even though LTL and CTL model checking allow for an automatic checking of important system properties in complex systems, there are still systems that are relevant in real life where it is not feasible to check certain system properties because of the sheer size of the reachable state space. This is, checking whether a property holds in a cooperating system can become unfeasible if the number of subsystems or the size of individual subsystems increases. Symbolic model checking [BCM$^+$92] allows for dealing with systems with 10^{20} states and beyond, e.g., cooperating systems that consists

of 20 subsystems with each 10 states. The improved approach by Burch et al. [BCL91] can deal with systems with 10^{120} states, e.g., 120 subsystems with each 10 states. In [HKW12] the authors analyze the control software of the *CERN Compact Muon Solenoid experiment* which consists of over $30,000$ cooperating *finite state machines*, i.e., a relevant system that is far too large for a direct application of model checking techniques. The authors only consider systems properties that can be verified by considering subsystems consisting of reasonable numbers of state machines. In Chapter 4 we introduce an approach, based on a sufficient condition, that establishes deadlock-freedom in cooperating systems in time polynomial in the size of the input system. In order to illustrate our results we introduce in Chapter 5 several parameterized examples the state space of which is considerable larger than 10^{120}, e.g., we consider a model consisting of $1,200$ subsystems with at least 2 states in each subsystem.

1.1.3 Dealing with Complexity Issues

If a system is very complex then a formal verification can become unfeasible because even a computer aided application of known techniques can require too much resources. This effect is supported by various complexity theoretic results regarding various formalisms, for modeling cooperating systems, and system properties which prove that a verification can not be achieved in polynomial time in the size of the input system. Approaches to circumvent this issue include the following items.

- Exploiting characteristics of *subclasses* of systems. Even if a complexity result states that we can not expect to decide whether or not a certain system property holds in polynomial time there might be interesting subclasses where this can be achieved.
- A modification of the input system such that known techniques need less resources. One approach is to reduce the size of the system in con-

sideration while preserving whether or not a system property holds in the modified system. Techniques that follow this approach are widely referred to as *state space reduction techniques*.

- Another approach is to consider verification techniques that are based on *sufficient conditions* and require less resources, i.e., if a technique like this succeeds then a property is guaranteed, if not then we cannot conclude whether or not the property holds.

These concepts are not mutually exclusive from each other, i.e., a technique can be based on several of these concepts. In the following we discuss techniques that are based on these approaches.

Subclasses

Decision problems that are complete in a complexity class that indicates that we can not expect that there is an algorithm that decides the problem in polynomial time might include "interesting" subclasses where the decision problem is decidable in polynomial time. A well known example is the Boolean satisfiability problem where 3SAT is NP-complete and 2SAT is decidable in polynomial time. The problem HORNSAT (the problem of deciding whether a given set of propositional Horn clauses is satisfiable) is even decidable in linear time. Similarly, the quantified 3SAT problem is PSPACE-complete, whereas the quantified 2SAT problem and the quantified HORNSAT problem [KBS88] is decidable in polynomial time. See [GJ79] for descriptions and more examples. In context of system properties for which it is hard (e.g., PSPACE-complete) to decide whether or not they hold in a system, the examples above rise the question whether there are interesting subclasses of systems where we can decide in polynomial time whether or not the system property holds.

If we show that a property P is decidable in polynomial time in a subclass then this subclass is interesting if the subclass consists of systems that

are relevant in practice. If, on the other hand, we show that deciding P is PSPACE-complete in a certain subclass then this subclass is interesting if it is as restricted as possible. This is because we can conclude that deciding P in each superclass is PSPACE-hard. Furthermore, the PSPACE-completeness of deciding P in a subclass justifies the research of techniques, based on sufficient conditions, that establish P in this class and can be tested in polynomial time (we discuss some of these approaches in the following).

An important approach in the design of cooperating systems is the so-called *correctness by construction* approach, i.e., the design of modeling rules that ensure that a system model fulfills certain properties. This can be generalized by providing a set of modeling rules that ensure that a certain system property can be decided or ensured efficiently in a cooperating system that is constructed by these rules. This is, a result that shows that a system property can be decided or ensured efficiently in a subclass of cooperating systems can be used to design correctness by construction techniques.

Many complexity results have been published for various subclasses of Petri nets. They show that various decision problems that are EXPSPACE-hard in general Petri nets become PSPACE-complete in interesting subclasses. In the same way various problems that are PSPACE-complete in general Petri nets become NP-complete in respective subclasses. [JLL77] considered reachability, liveness and boundedness in *free choice* Petri nets, *conflict free* Petri nets and *conservative* Petri nets and [HJR93] boundedness, reachability, containment and equivalence problems in *single path* Petri nets. A more recent results can be found in [PL08] where it was shown that the reachability problem is PSPACE-complete in Petri nets with fast growing markings (the best known lower bound in general Petri nets needs exponential space [Lip76]). However, in [Esp98] Esparza summarizes various results regarding various subclasses of Petri nets and sets up the following rule of thumb:

> "Many questions about marked graphs are solvable in polynomial time. Almost no questions about Petri net classes substantially

larger than marked graphs are solvable in polynomial time."

Marked graphs are a very basic subclass of Petri nets that is included in all above mentioned subclasses A marked graph is a Petri net where each place has exactly one incoming and one outgoing arc.

Various works deal with subclasses of cooperating systems that are defined by architectural constraints. For this a graph structure is defined which represents the communication structure among the subsystems. In this graph, the nodes are the subsystems and an undirected edge connects two subsystems if the glue-code specifies a cooperation between these subsystems. Based on this structure one can define subclasses of systems the communication structure of which forms, for example, a tree, a star or a linear arrangement of subsystems. Several works considered tree-like communication patterns and in particular established conditions that ensure deadlock-freedom. *Communicating Sequential Processes* are introduced in [Hoa85] where a directed communication structure based on input/output communication is considered. It is argued that communicating processes, if the directed input/output communication structure forms a rooted tree, can not deadlock. [BR91] describes a general communication graph for CSP models and provides conditions that guarantee deadlock-freedom in systems the communication graph of which forms a tree. [BCD02] examined a process algebra based on an architectural description language called *PADL* and considers deadlock-freedom in systems with a tree-like communication pattern (a proper superclass of systems with a star-like or linear pattern). The technique is based on a compatibility condition that is tested among pairs of cooperating subsystems, i.e., the composite behavior of two subsystems is weak bisimilar to the behavior of one of the components. An efficient technique based on a sufficient conditions for establishing deadlock-freedom in interaction systems with a star-like communication pattern is introduced in [Lam09] where, similar to [BCD02], a compatibility condition based on branching bisimilarity is tested. A sufficient condition for establishing deadlock-freedom for the subclass of tree-like interaction sys-

tems is described in [MCM08a] where a condition is tested on the reachable state spaces of pairs of interacting subsystems. In [LMC11] the condition in [MCM08a] is extended such that deadlock-freedom can be established in a proper superclass of tree-like interaction systems. Hennicker et al. proposed in [BHH+06, HJK10] a technique to construct so called *observable* behavior of a cooperating system with an acyclic communication pattern which can be used to establish certain system properties.

State Space Reduction

In order to circumvent the state space explosion problem in cooperating systems one can apply so-called state space reduction techniques in order to reduce the size of the reachable state space. These techniques include various methods that remove states or entire subsystems from a system or providing an associated system with a smaller state space. Given a system and a desired property, the idea is to construct a modified system such that

1. the state space of the modified system is small enough such that a technique can be applied to check whether the property holds in a reasonable amount of time and
2. if the property holds in the modified system then the property holds in the original system.

Three approaches which follow these requirements are techniques that exploit symmetries in a system, partial order reduction and abstract interpretation.

The idea behind exploiting the symmetry of a system is to compress the state space of a system by combining states into equivalence classes based on an equivalence relation. The equivalence relation is chosen such that paths in the original system correspond to paths in the compressed system and vice versa. The compression ratio and the properties that can be established by examining the compressed system is highly depending on the choice of the

equivalence relation. In [ID96] this idea is used to check whether a state is reachable in a system and for establishing deadlock-freedom. [ES96] uses this method on cooperating systems with identical or isomorphic subsystems and establishes system properties in CTL* in the compressed systems. [CJEF96] provides a state space reduction based on symmetries for a system which transition relation is given symbolically as a BDD and identifies a subclass of temporal logic formulas that is preserved by this reduction. [Jen96] uses this method to compress the state space of colored Petri nets.

Partial order reduction exploits that a state in a cooperating system might be reached by executing actions in different orders. The idea is that in the process of checking a suitable system property only one of these paths needs to be examined. [Pel93] introduces so-called model checking with representatives. An equivalence relation on the paths in a cooperating system is considered such that either all or none of the paths in a class model a formula in LTL. Based on this equivalence relation a labeled transition system is constructed such that for each equivalence class there is at least one path in this class (called representative) present in this system. The resulting system can become considerable smaller and if the resulting system satisfies an LTL formula then the respective cooperating systems in consideration models this LTL formula as well. [GW92] presents an algorithm that uses partial order reduction and checks deadlock-freedom in concurrent finite state systems. Furthermore, [GW92] shows how the algorithm can be used to check a certain class of system properties in concurrent finite state systems. A description of an application of techniques that use partial order reduction on practical relevant systems can be found in [GPS96].

Abstract interpretation is based on the idea to neglect parts of a system that do not have an impact on whether or not a desired system property holds by applying a more abstract semantic to the specifications of a system. The most frequently mentioned example is the abstraction of data values by bounded intervals. Abstract interpretation was introduced in [CC77]. Formally, the method consists of defining an abstraction function $\alpha : L \to L'$

and a concretization function $\gamma : L' \to L$ where L is the concrete state space, i.e., the state space of the original system and L' is the (desirable smaller) abstract state space on which one can perform model checking techniques. Many results request that (α, γ) is a Galois-connection from L to L' [CGL94, GS97, Lon93].

Sufficient Conditions

Consider an arbitrary decision problem. A sufficient condition on instances of the problem guarantees that the answer of the decision problem on an instance is "yes" if the condition holds for the instance. On the other hand, if the condition does not hold for an instance then we can not conclude the answer of the decision problem for the instance. Techniques that are based on a sufficient condition are useful if checking the condition requires much less resources as techniques that actually decide the problem, e.g., if deciding the problem is PSPACE-complete and applying the sufficient condition can be achieved in polynomial time in the size of the input. In our context an instance consists of a cooperating system and a system property and the question is whether or not the property holds in the system.

Several formal verification techniques that are based on sufficient conditions establish system properties by analyzing approximations of the global state space or the global behavior of a cooperating system. An approximation is called under-approximation if it describes a subset of the reachable states or transitions of the system in consideration. An over-approximation on the other hand describes a superset of the reachable states or transitions. Depending on the system property in consideration an under- or an over-approximation is needed to establish the property. For example, if the property specifies that certain states are not reachable and these states are not reachable in an over-approximation then these states are not reachable in the behavior of the system as well. If a property on the other hand requires that certain states are reachable then these states are reachable in the global be-

havior if they are reachable in an under-approximation of the system. A well researched class of system properties are so-called *safety properties* which state that "something bad does never happen" [Lam77, LS85]. Properties of this kind can be established in over-approximations of a cooperating system. If something bad does never happen in an over-approximation of a system then it certainly does never happen in the behavior of the system. Especially deadlock-freedom and the negated reachability property (deciding whether a certain state is not reachable) are safety properties.

Of course, if a property does not hold in an under-approximation respectively over-approximation then we can not make any statement as to whether the property in consideration holds in the system. In this case a refinement approach might help in order to modify an approximation such that a property can be established by analyzing a modified approximation. Refining under-approximations of cooperating systems means here to add states or transitions such that the resulting object remains to be an under-approximation. Whereas states or transitions are removed in over-approximations when refinement is applied. Clearly, if a technique fails to establish a property in an under-approximation respectively over-approximation then the technique might succeed in respective refined approximations.

Techniques based on over-approximations are, for example, introduced in [AC05, GDHH98, GD99, CHM$^+$93, CGL94, Kur94]. [CCQ94] constructs over-approximations by an approximative forward state space exploration that is refined by an exact backward exploration. Moon et al. [MJH$^+$98] supports CTL model checking by using over-approximations that are constructed by approximative forward traversal. [LPJ$^+$96] uses an approximative backward-analysis in order to construct over-approximations and refines them until an ACTL or ECTL formula can be proved or refuted. Over-Approximations of interaction systems are considered in [MCMM07] for establishing deadlock-freedom (the over-approximations are described in more detail in [MMC09b]). Under-approximations are for example considered in [PH98]. In Chapter 3 we describe an approach to efficiently represent

and refine over-approximations of interaction systems and in Chapter 4 we introduce a technique to establish deadlock-freedom in interaction systems that exploits over-approximations and is based on a sufficient condition that can be applied in polynomial time.

[AG97] describes an approach that tests a condition on the glue-code of a cooperating system in order to guarantee the construction of deadlock-free systems, i.e., this approach does not make use of under- or over-approximations. A technique based on a sufficient condition that is introduced in [IU01] uses partial equivalence relations between graphs constructed from the subsystems of a cooperation system without any state space exploration. An approach based on a sufficient condition for establishing liveness in interaction systems is described in [MCMM08].

1.2 Contribution

The contribution of this work consists of two parts. In the first part we discuss, how cooperating systems can be classified in order to describe interesting subclasses with respect to a complexity theoretic examination of deciding system properties in these subclasses. We describe several basic subclasses that are based on constraints regarding the communication structure between the subsystems and are relevant in practice. We show that deciding the reachability problem in these classes is PSPACE-complete. Thus, we cannot expect that there is a technique that decides the reachability problem in these classes in polynomial time. Our results justify correctness by constructions approaches that exploit characteristics of these subclasses and the application of techniques that are based on sufficient conditions and establish the reachability problem in these classes in polynomial time [Hoa85, BR91, BCD02, MCM08a, Lam09, HJK10]. In addition, the results motivate the research of further sufficient conditions, that exploit the individual characteristics of our subclasses, in order to construct more efficient

techniques that tests for the reachability of states.

In a second part, which makes up the larger part of this work, we introduce a technique for establishing deadlock-freedom in cooperating systems that is based on a sufficient condition and can be tested in polynomial time. Our approach is based on the analysis of compact representations of over-approximations of the reachable global behavior of a cooperating system. We call these representations *abstract over-approximations.* An abstract over-approximation is based on a subset of subsystems and induces an over-approximation of the global behavior. Thus, we argue that our abstract over-approximations have the potential to be the basis of techniques that establish safety properties in cooperating systems based on a sufficient condition. We introduce an operator called *Edge-Match* that we use to refine a family of abstract over-approximations by a pairwise comparison. Our definition of abstract over-approximations and our approach of refining these can, in a certain way, be seen as a state space reduction approach. The sum of the sizes of all abstract over-approximations in our approach is usually significantly smaller than the size of the global behavior of a cooperating system, i.e., we apply our approach to establish deadlock-freedom to objects that are significantly smaller than the global behavior of the system in consideration. On the other hand, in comparison to state space reduction techniques, our abstract over-approximations are not suited to directly apply known model checking techniques.

To circumvent complexity issues regarding the verification of system properties in cooperating systems we proceed as follows. In order to establish a safety property P of a complex cooperating system, we propose a three step approach:

1. The construction of polynomially many so-called abstract over-approximations of the reachable state space such that each abstract over-approximation is of polynomial size and induces an over-approximation of the system. This topic is discussed in Chapter 3.

2. The refinement of the abstract over-approximations by a pairwise comparison with an operator that we call Edge-Match which can be performed in polynomial time. This refinement approach is introduced in Chapter 3.

3. The construction of a predicate P' on the abstract over-approximations such that

   i) P holds if P' holds for all abstract over-approximations and

   ii) P' can be checked in polynomial time.

   This concept is discussed for the system property of deadlock-freedom in Chapter 4.

Viewed abstractly our method establishes a sufficient condition for the validity of property P. This condition can be checked in polynomial time.

To the best of our knowledge there are very few other approaches that deal with the refinement of over-approximations of cooperating systems in our sense that are based on subsets of subsystems. Approaches that are similar or related to our work are Minnameier's *Cross-Checking* operator for the refinement of overlapping over-approximations of the reachable state-space of interaction systems [MMC09b], the work of Govindaraju et al. [GDHH98, GD99] that concerns approximative reachability in cooperating systems and establishing invariants (a subclass of safety properties) in synchronous hardware modeled by *Mealy machines* and Attie and Chockler's approach to establish deadlock-freedom in cooperating systems [AC05] by analyzing over-approximations that are based on subsets of three subsystems each. These approaches are discussed in detail in Chapter 3 in Section 3.4.

1.3 Road Map

This work is structured as follows. In the next section we introduce the formalism of interaction systems – a formalism for modeling cooperating systems that was introduced by Sifakis and Gössler in [GS03]. In the remainder of this work we use this formalism to model cooperating systems.

In the second chapter we present several reductions which prove that deciding important system properties in certain subclasses of interaction systems is PSPACE-complete. These subclasses are defined by restricting the topology that is induced by the glue-code of a cooperating system, i.e., the communication structure between the subsystems.

In Chapter 3 we consider a concept of over-approximations of the global behavior of interaction systems. These over-approximations are suited as a basis for techniques that establish safety properties in interaction systems, i.e., one can directly apply model checking techniques for safety properties. These over-approximations suffer from the state space explosion problem just like the global behavior of an interaction system, i.e., they are not suited for an approach that efficiently ensures a system property the verification of which requires the examination of the reachable state space. To circumvent this problem we introduce a special kind of over-approximations – abstract over-approximations. An abstract over-approximation is a compact representation of an over-approximation and, suitably chosen, it is of polynomial size with respect to the parameters of the underlying interaction system. This is, an abstract over-approximation induces an over-approximation of the global behavior of an interaction system. We use abstract over-approximations as a basis for an approach to establish the safety property of deadlock-freedom in interaction systems in polynomial time in the size of an underlying interaction system. This approach is introduced in Chapter 4. The approach consists of a condition that is tested on a family of abstract over-approximations which, if true, guarantees that there is a deadlock-free over-approximation of the global behavior, i.e., the global

behavior is deadlock-free as well. Moreover, in case our approach fails, we exemplify how we can use information that was produced in our approach in order to modify a system such that our approach ensures deadlock-freedom. If our approach fails and the system in consideration is in fact deadlock-free then a refinement of the abstract over-approximations, i.e., the removal of states and transitions such that the result remains to be an abstract over-approximation, might result in a family of abstract over-approximations on which our approach succeeds. For this purpose we introduce in Chapter 3 an operator that refines abstract over-approximations by a pairwise comparison and propose an approach to calculate a fixed-point with respect to an application of this operator on a family of abstract over-approximations in polynomial time in the size of the underlying interaction system. In Chapter 5 we describe a tool that implements our techniques, introduce several complex and parameterized examples and provide results of our refinement approach. Chapter 6 establishes a connection between our concept of abstract over-approximations and their refinement and the field of database theory. Besides of pointing out this interesting connection, we use results from this field in order to make preciseness statements about our refinement approach.

Chapter 7 provides an overall conclusion of this thesis.

1.4 Interaction Systems

Here we give a brief introduction to interaction systems that have been proposed by Sifakis and Gössler in [GS03, Sif05] to model component based systems. The model was studied, e.g., in [BBSN08, BBG11, LMC11, MCMM07, MCM08a, MCMM08, MMC09a, BBNS09, BGL$^+$11, GGMC$^+$06], has been used to model, e.g., biochemical reactions [MCSW07] and was integrated in the BIP framework [BBS06].

An interaction system consists of subsystems called components that offer

interfaces for a cooperation among them. The cooperation is specified by a glue-code that connects interfaces of different components. The glue-code is modeled by so called interactions. An interaction specifies a multiway cooperation among components by connecting different interfaces (called ports) of different components. The model is defined in two layers. The first layer, the interaction model, provides the names of the components, their interfaces and the glue-code. In the second layer, the interaction system, in addition the behavior of the components by labeled transition systems is described. Thus, the description of the glue-code and of the behavior of the components are clearly separated. In our context, this separation is important because many results in this work are based on the glue-code of interaction systems and independent of the behavior of the components.

In this work, we use interaction systems as a formalism to model cooperating systems. This is, all our results and techniques are based on this formalism. Nevertheless, we want to point out that all our techniques can be easily applied to other formalisms that model cooperating systems. This can be achieved by either adapting our techniques or by using a mapping among formalisms – see, e.g., [MCM08b] for a mapping between interaction systems and 1-safe Petri nets. In the following we discuss some formalisms that can be used to model cooperating systems and are similar to interaction systems.

1.4.1 Related Formalisms

The formalism of interaction systems is a very general formalism for modeling cooperating systems and abstracts from data values, timed behavior, the description of input/output relations, probabilistic behavior or guarded commands. Some formalisms for modeling cooperating systems that are similar to interaction systems are Pnueli's *concurrent programs* [Pnu79], Lynch's *I/O automata* (Input/Output automata) [LT87, CCK+05, KLSV06] and Henzinger's and de Alfaro's *interface automata* [dAH01]. These formalisms are briefly discussed in the following.

Pnueli uses concurrent programs in order to introduce the temporal logic LTL in concurrent systems. The behavior of $n \in \mathbb{N}$ processes that share a set of variables is described by labeled transition systems the edges of which are labeled by commands and guards. A command can be executed if a predicate, named *guard*, on the shared variables is true. Guards are obsolete if each guard represents the value true. A command is an assignment that changes the values of a set of shared program variables (if there are no shared variables then each command is empty). A state change in the overall behavior of the processes corresponds to the execution of a transition of exactly one process. The model is used in [BVW94] in order to prove the PSPACE-completeness of CTL and LTL model checking in concurrent programs.

An I/O automaton is given by a labeled transition system the edges of which are labeled by actions. The authors distinguish between input, output and internal actions and demand that in each state of an I/O automaton each input action is available, i.e., an I/O automaton is always ready to receive any input. The cooperation among several I/O automata is achieved by an associative composition which depends on a pairwise composability condition, i.e., two I/O automata are composable if their actions are disjoint, except that input actions of one may coincide with output actions of the other. The cooperation takes place on the shared input/output actions which become internal actions in the composed system. There are extensions that extend the expressive strength of the formalism of I/O automata by the ability to model, e.g., probabilistic [CCK$^+$05] or time dependent [KLSV06] behavior. An interface automaton is an I/O automaton without the restriction that in each state each input action must be available.

Other formalisms that are similar to interaction systems are Arnold's synchronous product of labeled transition systems [Arn94], *team automata* [Ell97, TBEKR03] and even finite state machines as *Mealy machines* [Mea55] or *Moore machines* [Moo56] can be used to model cooperating systems by considering sets of states variables (see, e.g., [GDHH98, GD99] for an approach

to analyze synchronous hardware modeled by Mealy machines).

1.4.2 Definitions

In the following we formally describe the formalism of interaction systems and provide illustrative examples.

Definition 1.1:
Let K be a set of **components** and $\{A_i\}_{i \in K}$ a family of pairwise disjunct sets. A_i is the **port set** of component $i \in K$. We denote components by lowercase letters i, j, k or l and, if not stated otherwise, assume that $K = \{1, 2, \ldots, n\}$. A port is denoted by the lowercase letter a indexed by the respective component, i.e., a_i. An **interaction** α is a nonempty set of ports from different components, i.e.,

a) $\alpha \subseteq \bigcup_{i \in K} A_i$ and

b) for all $i \in K$ holds that $|\alpha \cap A_i| \leq 1$.

An interaction $\alpha_i = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ with $a_{i_j} \in A_{i_j}$ $(1 \leq j \leq k)$ denotes a possible cooperation among the components $i_1, \ldots, i_k$ via their respective ports. For an interaction α and a component $i \in K$ let $i(\alpha) = A_i \cap \alpha$. Note that b) ensures that $|i(\alpha)| \leq 1$. If $i(\alpha) \neq \emptyset$, i.e., $i(\alpha)$ is a set that consists of exactly one port $a_i \in A_i$, then we say that i **participates** in α and a_i is the port of i that participates in α. If $i(\alpha) = \emptyset$ then we say i does not participate in α.

A set Int of interactions is called **interaction set (for K)**, if each port appears in at least one interaction in Int, i.e., $\bigcup_{i \in K} A_i = \bigcup_{\alpha \in Int} \alpha$. The tuple $IM = (K, \{A_i\}_{i \in K}, Int)$ is called **interaction model** if Int is an interaction set for K.

The glue-code among the components, that is modeled by interactions, allows for a multiway cooperation, i.e., an interaction can contain ports of more than two components. This is in contrast to other formalisms that

model cooperating systems and only allow a cooperation between two subsystems. The concept of multiway cooperation among subsystems is called *global synchronous communication* in [Osa12] and is listed as a design guideline that can result in a substantially reduced number of states in the global behavior of a system compared to a design guideline that only allows a cooperation among two subsystems. For example I/O automata respectively interface automata only allow cooperation between the input action of one automaton and the output action of another. Other well-known formalisms with a two-way communication are, e.g., the process algebras *CCS* [Mil82] where communication takes place between an action a and a counterpart $\bar{a}$ and *CSP* [Hoa85] where communication occurs between an input and an output channel. Many process algebras that are based on CCS or CSP pursue this restriction, e.g., *PEPA* [Hil96] or the π-*calculus* [MPW92a, MPW92b]. Multiway communication can be found in, e.g., *Petri nets* [Pet67] where one transition can move several tokens among multiple places or in the process algebra *LoCo* [vW08] which is inspired by Petri nets.

We introduce here a simple example that we use as a running example in the remainder of this work in order to illustrate our techniques.

Example 1.1:
Users login into terminals $(\mathrm{TER}_1, \mathrm{TER}_2, \ldots, \mathrm{TER}_k)$ in order to retrieve information. The terminals are connected to a gateway server (GS) that connects to an authentication database (ADB) in order to validate a user request. The ADB sends a confirmation to a database (DB) which transfers the requested information to the GS which in turn forwards the information to the terminal from which the request was initiated.

For ease of presentation, we model here a system with only two terminals. Note that the results and observations in this work can be adapted to models with an arbitrary number k of terminals. Let

$$K = \{\mathrm{TER}_1, \mathrm{TER}_2, \mathrm{GS}, \mathrm{ADB}, \mathrm{DB}\}$$

be a set of components. From a terminal TER_i $(i = 1, 2)$ a user can request

(req_i) an information and get an information (get_i) if the authentication process is finished. The gateway server GS can get a request (get_req) for an information, request an authentication (req_auth) of a user, get a value (get_val) that represents the requested information and send a value ($send$) to the user. The authentication database ADB can get an authentication request (get_auth_req) and authenticate a user ($auth$). The database DB can get an authentication (get_auth) and send a value ($send\_val$). Thus, the sets of ports for the components in K are defined as follows.

$$\begin{aligned} A_{\mathrm{TER}_i} &= \{req_i, get_i\} \ \ (i = 1, 2) \\ A_{\mathrm{GS}} &= \{get\_req, req\_auth, get\_val, send\} \\ A_{\mathrm{ADB}} &= \{get\_auth\_req, auth\} \\ A_{\mathrm{DB}} &= \{get\_auth, send\_val\} \end{aligned}$$

Let Int be a set, consisting of the following interactions. Note that Int is an interaction set for K.

$$\begin{aligned} send\_req_i &= \{req_i, get\_req\} \ \ (i = 1, 2) \\ ask\_auth &= \{req\_auth, get\_auth\_req\} \\ authorize &= \{auth, get\_auth\} \\ send\_data &= \{send\_val, get\_val\} \\ get\_replay_i &= \{send, get_i\} \ \ (i = 1, 2) \end{aligned}$$

$\mathrm{IM} = (K, \{A_i\}_{i \in K}, Int)$ is a well defined interaction model. We display an interaction model graphically by drawing the components as squares that are labeled by the names of the components. On the edge of the squares we draw the ports as black dots that are labeled by the names of the ports. The interactions are depicted as lines, labeled by the names of the interactions, that connect the appropriated ports. Figure 1.1 shows a graphical representation of IM.

An interaction model gives the names of components of a cooperating system, their ports and specifies the cooperation between the components via ports. An interaction system extends this specification by assigning a be-

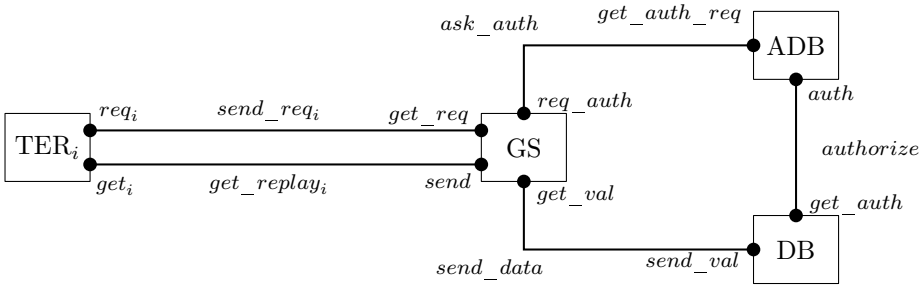


Figure 1.1: Graphical representation of the interaction model IM from Example 1.1. For ease of presentation the two components TER_1 and TER_2 are depicted as one box TER_i $(i = 1, 2)$ with respective ports.

havior to each component in form of a labeled transition system. We use the following definition of labeled transition systems.

Definition 1.2:
A **labeled transition system** is a tuple $S = (Q, A, \rightarrow_S, q^0)$. The set Q is the **state space** of S, A a set of **labels**, $\rightarrow_S \subseteq Q \times A \times Q$ a **transition relation** and $q^0 \in Q$ the **initial state**. An element $q \in Q$ is called **state** and an element (q, a, q') of $\rightarrow_S$ is called **transition**. Instead of $(q, a, q') \in \rightarrow_S$ we use the notation $q \xrightarrow{a}_S q'$. In graphical representations of labeled transition systems we mark the initial state by an incoming arrow (see, e.g., Figure 1.2 in Example 1.2).

For a state $q \in Q$ let

$$en(q) = \{a \in A | \exists_{q' \in Q} q \xrightarrow{a}_S q'\}.$$

We say the label $a \in en(q)$ is **enabled** in q.

We use the standard definition of reachability in transition systems, i.e., a state $q \in Q$ is **reachable** in S if $q = q^0$ or there is a sequence of transitions

$$q^0 \xrightarrow{a_1}_S q^1, q^1 \xrightarrow{a_2}_S q^2, q^2 \xrightarrow{a_3}_S q^3, \ldots, q^{n-1} \xrightarrow{a_n}_S q^n$$

with $n \geq 1$ and $q^n = q$. This is, a sequence that starts in the initial state and ends in q. A transition $q \xrightarrow{a}_S q'$ is **reachable** if q is reachable. The set of reachable transitions in S is called the **reachable behavior** of S.

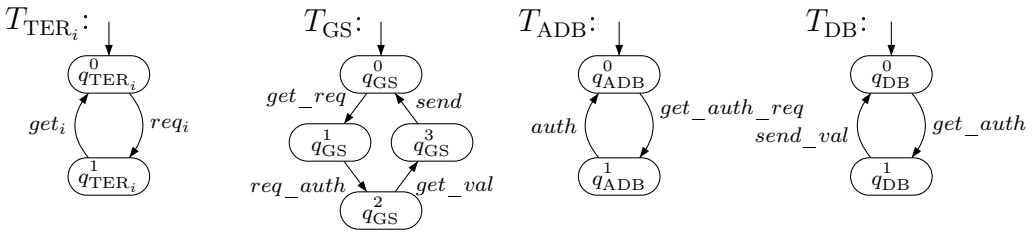


Figure 1.2: Behavior of the components in Example 1.1. T_{TER_i} is depicted for $i = 1, 2$.

A state $q \in Q$ is called a **deadlock** if $en(q) = \emptyset$, i.e., if no label is enabled in q. S is called **deadlock-free** if no deadlock is reachable in S.

Definition 1.3:
Let IM $= (K, \{A_i\}_{i \in K}, Int)$ be an interaction model. The tuple Sys $=$ (IM, $\{T_i\}_{i \in K}$) is called **interaction system** where $T_i = (Q_i, A_i, \rightarrow_i, q_i^0)$ for $i \in K$ is a labeled transition system. In the following, for $i \in K$, we refer to T_i as the **local behavior**, to Q_i as the **local state space**, $\rightarrow_i$ as the **local transition relation** and to q_i^0 as the **local initial state** of component i.

The following example provides local behaviors for the components in Example 1.1.

Example 1.2:
Figure 1.2 gives local behaviors to the components from Example 1.1. The interaction model IM that was defined in Example 1.1 together with the local behavior given by the transition systems in Figure 1.2 forms a well defined interaction system Sys = (IM, $\{T_i\}_{i \in K}$).

From the local behavior of the components of an interaction system Sys and the interaction set we can determine the global behavior of Sys, in form of a transition system, as follows.

Definition 1.4:
Let Sys $=$ (IM, $\{T_i\}_{i \in K}$) be an interaction system with interaction model IM $= (K, \{A_i\}_{i \in K}, Int)$ and the set of components $K = \{1, 2, \ldots, n\}$.

The **global behavior** of Sys is the transition system $T = (Q, Int, \rightarrow_T, q^0)$ where

- the Cartesian product $Q = \prod_{i \in K} Q_i$, which we consider to be independent from the order of the components, is the **global state space**,
- $q^0 = (q_1^0, \ldots, q_n^0)$ is the **global initial state** and
- $\rightarrow_T \subseteq Q \times Int \times Q$ is the **global transition relation** with $q \xrightarrow{\alpha}_T q'$ if for all $i \in K$:
  - if $\alpha \cap A_i = \{a_i\}$ then $q_i \xrightarrow{a_i}_i q_i'$ and
  - if $\alpha \cap A_i = \emptyset$ then $q_i = q_i'$.

If $K = \{1, 2, \ldots, n\}$ then a global state $q \in Q$ has the form $q = (q_1, q_2, \ldots, q_n)$. If we consider a global state $q \in Q$ then we denote the local state of component $i \in K$ in q by q_i.

The interaction system Sys is called **deadlock-free** if there is no reachable state $q \in Q$ in the global behavior T such that q is a deadlock.

Globally a transition $q \xrightarrow{\alpha}_T q'$ can be performed if each port in α is enabled in the state of the local behavior of its respective component.

Remark 1.1:
Let Sys $= (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model IM $= (K, \{A_i\}_{i \in K}, Int)$ and global behavior $T = (Q, Int, \rightarrow_T, q^0)$. In this work we only consider interaction systems where K and each A_i and Q_i for $i \in K$ are finite.

The **size** of Sys is given by the sum over the sizes of K, Int and A_i, Q_i and $\rightarrow_i$ for $i \in K$.

Example 1.3:
The interaction system introduced in Example 1.1 and 1.2 consists of five components where the local behavior of four components (T_{TER_1}, T_{TER_2}, T_{ADB} and T_{DB}) contains exactly two states and the local behavior of one

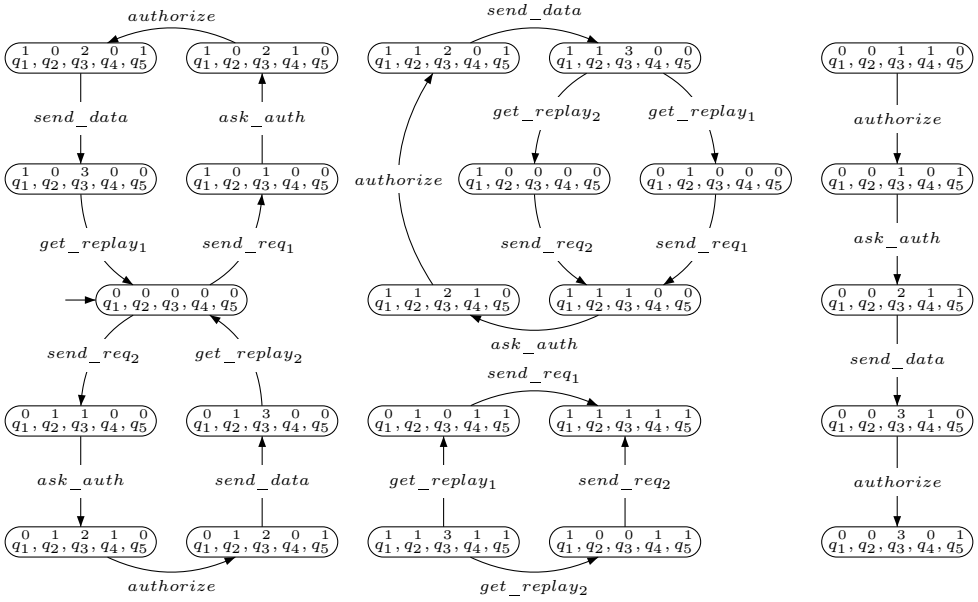


Figure 1.3: Part of the global behavior T in Example 1.3.

component (T_{GS}) exactly four states. It follows that the global behavior $T = (Q, \mathrm{Int}, \rightarrow_T, q^0)$ has a state space Q that consists of $2^4 \cdot 4 = 64$ states. The set of reachable states of this system is relatively small and consists of only nine reachable states. The reachable transitions of T and some of the unreachable transitions are depicted in Figure 1.3. For better readability, in Figure 1.3, the local states are indexed by the numbers 1 to 5, where 1 corresponds to TER_1, 2 to TER_2, 3 to GS, 4 to ADB and 5 to DB. Note that, in each state in the reachable state space of T, there is an enabled interaction and there are states in the unreachable state space where no interaction is enabled. This is, the reachable behavior of T and thus T itself is deadlock-free. Nevertheless, note that there are unreachable deadlocks.

Chapter 2

Architectural Constraints & Reachability

2.1 Introduction

In this chapter we explore the complexity theoretic classification of the reachability problem in subclasses of cooperating systems, i.e., the problem of deciding whether or not a certain state is reachable in a cooperating system. We use the formalism of interaction systems that was introduced in Chapter 1 to model cooperating systems. Deciding reachability in general interaction systems was proven to be PSPACE-complete [MCM08c]. Here we define different subclasses of interaction systems by architectural constraints and show that deciding the reachability problem in these subclasses remains PSPACE-complete.

Popular decision problems that are complete in NP or even in PSPACE are decidable in polynomial time in certain subclasses of instances. Maybe the most popular example is the Boolean satisfiability problem where 3SAT is NP-complete and 2SAT is decidable in polynomial time. The problem HORNSAT (the problem of deciding whether a given set of propositional

Horn clauses is satisfiable) is even decidable in linear time. Similarly, the quantified 3SAT problem is PSPACE-complete, whereas the quantified 2SAT problem is also decidable in polynomial time. See [GJ79] for descriptions and more examples. These examples raise the question whether there are "interesting" classes of cooperating systems for which the reachability problem is efficiently decidable.

There are various starting points to specify subclasses of cooperating systems.

1. Restrictions regarding the behavior of the subsystems.
2. The degree of synchronization among the subsystems as systems with a very high degree of synchronization tend to display a smaller reachable state space.
3. The glue-code, i.e., the structure of the interaction among the subsystems.

Here, our concern lies on the latter. In this chapter we examine restrictions with respect to the interactions of a system, i.e., restrictions regarding the communication structure. We show that deciding the reachability problem remains PSPACE-complete even if we strongly restrict the communication structure between the components in various ways. For this purpose we define an undirected graph such that the nodes are the components of an interaction system and two components are connected by an edge if there is an interaction in which both components participate – we call this graph *interaction graph*. By the structure of an interaction graph we can define subclasses of systems. Basic graph structures of an undirected graph $G = (V, E)$ are, e.g.,

- trees – G is connected and acyclic,
- stars – one node is of degree $|V| - 1$ and all other nodes are of degree 1,

- lines – G is connected, exactly two nodes are of degree 1 and all other nodes are of degree 2.

Especially systems with a tree-like communication structure induce an important class of cooperating systems. Many interesting systems belong to this class, e.g., hierarchical systems or networks built by a master-slave operator. This class has been early studied, e.g., in [Hoa85, BR91] and more recently, e.g., in [BHH$^+$06, BCD02, MCM08a]. Star structures appear in practice in, e.g., client/server systems as banking or booking systems. Properties of cooperating systems with a communication pattern that forms a star were considered for example in [Lam09, BCD02, GSM07]. Cooperating systems with a linear structures appear in, e.g., pipeline systems as instruction pipelines or general queue based algorithms.

We show the PSPACE-hardness of deciding the reachability problem in subclasses of systems with a tree-like communication pattern (i.e., the interaction graph is a tree) by providing a detailed reduction from the trueness problem of quantified Boolean formulas. Moreover we strengthen this result and show that the same complexity holds for even simpler systems with a linear and a star-like communication pattern. The PSPACE-hardness of deciding the reachability problem in systems with a linear pattern is accomplished by a reduction from the acceptance problem in linear bounded Turing machines and in systems with a star-like pattern by a reduction from the reachability problem in general interaction systems.

Additionally, we modify our reduction for systems with a tree-like communication pattern to proof that deciding progress in such systems is PSPACE-complete as well.

A communication structure that forms a line respectively a star forms particularly a tree, i.e., our PSPACE-completeness results of deciding the reachability problem in cooperating systems with a linear respectively star structure imply the PSPACE-hardness of deciding the reachability problem in systems with a communication structure that forms a tree. When we started to

think about the complexity theoretic classification of the reachability problem in cooperating systems with a tree-like communication structure, our question in mind was whether techniques based on sufficient conditions that establish reachability or reachability based systems properties in such systems are justified [Mar09, Hoa85, BR91, BHH+06, BCD02, MCM08a], i.e., could it be the case that the reachability problem in such systems can be decided efficiently? The generalization of the tree-like result to systems with a linear and star-like communication structure was considered by us at a later stage. This is, we introduce our reduction to systems with a tree-like communication pattern for the sake of completeness. On the other hand our reduction to systems with a tree-like communication pattern brought up a general technique of propagating information through a system while avoiding circles in the communication structure, i.e., the cooperation between two subsystems in a system which communication structure forms circle can be remodeled by propagating the cooperation among the other subsystems on the circle. We refer to this technique in our proof that deciding the reachability problem in linear and star-like systems is PSPACE-hard. Furthermore, we use our proof of the PSPACE-hardness of deciding the reachability problem in tree-like systems in order to show that the decision problem whether a global state is reachable where a fixed component is in a fixed local state can not be decided in polynomial time.

This chapter is organized as follows. In Section 2.2 we provide definitions that we need in the remainder of this chapter. In Section 2.3 we show that the reachability problem in tree-like systems is in PSPACE and present a reduction that proves the PSPACE-hardness of this problem [MCS10]. In Section 2.4 we argue why deciding whether a global state is reachable in the behavior of a tree-like interaction system that contains a certain fixed local state of a fixed component can not be decided in polynomial time. Section 2.5 and 2.6 provide reductions that show that the reachability problem remains PSPACE-hard in linear respectively star-like interaction systems [MCS13b]. In Section 2.7 we outline why deciding progress in tree-like inter-

action systems is PSPACE-complete. A conclusion can be found in Section 2.8.

2.2 Definitions

We focus on structural constraints on interaction systems. By this we mean constraints concerning the communication structure of a system. We first consider systems with a tree-like communication pattern. Then we further restrict the pattern to linear and star-like communication. The following definition introduces the interaction graph of an interaction model. This graph is the basis of our approach to define subclasses with respect to a certain communication structure. The nodes of this undirected graph are the components of an interaction model and two nodes are connected if there is an interaction in which both respective components participate.

Definition 2.1:
Let IM $= (\mathrm{K}, \{A_i\}_{i \in \mathrm{K}}, \mathrm{Int})$ be an interaction model with $|\,\mathrm{K}\,| = n$. The **interaction graph** $G = (\mathrm{K}, E)$ of IM is an undirected graph with $\{i, j\} \in E$ if and only if there is an interaction $\alpha \in \mathrm{Int}$ with $i(\alpha) \neq \emptyset$ and $j(\alpha) \neq \emptyset$.

Let Sys be an interaction system with interaction model IM. Let G be the interaction graph of IM. We call IM respectively Sys

- **tree-like** if and only if G is a tree, i.e., G is connected and acyclic,
- **star-like** if and only if G is a star, i.e., exactly one node is of degree $n - 1$ and all other nodes are of degree 1 and
- **linear** if and only if G is connected and exactly two nodes have degree 1 and any other node degree 2.

Let $\mathrm{IS}_{\mathrm{general}}$ be the class of all interaction systems with no restrictions to G, i.e., the set of all interaction systems, $\mathrm{IS}_{\mathrm{tree}}$ the subclass of all tree-like interaction systems, $\mathrm{IS}_{\mathrm{star}}$ the subclass of all star-like systems and $\mathrm{IS}_{\mathrm{line}}$ the subclass of all linear systems.

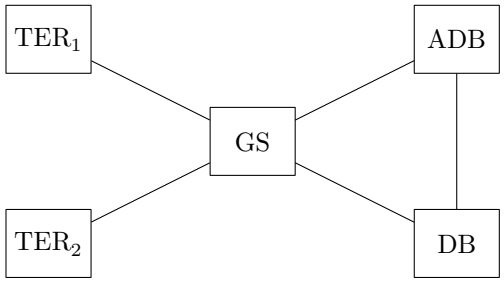


Figure 2.1: Interaction graph G of IM in Example 1.1.

Remark 2.1:
Note that tree-like, star-like and linear interaction systems with a set Int of interactions imply that for all $\alpha \in \text{Int}$ $|\alpha| \leq 2$.

Remark 2.2:
For the subclasses defined in Definition 2.1 the following set inclusions hold. Obviously, IS_{tree}, IS_{star} and IS_{line} are all included in $\text{IS}_{\text{general}}$. The subclasses IS_{star} and IS_{line} are included in IS_{tree}. The intersection of IS_{star} and IS_{line} is not empty, and neither $\text{IS}_{\text{star}} \setminus \text{IS}_{\text{line}}$ nor $\text{IS}_{\text{line}} \setminus \text{IS}_{\text{star}}$ is empty.

Here we give three examples of interaction systems and their respective interaction graphs.

Example 2.1:
Consider Example 1.1 that was introduced in Chapter 1. The interaction graph G of the interaction model in Example 1.1 is depicted in Figure 2.1. The graph G contains a cycle, i.e., an interaction system based on the interaction model from Example 1.1 is not included in IS_{tree}, IS_{line} or IS_{star}.

Example 2.2:
This example illustrates a simple component based client/server model. A server offers a service that can be requested by a client. Let s be a component that models a server then this component offers the ports o_s (offer a service) and f_s (finish a service). A component c that models a client in need of a certain service features the ports r_c (request a service) and g_c (gained

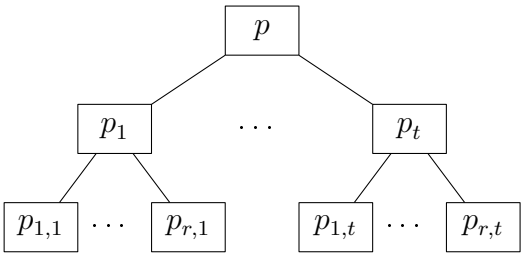


Figure 2.2: Interaction graph G of $\mathrm{IM}_{r,t}$ in Example 2.2.

a service). We consider here components that are able to both offer and request services.

In the following we consider a parameterized instance of a client/server model. Let $\mathrm{IM}_{r,t} = (K, \{A_i\}_{i \in K}, \mathrm{Int})$ be an interaction model with $r, t > 0$ where

$$K = \{p_{i,j} | 1 \leq i \leq r, 1 \leq j \leq t\} \cup \{p_j | 1 \leq j \leq t\} \cup \{p\}.$$

For $1 \leq i \leq r$ and $1 \leq j \leq t$ a component $p_{i,j}$ models a process that has a client function only, i.e., let $A_{p_{i,j}} = \{r_{p_{i,j}}, g_{p_{i,j}}\}$. For $1 \leq j \leq t$ a component p_j models a process that has a client and a server function, i.e., let $A_{p_j} = \{r_{p_j}, g_{p_j}, o_{p_j}, f_{p_j}\}$. The component p models a process with a server function only, i.e., $A_p = \{o_p, f_p\}$.

For $1 \leq i \leq r$ and $1 \leq j \leq t$ component $p_{i,j}$ needs the service that is offered by component p_j. The component p_j on the other hand is in need of the service that is offered by p. Thus, the interaction set of IM is given as follows.

$$\begin{aligned} \mathrm{Int} \;=\; & \{\{r_{p_{i,j}}, o_{p_j}\}, \{g_{p_{i,j}}, f_{p_j}\} | 1 \leq i \leq r, 1 \leq j \leq t\} \cup \\ & \{\{r_{p_j}, o_p\}, \{g_{p_j}, f_p\} | 1 \leq j \leq t\}. \end{aligned}$$

The interaction graph G of the interaction model $\mathrm{IM}_{r,t}$ is given in Figure 2.2 and apparently a tree, i.e., an interaction system Sys with interaction model $\mathrm{IM}_{r,t}$ is tree-like and thus included in $\mathrm{IS}_{\mathrm{tree}}$.

Example 2.3:
This example illustrates a linear interaction system. We consider a simple communication pipeline consisting of n stations. Station one initiates passing a message to station two, station two passes the message to station three and so on. If the message arrives at station n then station n passes an acknowledge message, on the same way, back to station one.

Let IM $= (K, \{A_i\}_{i \in K}, \text{Int})$ be the interaction model with components $K = \{s_1, s_2, \ldots, s_n\}$ for $n \geq 2$ where s_i models station i for $1 \leq i \leq n$. A station s_i with $1 < i < n$ can receive a message (rec_m_i), pass the message forward ($send\_m_i$), receive an acknowledge (rec_a_i) and pass the acknowledge forward ($send\_a_i$). Station s_1 can only send the initial message and receive the acknowledge and station s_n can only receive a message and send an acknowledge. This is, the port sets of the components are defined as follows.

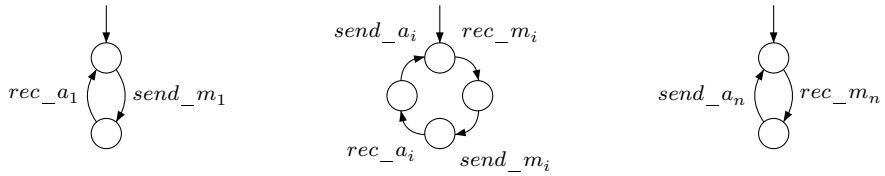
$$\begin{aligned} A_{s_1} &= \{send\_m_1, rec\_a_1\} \\ A_{s_i} &= \{rec\_m_i, send\_m_i, rec\_a_i, send\_a_i\},\ 1 < i < n \\ A_{s_n} &= \{rec\_m_n, send\_a_n\} \end{aligned}$$

The interaction set Int is given by the following interactions.

$$\begin{aligned} send\_message_i &= \{send\_m_i, rec\_m_{i+1}\},\ 1 \leq i < n \\ send\_acknowledge_i &= \{send\_a_i, rec\_a_{i-1}\},\ 1 < i \leq n \end{aligned}$$

Let $Sys = (IM, \{T_i\}_{i \in K})$ be the interaction system with local behavior depicted in Figure 2.3.

The interaction graph G of IM is depicted in Figure 2.4. G forms a line of components. Thus, IM is a linear interaction model and Sys is a linear interaction system.



(a) T_{s_1} (b) T_{s_i}, $1 < i < n$ (c) T_{s_n}

Figure 2.3: Local behavior of the components in a simple communication pipeline.

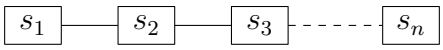


Figure 2.4: Interaction graph G for the interaction model IM in Example 2.3.

2.3 PSPACE-completeness of Reachability in Tree-Like Systems

Here we show that deciding the reachability problem in tree-like interaction systems is PSPACE-complete. We show the PSPACE-hardness by providing a reduction from the trueness problem of quantified Boolean formulas (QBF) [GJ79], i.e., we describe how a quantified Boolean formula H can be mapped to a tree-like interaction system Sys_H and a state q in the global behavior T of Sys_H in polynomial time such that deciding the reachability problem for q in Sys_H corresponds to deciding whether or not H is true.

First we formally introduce the trueness problem of quantified Boolean formulas and the reachability problem in tree-like interaction systems and show that deciding this problem is in PSPACE. In a second part we define a tree-like interaction system Sys_H for a quantified Boolean formula H by specifying the components, the ports of the components, the interactions and the local behavior of the components. In the last part we prove in detail that deciding the reachability problem for a certain state in the global behavior of Sys_H corresponds to deciding whether H is true.

QBF and RIST

QBF

A **quantified Boolean formula** (**QBF**) (see [GJ79] for details) consists of a Boolean formula f over variables $x_1, \ldots, x_n$ where each variable is bounded by a quantifier $\forall$ or $\exists$. For example, in *prenex normal form*, i.e., the string representation of a formula in which all quantifiers are written in front of the Boolean formula, the formula

$$P = (\mathbb{Q}_1 x_1)(\mathbb{Q}_2 x_2) \ldots (\mathbb{Q}_n x_n) f$$

is a QBF where $\mathbb{Q}_i$ is either an existential quantifier $\exists$ or an universal quantifier $\forall$ for $1 \leq i \leq n$.

If P is a QBF then P is a **subformula** of P. If P' is a subformula of P and P' is of the form $P' = \oplus P''$, where $\oplus$ is an unary operator (e.g. , $P' = \neg P''$ or $P' = \exists x.P''$) then P'' is a subformula of P. If P' is of the form $P' = P_1 \otimes P_2$, where $\otimes$ is a binary operator (e.g., $P' = P_1 \vee P_2$ or $P' = P_1 \wedge P_2$) then P_1 and P_2 are subformulas of P. The **size** of a QBF P is the number of subformulas of P.

Each QBF instance P' can be rewritten into an equivalent instance P over the grammar

$$P ::= x | \neg P | P \wedge P | \exists x.P.$$

The rewriting can be achieved in polynomial time in the size of P' by parsing the subformulas of P' top down, starting with P', and applying respective equivalences. Without loss of generality, in the following we only consider QBF instances that are rewritten over this grammar. Let P be a QBF then the question is whether P is true. The language $TQBF$ is defined as the set of true QBF instances and it is well known that the decision problem whether a QBF is in $TQBF$ is PSPACE-complete ([GJ79]).

In order to determine whether a QBF P is true or false we introduce the straightforward, recursive Algorithm 1 called *eval*. This algorithm is used in

the remainder to illustrate our reduction. The algorithm recursively parses a QBF P and evaluates each subformula for each combination of truth assignments to the variables. In Line 1 and 2 a subformula of the form $P = \exists x.P'$ is parsed. The term $P'_{x=true}$ respectively $P'_{x=false}$ denotes the subformula P' with $true$ respectively $false$ assigned to all occurrences of variable x in P'. In Line 5 respectively 8 a subformula of the form $P = \neg P'$ respectively $P = P' \wedge P''$ is parsed. In Line 11 a subformula that consists of a variable is parsed. The function $value(x)$ returns the truth value that is currently assigned to the variable x. Remember that each variable in QBF is bounded by a quantifier, i.e., in our case bounded by an existence quantifier. Thus, before Line 11 is executed for variable x, Line 2 was executed where the subformula that quantifies x was parsed and a truth value was assigned to all occurrences of x. Obviously, $P \in TQBF$ if and only if $eval(P)$ returns $true$.

Algorithm 1 $eval(P)$

1: **if** $P = \exists x.P'$ **then**
2: **return** $eval(P'_{x=true}) \vee eval(P'_{x=false})$
3: **end if**
4: **if** $P = \neg P'$ **then**
5: **return** $\neg eval(P')$
6: **end if**
7: **if** $P = P' \wedge P''$ **then**
8: **return** $eval(P') \wedge eval(P'')$
9: **end if**
10: $\{P = x$ is the only remaining possibility, i.e., P is a variable$\}$
11: **return** $value(x)$

RIST

We now formally introduce the reachability problem in tree-like interaction systems. For Sys $\in \mathrm{IS}_{\mathrm{tree}}$ let $Q(\mathrm{Sys})$ be the state space of the global behavior

of Sys and $Reach(\text{Sys}) \subseteq Q(\text{Sys})$ be the set of reachable states in the global behavior. Let

$$RIST = \bigcup_{\text{Sys} \in \text{IS}_{\text{tree}}} (\{\text{Sys}\} \times Q(\text{Sys}))\,.$$

For $(\text{Sys}, q) \in RIST$ we want to decide whether q is reachable in the global behavior of Sys. Let $TRIST \subseteq RIST$ be the set of $RIST$ instances of the form (Sys, q) where q is reachable in the global behavior of Sys, i.e.,

$$TRIST = \bigcup_{\text{Sys} \in IST} (\{\text{Sys}\} \times Reach(\text{Sys}))\,.$$

Deciding $(\text{Sys}, q) \in TRIST$ is in PSPACE. Given a tree-like interaction system and a global state q one can guess a sequence of interactions (because PSPACE=NPSPACE [Sav70]) and check in linear space if it leads from the initial state q^0 to q. At any time we store exactly one global state from which we guess a successor state.

A Mapping from QBF to Tree-Like Interaction Systems

In the following we introduce for a QBF H a tree-like interaction system Sys_H (with an interaction model IM_H) and a global state q^t such that

i) $H \in TQBF \Leftrightarrow (\text{Sys}_H, q^t) \in TRIST$ and

ii) the size of Sys_H is polynomial in the size of H.

The idea for the construction of Sys_H can be sketched as follows: the interaction system basically simulates the evaluation of the formula H, as in Algorithm 1, based on the syntax tree of H. The subformulas of H are the components of the system, and the interaction model describes the propagation of truth values between the nodes of the syntax tree. Example 2.4 shows the interaction graph of an interaction model IM_H with respect to a QBF H that results from our reduction. If a subformula that models an existential quantifier is called recursively during the evaluation of Algorithm

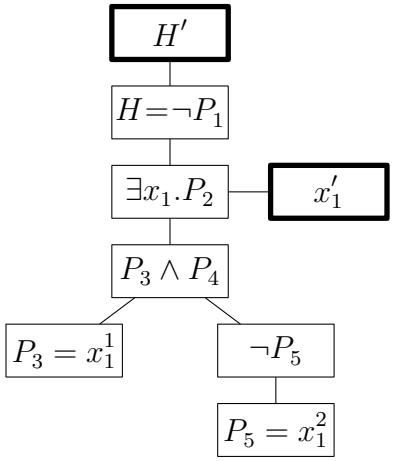


Figure 2.5: Interaction graph G_H of IM_H in Example 2.4.

1 then a truth value is assigned to all occurrences of the respective quantified variable. In Example 2.4 this would suggest a communication between the component that models the subformula $\exists x_1.P_2$ and the components that model x_1^1 respectively x_1^2, i.e., modeling this interaction would result in an interaction model such that the interaction graph of which is not a tree. The idea to avoid these communications is to store the truth assignment in an auxiliary component (x_1' in Example 2.4) and propagate the respective assignment down the tree to the components that model the respective variables.

Example 2.4:
Consider the formula $H = \neg\exists x_1.(x_1 \wedge \neg x_1)$. The associated interaction graph G_H of IM_H is given in Figure 2.5. The syntax tree of H is contained in G_H. Components with highlighted frames denote components that do not model subformulas of H.

We now describe in detail how Sys_H is constructed. First we introduce the components of Sys_H, then the ports of the components, followed by the specification of the interactions and the local behavior of the components.

Components

Let H be a QBF over the variables $x_1, \ldots, x_n$. Generally, there may be several occurrences of a variable x_i in H. Let x_i occur k_i times for $i = 1, \ldots, n$ as a subformula in H, then we assume that the jth occurrence of variable x_i is renamed in H as x_i^j for $1 \leq j \leq k_i$, e.g., in Example 2.4 the formula $\neg\exists x_1.(x_1 \wedge \neg x_1)$ is renamed to $\neg\exists x_1.(x_1^1 \wedge \neg x_1^2)$.

Let $\mathrm{K}_H = \mathrm{K}_1 \cup \mathrm{K}_2 \cup \{H'\}$ be a set of components such that

- $\mathrm{K}_1 = \{P | P \text{ is a subformula of } H\}$ and
- $\mathrm{K}_2 = \{x_1', x_2', \ldots, x_n'\}$.

In context of our intention to model the evaluation of Algorithm 1, the components in K_2 store the current truth assignment to the variables during the evaluation. The component H' is an auxiliary component that is used to model the initialization and the termination of Algorithm 1. The components in K_1 model the subformulas of H and have the same name as the subformulas. In the following it is clear from the context whether we speak about a subformula P of H or the component P that models this subformula.

Given a truth assignment to the variables, subformulas are evaluated to true or to false. Therefore, when we mention an assignment to a component in K_H then we refer to the current truth assignment to the respective variable respectively the current evaluation of the respective subformula that is modeled by this component.

In the following we introduce the port sets of the components in K_H. Many ports of different components model the same functionality and only differ in their subscripts.

Port sets of components modeling subformulas

Each component $P \in K_1$ that models a subformula P has to offer the following functionalities in form of ports.

- $\mathfrak{a}_P$ abbreviates "activate P": induces the evaluation of all subformulas in P with respect to the current truth assignment to the variables.
- t_P respectively f_P: the current truth assignment of P is true respectively false.
- $r_P x_l t$ ($r_P x_l f$) abbreviates "P receives instruction to set x_l true (false)": all occurrences of the variable x_l in P shall be set to true (false).
- $r_P t$: assign true to this subformula.

The following ports are offered by components that model subformulas P that are not variables, i.e., P consists of an operator and one subformula P_1 (e.g., $P = \neg P_1$) or two subformulas (e.g., $P = P_1 \wedge P_2$).

- $\mathfrak{e}_P^1$ ($\mathfrak{e}_P^2$) abbreviates "evaluate P_1 (P_2)": evaluate P_1 (P_2) with respect to the current truth assignment to the variables.
- $sub_P^1 t$ ($sub_P^2 t$) respectively $sub_P^1 f$ ($sub_P^2 f$) abbreviates "subformula P_1 (P_2) is true respectively false": P_1 (P_2) was evaluated to true respectively false.
- t_P respectively f_P: P was evaluated true respectively false with respect to the current truth assignments of the variables.
- $s_P^1 x_l t$ ($s_P^2 x_l t$) respectively $s_P^1 x_l f$ ($s_P^2 x_l f$) abbreviates "set x_l true (false) in P_1 respectively P_2"): all occurrences of the variable x_l in P_1 (P_2) shall be set to true respectively false.
- $s_P^1 t$ ($s_P^2 t$): assign true to P_1 (P_2).

For $i = 1, \ldots, n$ and $j = 1, \ldots, k_i$ the component $P = x_i^j \in \mathrm{K}_1$ represents the jth occurrence of variable x_i in H. The set A_P of ports is given by

$$A_P = \{\mathfrak{a}_P, t_P, f_P, r_P t\} \cup \{r_P x_l t, r_P x_l f | l = 1, \ldots, n\}.$$

A component modeling a negation, i.e., a subformula of the form $P = \neg P_1$ has the following set of ports

$$\begin{aligned} A_P &= \{\mathfrak{e}_P^1, \mathfrak{a}_P, sub_P^1 t, sub_P^1 f, t_P, f_P, r_P t, s_P^1 t\} \cup \\ &\quad \{r_P x_l t, r_P x_l f, s_P^1 x_l t, s_P^1 x_l f | l = 1, \ldots, n\}. \end{aligned}$$

A component that models a conjunction, i.e., a subformula of the form $P = P_1 \wedge P_2$ has the set of ports

$$\begin{aligned} A_P &= \{\mathfrak{a}_P, \mathfrak{e}_P^1, \mathfrak{e}_P^2, sub_P^1 t, sub_P^1 f, sub_P^2 t, sub_P^2 f, t_P, f_P, r_P t, s_P^1 t, s_P^2 t\} \cup \\ &\quad \{r_P x_l t, r_P x_l f, s_P^1 x_l t, s_P^1 x_l f, s_P^2 x_l t, s_P^2 x_l f | l = 1, \ldots, n\}. \end{aligned}$$

A component that models a subformula of the form $P = \exists x_i.P_1$ $(1 \leq i \leq n)$ needs to have access to the current truth assignment of the variable x_i. We store the current truth assignment of the variable x_i, that can occur multiple times in the QBF H, in the behavior of the component $x_i' \in \mathrm{K}_2$. This component will exclusively interact with the component that models $P = \exists x_i.P_1$. The set of ports $A_{x_i'}$ is given by

$$A_{x_i'} = \{r x_i t, r x_i f, t_{x_i}, f_{x_i}\}.$$

t_{x_i} respectively f_{x_i} models that true respectively false is assigned to all occurrences of variable x_i. The port $r x_i t$ assigns true to x_i'. Analogously $r x_i f$ switches the assignment to false.

The port set A_P for $P = \exists x_i.P_1$ is given by

$$\begin{aligned} A_P &= \{\mathfrak{a}_P, \mathfrak{e}_P^1, sub_P^1 t, sub_P^1 f, t_P, f_P, x_i t, x_i f, s x_i t, s x_i f, r_P t, s_P^1 t\} \cup \\ &\quad \{r_P x_l t, r_P x_l f, s_P^1 x_l t, s_P^1 x_l f | l = 1, \ldots, n\}. \end{aligned}$$

$\mathfrak{a}_P$, $\mathfrak{e}^1_P$, $sub^1_P t$, $sub^1_P f$, t_P and f_P act similarly to the corresponding ports of the other components specified above. The port $x_i t$ confirms that true is assigned to all occurrences of the variable x_i, and $sx_i t$ models that true is assigned to all occurrences of x_i. On the other hand $x_i f$ confirms that false is assigned to x_i, and $sx_i f$ assigns false to x_i.

The component H' models the call to the algorithm *eval* and the termination. The set of ports $A_{H'}$ is given by

$$A_{H'} = \{\mathfrak{e}^1_{H'}, sub^1_{H'} t, sub^1_{H'} f, s^1_{H'} t, end_{H'}\}.$$

All ports but $end_{H'}$ have the same functionality as the ports described above. We show that the formula H is in $TQBF$ if and only if the component associated with H is evaluated to true, i.e., $sub^1_{H'} t$ becomes enabled eventually. The port $end_{H'}$ models that the simulated evaluation of Algorithm 1 terminated. This port shall become enabled, either if H was evaluated to true or to false. We use this port in order to prevent Sys_H from being stuck after the evaluation of H.

Interactions

We now define the interaction set Int of IM_H. Let $P \in \mathrm{K}_1 \cup \{H'\}$ model a subformula which is not an occurrence of a variable. The component P can model a subformula that consists of an operator and one subformula P_1 (e.g., $P = \neg P_1$) or two subformulas P_1 and P_2 (e.g., $P = P_1 \wedge P_2$). If P needs the truth value of P_k, $k \in \{1, 2\}$, to be evaluated then the evaluation in P_k needs to be activated. This is realized by the synchronization of $\mathfrak{e}^k_P$ and $\mathfrak{a}_{P_k}$. Furthermore P can ask P_k for its current truth value. These interactions are realized by

$$\begin{aligned} eval\_P \to P_k &= \{\mathfrak{e}^k_P, \mathfrak{a}_{P_k}\} \\ P\_ask\_P_k\_true &= \{sub^k_P t, t_{P_k}\} \\ P\_ask\_P_k\_false &= \{sub^k_P f, f_{P_k}\} \end{aligned}$$

for $k \in \{1, 2\}$. These interactions connect all components in $\mathrm{K}_1 \cup \{H'\}$ and result in an interaction graph that is related to the syntax tree of the QBF H.

If a subformula of the form $P = \exists x_i.P_1$ $(1 \leq i \leq n)$ needs all occurrences of variable x_i to be set to true or to false a direct interaction with the components that model these variables would lead to a cycle in the associated interaction graph. Therefore, P passes this information to its subformula P', i.e., $s^1_P x_i t$ in P has to synchronize with $r_{P_1} x_i t$ in P_1. The component P_1 needs to pass on this information to the components that model the subformulas of P_1 and so on. Let $i \in \{1, \ldots, n\}$ and $k \in \{1, 2\}$. The following interactions realize the synchronizations needed to propagate the information to assign a truth value to a variable.

$$\begin{aligned} set\_x_i\_true\_P \to P_k &= \{s^k_P x_i t, r_{P_k} x_i t\} \\ set\_x_i\_false\_P \to P_k &= \{s^k_P x_i f, r_{P_k} x_i f\} \end{aligned}$$

If the QBF H is true, we need all components to be in one designated state – whether or not this global state is reachable corresponds to whether or not H is true. If H was evaluated to true then the component H' reaches a designated local state. To assure that all components can reach a corresponding designated state, a similar technique as above is used. We propagate, starting in the component H', the information through the tree that all components shall reach their designated state that indicates that H is true. A component P that models a subformula propagates this information to the components that model the subformulas of P by the following interaction. For $k \in \{1, 2\}$ let

$$set\_P_k\_true\_P \to P_k = \{s^k_P t, r_{P_k} t\}.$$

Consider a subformula of the form $P = \exists x_i.P_1 \in \mathrm{K}_1$ and the associated component $x'_i \in \mathrm{K}_2$. The component that models P can assign true or false to x'_i and can ask x'_i whether the current truth assignment is true or false.

This is realized by

$$\begin{aligned} set\_x_i'\_true &= \{sx_it, rx_it\} \\ ask\_true_{x_i'} &= \{x_it, t_{x_i}\} \\ set\_x_i'\_false &= \{sx_if, rx_if\} \\ ask\_false_{x_i'} &= \{x_if, f_{x_i}\} \end{aligned}$$

We model Sys_H such that if H' reaches a state that indicates that H was evaluated to true or false, i.e., the simulation of the evaluation of H is finished, then the unary interaction $evaluated = \{end_{H'}\}$ becomes enabled in the global behavior of Sys_H.

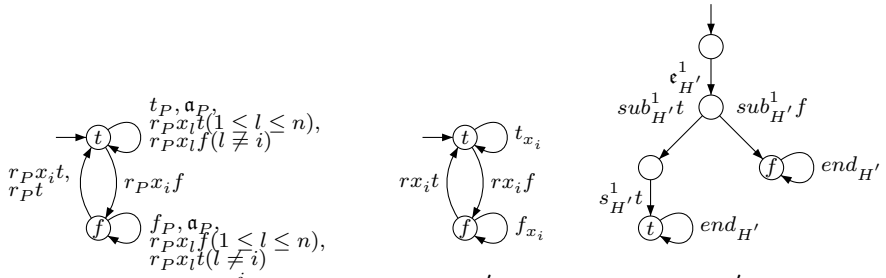
Let Int be the set of interactions given by

$\{eval\_P \to P_k | P \in \mathrm{K}_1 \cup \{H'\}$ with succ. $P_k\} \cup$
$\{P\_ask\_P_k\_true | P \in \mathrm{K}_1 \cup \{H'\}$ with succ. $P_k\} \cup$
$\{P\_ask\_P_k\_false | P \in \mathrm{K}_1 \cup \{H'\}$ with succ. $P_k\} \cup$
$\{set\_x_i'\_true, set\_x_i'\_false, ask\_true_{x_i'}, ask\_false_{x_i'} | x_i' \in \mathrm{K}_2\} \cup$
$\{set\_P_k\_true\_P \to P_k | P \in \mathrm{K}_1 \cup \{H'\}$ with succ. $P_k\} \cup$
$\{set\_x_i\_true\_P \to P_k | P \in \mathrm{K}_1$ with succ. $P_k, i \in \{1, \ldots, n\}\} \cup$
$\{set\_x_i\_false\_P \to P_k | P \in \mathrm{K}_1$ with succ. $P_k, i \in \{1, \ldots, n\}\} \cup$
$\{evaluated\}$.

Remark 2.3:
The interaction graph G_H, associated to IM_H, is a tree, as it is constructed along the syntax tree and augmented with the components H' and x_i' for $1 \leq i \leq n$ without forming cycles.

Local Behavior

The local behavior of the components is given by labeled transition systems. Every system has one state labeled t and one labeled f. These states model the fact that either true respectively false is assigned to this component or it was evaluated to true respectively false.



(a) $P = x_i^j$ (b) x_i' (c) H'

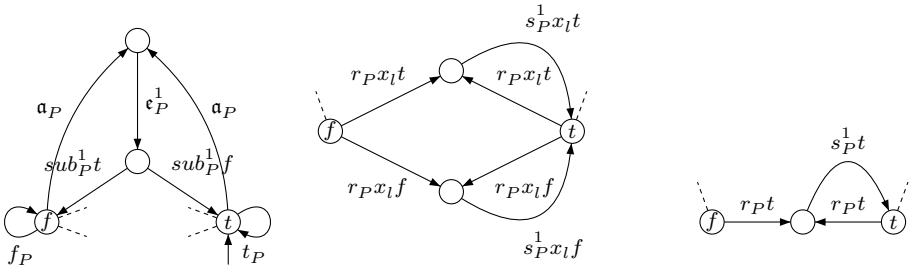
Figure 2.6: Local behavior $T_{x_i^j}$ for a component x_i^j (2.6a), $T_{x_i'}$ for x_i' (2.6b) and $T_{H'}$ for the component H' (2.6c).

Figure 2.6a depicts the transition system of the component modeling the jth occurrence of variable x_i. Figure 2.6b gives the local behavior of a component $x_i' \in \mathrm{K}_2$. The behavior of H' is given in Figure 2.6c. The transition systems for a component that models a variable x_i^j respectively for $x_i' \in \mathrm{K}_2$ are self-explanatory. If in $T_{H'}$ the port $\mathfrak{e}^1_{H'}$ is performed, i.e., component H needs to be evaluated, then $T_{H'}$ waits to perform either $sub^1_{H'} t$ or $sub^1_{H'} f$. These ports can only be performed if T_H reaches its state labeled t respectively f. We show that this indicates whether the associated QBF is true respectively false.

In Figure 2.7 the local behavior for a component of the form $P = \neg P_1$ is depicted. Note that, for better readability, the transition system in Figure 2.7a is not completely shown. In Figure 2.7a the transitions and states displayed in Figure 2.7b and 2.7c have to be included between the states labeled t and f for $l = 1, \ldots, n$.

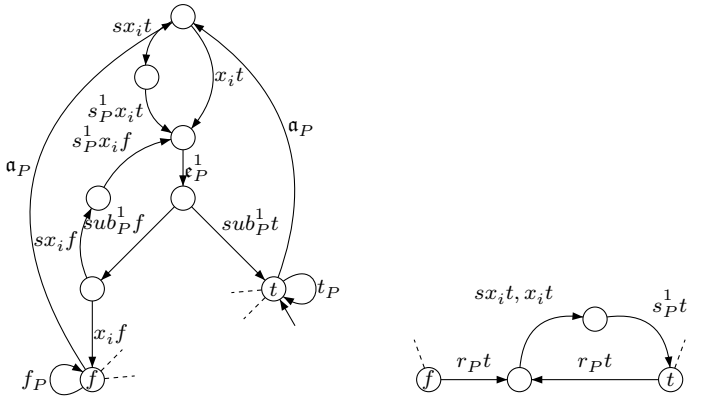
In Figure 2.8 the local behavior for a component of the form $P = \exists x_i.P_1$ is depicted. For better readability, the transition system in Figure 2.8a is not completely shown. In Figure 2.8a the transitions and states displayed in Figure 2.7b and 2.8b have to be included between the states labeled t and f for $l = 1, \ldots, n$.

In Figure 2.9 the local behavior for a component of the form $P = P_1 \wedge P_2$ is



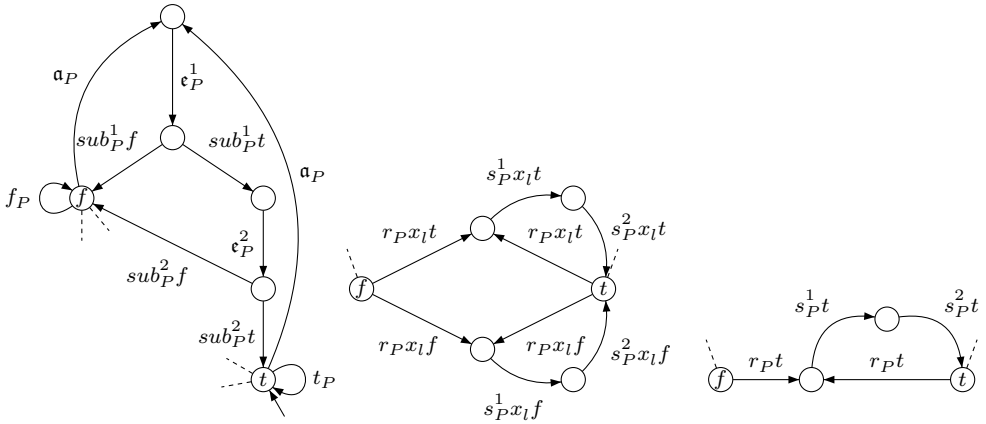
(a) $P = \neg P_1$ (b) Section of 2.7a and 2.8a, for $1 \leq l \leq n$ (c) Section of 2.7a

Figure 2.7: Main section of the transition systems $T_{\neg P_1}$ (2.7a), part of $T_{\neg P_1}$ for $1 \leq l \leq n$ (2.7b) and part of $T_{\neg P_1}$ (2.7c).



(a) $P = \exists x_i.P_1$ (b) Section of 2.8a

Figure 2.8: Main sections of the transition system $T_{\exists x_i.P_1}$ (2.8a) and part of $T_{\exists x_i.P_1}$ (2.8b).



(a) $P = P_1 \wedge P_2$ (b) Section of 2.9a, for $1 \leq l \leq n$ (c) Section of 2.9a

Figure 2.9: Main sections of the transition system $T_{P_1 \wedge P_2}$ (2.9a), part of $T_{P_1 \wedge P_2}$ (2.9b) and part of $T_{P_1 \wedge P_2}$ (2.9c).

depicted. Note that the transition system in Figure 2.9a is not completely shown. The transitions and states displayed in Figure 2.9b and 2.9c have to be included between the states labeled t and f for $l = 1, \ldots, n$.

The resulting interaction system is denoted by $\mathrm{Sys}_H = (\mathrm{IM}_H, \{T_P\}_{P \in \mathrm{K}_H})$.

$H \in TQBF \Leftrightarrow (\mathrm{Sys}_H, q^t) \in TRIST$

In order to show that the reachability problem in tree-like interaction systems is PSPACE-hard we need to show that a quantified Boolean formula H is true if and only if a designated state is reachable in the global behavior of Sys_H. This proposition is formulated in the following theorem.

Theorem 2.1:
Let H be a QBF over the grammar $P ::= x | \neg P | P \wedge P | \exists x.P$ and Sys_H the associated interaction system obtained from the reduction. Let q^t be the global state in which all components are in their state labeled t, then

$$H \in TQBF \Leftrightarrow (\mathrm{Sys}_H, q^t) \in TRIST.$$

Proof. The proof can be found in Appendix A on Page 183. □

2.4 Reachability of Local States

In the previous section we discussed the complexity theoretic classification of the reachability problem in tree-like interaction systems by introducing a reduction from the trueness problem of quantified Boolean formulas. Here we consider a special kind of the reachability problem in tree-like interaction systems. Given a tree-like interaction system Sys with a set of components K, a component $i \in K$ and a local state q_i in the behavior T_i of component i we discuss the question whether there is a state q reachable in the global behavior T of Sys where component i is in the state q_i.

We show that deciding this problem is PSPACE-complete in tree-like interaction systems. The reduction uses parts of the proof of Theorem 2.1.

First we formulate the respective decision problem.

Let $\text{Sys} = (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\text{IM} = (K, \{A_i\}_{i \in K}, Int)$. We assume that the local state spaces of the components in Sys are pairwise disjoint. Let $Q_{loc}(\text{Sys}) = \bigcup_{i \in K} Q_i$, i.e., $Q_{loc}(\text{Sys})$ is the union of all local state spaces of the components in Sys. Remember that IS_{tree} is the class of tree-like interaction systems. Let

$$RISTL = \bigcup_{\text{Sys} \in \text{IS}_{\text{tree}}} (\{\text{Sys}\} \times Q_{loc}(\text{Sys}))\,.$$

For $(\text{Sys}, q_{loc}) \in RISTL$ let $q_{loc} \in Q_i$ for some $i \in K$. We want to decide whether a global state $q \in Q$ is reachable in the global behavior of Sys such that component i is in state q_{loc} in q. Let $TRISTL \subseteq RISTL$ be the set of $RISTL$ instances where this is the case.

Let $\mathrm{Sys} \in \mathrm{IS}_{\mathrm{tree}}$ be a tree-like interaction system with a set of components K. For $i \in K$ let $\mathit{ReachLocal}_i(\mathrm{Sys}) \subseteq Q_i$ be the set of states $q_i \in Q_i$ such that there is a state q reachable in the global behavior of Sys where component i is in the state q_i. Let

$$\mathit{ReachLocal}(\mathrm{Sys}) = \bigcup_{i \in K} \mathit{ReachLocal}_i(\mathrm{Sys})$$

be the union over all reachable local states in Sys then

$$\mathit{TRISTL} = \bigcup_{\mathrm{Sys} \in \mathrm{IS}_{\mathrm{tree}}} (\{\mathrm{Sys}\} \times \mathit{ReachLocal}(\mathrm{Sys}))\,.$$

The following corollary relates the question whether $(\mathrm{Sys}, q_{loc}) \in \mathit{RISTL}$ is an instance of TRISTL to the question whether a quantified Boolean formula is true.

Corollary 2.1:
Let H be a QBF and Sys_H the associated interaction system obtained from the reduction given in Section 2.3. We assume that the local state spaces of the components in Sys_H are disjoint, e.g., each local state is indexed by the name of the respective component. Consider the state $t_{H'}$ of component H' (this state is labeled t in Figure 2.6c), then

$$H \in \mathit{TQBF} \Leftrightarrow (\mathrm{Sys}_H, t_{H'}) \in \mathit{TRISTL}.$$

Proof. The proof can be found in Appendix A on Page 194. □

Analogously to the argument why deciding the reachability problem in tree-like interaction systems is in PSPACE, it is easy to see that the problem of deciding TRISTL is in PSPACE as well. Let $(\mathrm{Sys}, q_{loc}) \in \mathit{RISTL}$ where K is the set of components in Sys and $q_{loc} \in Q_i$ for $i \in K$. We guess a sequence of interactions and check in linear space whether it leads from the global initial state q^0 to a state q where $q_i = q_{loc}$. Thus, by Corollary 2.1 follows that deciding whether there is a global state reachable where a fixed component is in a fixed state is PSPACE-complete.

2.5 PSPACE-completeness of Reachability in Linear Systems

In the following we give a reduction from the acceptance problem in linear bounded Turing machines to the reachability problem in linear interaction systems. This reduction strengthens the result of the QBF reduction as the subclass of component systems with a linear communication structure is a proper subset of systems with a tree-like communication structure. We use the following syntax for a Turing machine but we refrain from repeating the well known semantics (see [GJ79] for details).

Definition 2.2:
A tuple $M = (\Gamma, \Sigma, \mathbb{P}, \delta)$ is called **deterministic Turing machine** (**DTM**) with

- Γ is a finite set of **tape symbols**,
- $\Sigma \subseteq \Gamma$ is a set of **input symbols** with a distinguished **blank symbol** $b \in \Gamma \setminus \Sigma$,
- $\mathbb{P}$ is a finite set of **states**, including a distinguished **initial state** p^0 and two distinguished **halt states** p^Y and p^N and
- δ is the **transition function** of M, given by

$$\delta : (\mathbb{P} \setminus \{p^Y, p^N\}) \times \Gamma \to \mathbb{P} \times \Gamma \times \{-1, +1\}.$$

We consider a both-sided infinite tape with cells labeled by integers. Given an input $x \in \Sigma^*$ written on the cells labeled 1 through $|x|$ we assume M to be initially in the initial state p^0 and the tape head pointing at cell 1. For a string $x \in \Sigma^*$ with $|x| = n$ we denote the ith letter in x by x^i for $1 \leq i \leq n$.

A DTM M is called **linear bounded** if no computation on M uses more than $n+1$ tape cells, where n is the length of the input string. A **configuration** of a bounded DTM M is denoted by $(p; \gamma_0, \ldots, \underline{\gamma_i}, \ldots, \gamma_{n+1})$ where M is in

state p, γ_j is the tape symbol in cell $0 \leq j \leq n+1$ and the tape head is on cell i.

Definition 2.3:
The problem **linear space acceptance** (**LSA**) has as input a linear bounded DTM M and a finite string x over the input alphabet of M. The question is whether M accepts x, i.e., does M halt in the state p^Y.

It is well known that the problem LSA is PSPACE-complete [GJ79].

The idea for our reduction is to model the cells of a DTM M by components of an interaction system Sys_M and the transition function of M by interactions such that a path in the global behavior of Sys_M corresponds to an execution of M. In order for the transition function to calculate the next configuration of M we need the current position of the tape head, the current tape symbol in the respective cell and the current state of M. We model all these informations in each cell, i.e., in order to model the calculation of the next configuration we only need an interaction between the component that models the cell with the tape head and the respective components that model the neighboring cells.

Let $M = (\Gamma, \Sigma, \mathbb{P}, \delta)$ be a linear bounded DTM and $x \in \Sigma^*$ an input with $|x| = n$. Let $\mathrm{Sys}_M = (\mathrm{IM}_M, \{T_i\}_{i \in \mathrm{K}})$ be an interaction system with interaction model $\mathrm{IM} = (\mathrm{K}, \{A_i\}_{i \in \mathrm{K}}, \mathrm{Int})$ such that $\mathrm{K} = \{0, \ldots, n+1\}$.

The set of ports A_i for a component $i \in K$ with $1 \leq i \leq n$ is given by

$$A_i = \{(p,\gamma)_i^1, (p,\gamma)_i^2 | p \in \mathbb{P} \setminus \{p^Y, p^N\}, \gamma \in \Gamma\}.$$

A port $(p,\gamma)_i^1$ models that the tape head moves away from cell i where γ is the current tape symbol in this cell and M is in state p. Analogously, $(p,\gamma)_i^2$ models that the tape head moves onto cell i where γ is written on this cell and M is in state p.

Because of M being linear bounded, we now that δ does not move the tape head from cell 0 to the left respectively from cell $n+1$ to the right. Thus,

we can omit ports from A_0 and A_{n+1} that model a head movement from or onto cell -1 and $n+2$. The set of ports A_0 is given by

$$\begin{aligned} A_0 \;=\; & \{(p,\gamma)_0^1 | p \in \mathbb{P} \setminus \{p^Y, p^N\}, \gamma \in \Gamma, \neg\exists_{p',\gamma'} \delta(p,\gamma) = (p',\gamma',-1)\} \cup \\ & \{(p,\gamma)_0^2 | p \in \mathbb{P} \setminus \{p^Y, p^N\}, \gamma \in \Gamma, \neg\exists_{p',\gamma'} \delta(p,\gamma) = (p',\gamma',1)\}. \end{aligned}$$

Analogously, for component $n+1 \in \mathrm{K}$, i.e., for the rightmost used cell let

$$\begin{aligned} A_{n+1} \;=\; & \{(p,\gamma)_{n+1}^1 | p \in \mathbb{P} \setminus \{p^Y, p^N\}, \gamma \in \Gamma, \neg\exists_{p',\gamma'} \delta(p,\gamma) = (p',\gamma',1)\} \cup \\ & \{(p,\gamma)_{n+1}^2 | p \in \mathbb{P} \setminus \{p^Y, p^N\}, \gamma \in \Gamma, \neg\exists_{p',\gamma'} \delta(p,\gamma) = (p',\gamma',-1)\}. \end{aligned}$$

The set of interactions is given by

$$\mathrm{Int} = \{\{(p,\gamma)_i^1, (p,\gamma)_{i+\mathbb{T}}^2\} | \exists_{p',\gamma'} \delta(p,\gamma) = (p',\gamma',\mathbb{T}), 0 \leq i + \mathbb{T} \leq n+1\}.$$

For $i \in \mathrm{K}$ let $T_i = (Q_i, A_i, \rightarrow_i, q_i^0)$ be the local behavior of component i with $Q_i = \{(p,\gamma) | p \in \mathbb{P} \cup \{s\}, \gamma \in \Gamma\}$ where s is an auxiliary symbol that is not included in $\mathbb{P}$. The port $(p,\gamma) \in Q_i$ with $p \neq s$ models that the tape head is currently on cell i and the current tape symbol in this cell is γ. The port (s,γ) models that γ is the content of cell i and the tape head is not on this cell. The local initial states are derived from the initial word on the tape, i.e., $q_0^0 = (s,b)$, $q_1^0 = (p^0, x^1)$, $q_i^0 = (s, x^i)$ for $2 \leq i \leq n$ and $q_{n+1}^0 = (s,b)$. For $i \in \mathrm{K}$ let $\rightarrow_i$ be the union of the following transitions.

a) For all $\gamma, \gamma' \in \Gamma$ and $p \in \mathbb{P} \setminus \{p^Y, p^N\}$ let $(p,\gamma) \xrightarrow{(p,\gamma)_i^1}_i (s,\gamma')$ if there are $p' \in \mathbb{P}$ and $\mathbb{T} \in \{-1,1\}$ such that $\delta(p,\gamma) = (p',\gamma',\mathbb{T})$.

b) For all $\gamma, \tilde{\gamma} \in \Gamma$, $p \in \mathbb{P} \setminus \{p^Y, p^N\}$ and $p' \in \mathbb{P}$ let $(s,\tilde{\gamma}) \xrightarrow{(p,\gamma)_i^2}_i (p',\tilde{\gamma})$ if there are $\gamma' \in \Gamma$ and $\mathbb{T} \in \{-1,1\}$ such that $\delta(p,\gamma) = (p',\gamma',\mathbb{T})$.

The transitions described in a) model the impact of the transition function δ on cell i if the tape head is currently on this cell. Let M be in state p and the tape head on cell i reading γ, i.e., T_i is in the state (p,γ). If $\delta(p,\gamma) = (p',\gamma',\mathbb{T})$ then γ' is written and the tape head moves to a neighboring cell, i.e., T_i moves to the state (s,γ'). On the other hand, the transitions described

in b) model the impact of δ on cell i if the tape head moves onto this cell. Let $\tilde{\gamma}$ be the current tape symbol on cell i, i.e., T_i is in state $(s, \tilde{\gamma})$ before the head moves. After the movement let M changes its state to p', i.e., T_i moves to the state $(p', \tilde{\gamma})$.

Remark 2.4:
Note that in each component, local states of the form (p^Y, γ) and (p^N, γ) do not have outgoing transitions, i.e., a component that reaches one of these states can never interact again. We can avoid this situation by specifying a self-loop on these states that is labeled by a distinguished port. An interaction that consists of only this one port is permanently enabled if one of these states is reached.

Remark 2.5:
Sys_M satisfies the conditions of an interaction system: every port of a component occurs in at least one interaction. Let $i \in \mathrm{K}$, $(p, \gamma)_i^1 \in A_i$ and $\delta(p, \gamma) = (p', \gamma', \mathbb{T})$ then $0 \leq i + \mathbb{T} \leq n + 1$ and $\{(p, \gamma)_i^1, (p, \gamma)_{i+\mathbb{T}}^2\} \in \mathrm{Int}$. For $(p, \gamma)_i^2 \in A_i$ is $0 \leq i - \mathbb{T} \leq n + 1$ and $\{(p, \gamma)_{i-\mathbb{T}}^1, (p, \gamma)_i^2\} \in \mathrm{Int}$.

It is clear that Sys_M has a linear communication structure because every component $1 \leq i \leq n$ only interacts with its neighboring components $i - 1$ and $i + 1$.

Remark 2.6:
The reduction is polynomial, since $|\,\mathrm{Int}\,| \leq |\mathbb{P}| \cdot |\Gamma|$ and for all $i \in \mathrm{K}$ $|A_i| \leq 2 \cdot |\mathbb{P}| \cdot |\Gamma|$ and $|Q_i| \leq (|\mathbb{P}| + 1) \cdot |\Gamma|$.

Theorem 2.2:
Let $M = (\Gamma, \Sigma, \mathbb{P}, \delta)$ be a linear bounded DTM, $x \in \Sigma^*$ with $|x| = n$ an input for M and Sys_M the associated linear interaction system. We have M accepts x if and only if a global state $q = (q_0, \ldots, q_{n+1})$ is reachable in Sys_M such that there is $i \in \{0, \ldots, n + 1\}$ with $q_i = (p^Y, \gamma)$ for a tape symbol $\gamma \in \Gamma$.

Proof. The proof can be found in Appendix A on Page 194. □

Remark 2.7:
An instance of the reachability problem in linear interaction systems is a linear interaction system Sys and a global state q. The interaction system Sys_M for a linear bounded DTM M and an input x can be extended such that a distinguished global state can be reached if M halts on x. This can be achieved by the technique that was used in Section 2.3 in order to reach a distinguished global state in tree-like interaction systems. The idea is to invoke, starting from the component that reached (p^Y, γ), that each component shall reach a distinguished state. This invocation can be propagated through neighboring components.

The reachability problem in linear interaction systems is in PSPACE because each linear interaction system is particularly a tree-like interaction system and the reachability problem in tree-like interaction systems is in PSPACE. It follows by Theorem 2.2 and Remark 2.7 that the reachability problem in linear interaction systems is PSPACE-complete.

2.6 PSPACE-completeness of Reachability in Star-Like Systems

Here we show that deciding the reachability problem in the class of star-like interaction systems is PSPACE-complete. We show this by providing a reduction from a general interaction systems Sys to a star-like systems Sys$'$. The idea of the reduction is to construct a "control component" *cc* that forms the center of the star structure in Sys$'$ and is surrounded by the components of Sys. An interaction in Sys is modeled by multiple interactions in Sys$'$ where each consists of exactly two ports. The execution of an interaction in Sys then corresponds to the execution of a sequence of interactions in Sys$'$ that is coordinated by *cc* and achieved in two steps. Let α be an interaction in Sys.

a) In a first step cc interacts with each component that participates in α and checks whether the respective port in α is enabled without changing the local states of the components. If this check fails then cc returns to its initial state.

b) If the check succeeds then cc interacts with each respective component on the ports in α, i.e., a global transition in Sys that is labeled by α is simulated.

Let $Q = \prod_{i \in \mathrm{K}} Q_i$ be the global state space of Sys then we have a global state space $\prod_{i \in \mathrm{K} \cup \{cc\}} Q_i$ for Sys$'$ with the property that $q \in Q$ is reachable in Sys if and only if a state q' is reachable in Sys$'$ such that q' equals q up to the local state of the component cc. Since reachability in general interaction systems is PSPACE-complete, the consequence of this transformation is the PSPACE-completeness of reachability in star-like interaction systems.

Let $\mathrm{Sys} = (\mathrm{IM}, \{T_i\}_{i \in \mathrm{K}})$ be an interaction system with interaction model $\mathrm{IM} = (\mathrm{K}, \{A_i\}_{i \in \mathrm{K}}, \mathrm{Int})$ and $\mathrm{Sys}' = (\mathrm{IM}', \{T_i'\}_{i \in \mathrm{K}'})$ be an interaction system with interaction model $\mathrm{IM} = (\mathrm{K}', \{A_i'\}_{i \in \mathrm{K}'}, \mathrm{Int}')$.

Let $\mathrm{K}' = \mathrm{K} \cup \{cc\}$, where cc is a control component that coordinates sequences of interactions in Int$'$ that correspond to interactions in Int. For $i \in K$ let $A_i' = A_i \cup \{a_i^{ok}, a_i^{\neg ok} | a_i \in A_i\}$. The port a_i^{ok} respectively $a_i^{\neg ok}$ models that component i enables respectively does not enable the port $a_i \in A_i$. The set of ports A_{cc} of component cc is given by

$$A_{cc} = \{a\_i_{cc}^{ok}, a\_i_{cc}^{\neg ok}, a\_i_{fire}^{cc} | i = 1, \ldots, n, a_i \in A_i\} \cup \{\alpha_{cc} | \alpha \in \mathrm{Int}\}.$$

Let $i \in K$ and $a_i \in A_i$ a port in i then $a\_i_{cc}^{ok}$ models that component i currently enables a_i and $a\_i_{cc}^{\neg ok}$ models that a_i is currently not enabled by i. The port $a\_i_{fire}^{cc}$ models that component i performs a transition labeled by a_i. For an interaction $\alpha \in \mathrm{Int}$ the port α_{cc} models the initiation of a process that checks whether α is enabled by the respective components and, if applicable, coordinates that all ports in α interact one after another.

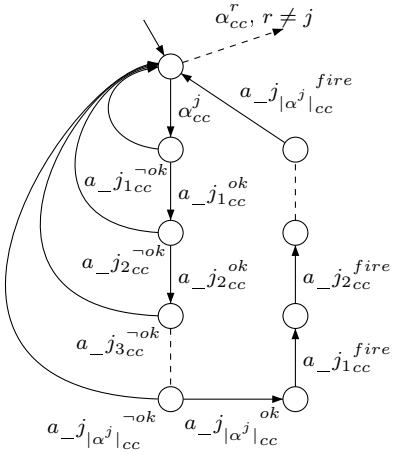


Figure 2.10: Parts of the local behavior of component cc.

The set of interactions Int' is given by

$$\begin{aligned}\text{Int}' \;=\; & \{\{a_i^{ok}, a\_i_{cc}^{ok}\}, \{a_i^{\neg ok}, a\_i_{cc}^{\neg ok}\}, \{a_i, a\_i_{cc}^{fire}\} | a_i \in A_i, i = 1, \ldots, n\} \cup \\ & \{\{\alpha_{cc}\} | \alpha \in \text{Int}\}.\end{aligned}$$

The local behavior of $i \in \text{K}$ is given by $T_i' = (Q_i, A_i', \rightarrow_i', q_i^0)$ with

$$\begin{aligned}\rightarrow_i' = \; & \rightarrow_i \cup \{(q_i, a_i^{ok}, q_i) | q_i \in Q_i \wedge a_i \in en(q_i)\} \cup \\ & \{(q_i, a_i^{\neg ok}, q_i) | q_i \in Q_i \wedge a_i \notin en(q_i)\}.\end{aligned}$$

T_i' extends T_i such that for each port $a_i \in A_i$ there is a self-loop on each state $q_i \in Q_i$ that is labeled by a_i^{ok} if q_i enables a_i and by $a_i^{\neg ok}$ otherwise. These transitions are used to check whether or not each port of an interaction $\alpha \in \text{Int}$ is enabled in a global state of Sys' without changing the local state of the respective components.

Let $\text{Int} = \{\alpha^1, \alpha^2, \ldots, \alpha^k\}$ and $\alpha^j = \{a_{j_1}, \ldots, a_{j_{|\alpha^j|}}\}$ for $j \in \{1, 2 \ldots, k\}$. Figure 2.10 depicts the local behavior $T_{cc} = (Q_{cc}, A_{cc}, \rightarrow_{cc}, q_{cc}^0)$ of component cc that coordinates a test that checks whether each port in α^j is enabled in Sys' and, if applicable, enables ports that can interact with each port in α^j.

Remark 2.8:
Each port of Sys' occurs in at least one interaction, i.e., Sys' satisfies the

conditions of an interaction system. Furthermore, the size of Sys$'$ is polynomial in the size of Sys because $|K'| = |K| + 1$, $|\,\mathrm{Int}'\,| = |\,\mathrm{Int}\,| + \sum_{i \in K} 3 \cdot |A_i|$ and for $i \in K$ holds $|A_i'| = 3 \cdot |A_i|$ and $|\,\rightarrow_i'\,| = |\,\rightarrow_i\,| + |Q_i| \cdot |A_i|$. For $cc \in K'$ holds $|A_{cc}| = |\,\mathrm{Int}\,| + \sum_{i \in K} 3 \cdot |A_i|$, $|Q_{cc}| = 1 + \sum_{\alpha \in \mathrm{Int}} 2 \cdot |\alpha|$ and $|\,\rightarrow_{cc}\,| = \sum_{\alpha \in \mathrm{Int}} (3 \cdot |\alpha| + 1)$.

Theorem 2.3:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\mathrm{IM} = (K, \{A_i\}_{i \in K}, Int)$ and Sys$'$ the associated star-like interaction system. A global state q is reachable in Sys if and only if a global state q' is reachable in Sys$'$ such that $q_i = q_i'$ for $i \in K$ and $q_{cc}' = q_{cc}^0$.

Proof. The proof can be found in Appendix A on Page 195. □

The reachability problem in star-like interaction systems is in PSPACE because each star-like interaction system is particularly a tree-like interaction system and the reachability problem in tree-like interaction systems is in PSPACE. It follows by Theorem 2.3 that the reachability problem in star-like interaction systems is PSPACE-complete.

2.7 PSPACE-completeness of Progress in Systems with a Restricted Communication Structure

Given an interaction system Sys with components K and a component $k \in K$, an interesting question is, whether or not there are reachable states in the global behavior of Sys from which k may never participate again in an interaction. If, for example, components in an interaction system model processes that can send requests among each other which need to be answered by a respective response then an obvious question is whether or not there are reachable global states where a request is never answered by

a response. The progress problem consists of the question whether or not a component $k \in K$ has to participate in infinitely many interactions from every reachable state in the global behavior of Sys. Of course, if there is a reachable global state where no interaction is enabled then this question is obsolete because from such a state no further interaction is enabled in which k may or may not participate. In [MCM08c] it was shown that in general interaction systems deciding progress is PSPACE-complete.

By a minor modification of the reduction given in Section 2.3 it is possible to show that deciding progress in tree-like interaction systems is PSPACE-complete as well. First we provide some definitions in order to introduce progress in interaction systems. We proceed by arguing why deciding this property in the subclass of tree-like interaction systems is PSPACE-complete. Furthermore, we provide two remarks about the PSPACE-completeness of deciding progress in systems with a linear or star-like communication pattern. Since deciding progress in general interaction systems is PSPACE-complete, deciding progress in interaction systems with any restriction on the communication structure is in PSPACE.

Definition 2.4:
Let Sys be a deadlock-free interaction system (see Definition 1.4) with global behavior $T = (Q, \mathrm{Int}, \rightarrow_T, q^0)$. A **run of** Sys is an infinite sequence σ

$$q^0 \xrightarrow{\alpha_1}_T q^1 \xrightarrow{\alpha_2}_T q^2 \ldots,$$

with $q^l \in Q$ and $\alpha_l \in \mathrm{Int}$ for $l \geq 1$.

Definition 2.5:
Let Sys be a deadlock-free interaction system with components K. A component $k \in \mathrm{K}$ **may progress** in Sys if for every run σ the component k participates infinitely often in σ. This is, there are infinitely many interactions α in σ with $k(\alpha) \neq \emptyset$.

An instance of the progress problem in interaction systems is given by a tuple (Sys, k) where Sys is a deadlock-free interaction system with components K and $k \in \mathrm{K}$. The question is if k may progress in Sys.

Figure 2.11: Transition system T_{pro} for the component *pro*.

In order to proof that deciding progress in tree-like interaction systems is PSPACE-hard we extend our reduction from Section 2.3. Let H be a QBF and Sys_H the associated tree-like interaction system, i.e., a certain global state q^t is reachable in the global behavior T of Sys_H if and only if H is true. The component H' reaches its state labeled t only if H is true and f only if H is false, i.e., exactly one of these local states is determined to be reached in every run of Sys_H. Both states only enable the port $end_{H'}$ that self-loops on these states (see Figure 2.6c). The port $end_{H'}$ is the only port in the unary interaction *evaluated*, i.e., Sys_H is deadlock-free because eventually *evaluated* becomes enabled permanently. The state q^t is the global state where the local behavior of each component is in its state labeled t. Note that this state is determined to be reached if and only if H is true and that the only enabled interaction in this state is *evaluated*.

The idea is to construct a modified system Sys'_H by introducing an additional component called *pro* that may progress if and only if H is true. Let $A_{pro} = \{t_{pro}\}$ be the set of ports of *pro*. The behavior is given by the transition system T_{pro} in Figure 2.11.

In addition we modify the component H' as follows. The set of ports $A_{H'}$ of the component H' is now given by

$$A_{H'} = \{\mathfrak{e}^1_{H'}, sub^1_{H'}t, sub^1_{H'}f, s^1_{H'}t, end\_true_{H'}, end\_false_{H'}\},$$

i.e., $end_{H'}$ is removed and the ports $end\_true_{H'}$ and $end\_false_{H'}$ are added. The modified behavior of H' is given by the transition system $T_{H'}$ in Figure 2.12.

In addition, the interaction *evaluated* is removed from the set Int of inter-

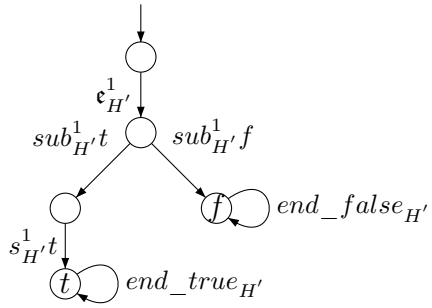


Figure 2.12: Modified transition system $T_{H'}$ for the component H'.

actions, and the interactions

$$\begin{aligned} evaluated\_true &= \{end\_true_{H'}, t_{pro}\} \text{ and} \\ evaluated\_false &= \{end\_false_{H'}\} \end{aligned}$$

are added. Let H be a QBF and Sys'_H the interaction system that is constructed as in Section 2.3 with the above modifications.

Theorem 2.4:
The progress problem in tree-like interaction systems, i.e., the question whether or not, given a tree-like interaction system Sys and a component k in Sys, k may progress in Sys, is PSPACE-complete.

Proof. The proof can be found in Appendix A on Page 196. □

In the following two remarks we argue why deciding progress in linear and star-like interaction systems is PSPACE-complete as well.

Remark 2.9:
We argue the PSPACE-completeness of deciding progress in linear interaction systems by describing how the reduction in 2.5 can be slightly modified to show that a certain component may progress if and only if the respective linear bounded Turing machine M accepts the respective input x. Let Sys_M be the interaction system that results from the reduction in Section 2.5 for a linear bounded Turing machine M and an input x for M. Note that our

reduction does not assure that Sys_M is deadlock-free. There are three cases on how M acts on the input x:

1. If M does not halt on x then it is easy to see that Sys_M is deadlock-free because the infinite execution of M is modeled by Sys_M.

2. If M actually decides that it does not accept x then M halts in the state p^N, i.e., Sys_M reaches a global state where a component $i \in \{0, 1, \ldots, n+1\}$ is in a state of the form (p^N, γ). A global state of this form is a deadlock. We argue that in each component each local state of the form (p^N, γ) can be extended by a self-loop that is labeled by a distinguished port. Furthermore, for each component we assume there is an interaction that consists only of this port, i.e., if this case is on hand, then our assumption guarantees deadlock-freedom in Sys_M.

3. If M accepts x then Sys_M reaches a global state where a component $i \in \{0, 1, \ldots, n+1\}$ is in a state of the form (p^Y, γ). In Remark 2.7 we argued that we can modify Sys_M such that the linear communication structure is preserved, while we can guarantee that a distinguished global state is reached. We assume here that this state is of the form $q^{end} = (q_0^{end}, q_1^{end}, \ldots, q_{n+1}^{end})$. For now we assume that a local state q_i^{end} ($i \in \{0, 1, \ldots, n+1\}$) does not enable any port, i.e., q^{end} is a deadlock.

Analogously as for tree-like interaction systems, we argue that Sys_M can be extended by an additional component *pro* (that is defined as in Figure 2.11) that may progress if and only if q^{end} is reachable in Sys_M. We extend the component that models cell 0 by a self-loop on the state q_0^{end} that is labeled by a port *end* and add the additional interaction $\{t_{pro}, end\}$. Thus, $\{t_{pro}, end\}$ becomes enabled permanently if and only if q^{end} is reached, i.e., the component *pro* may progress if and only if M accepts x.

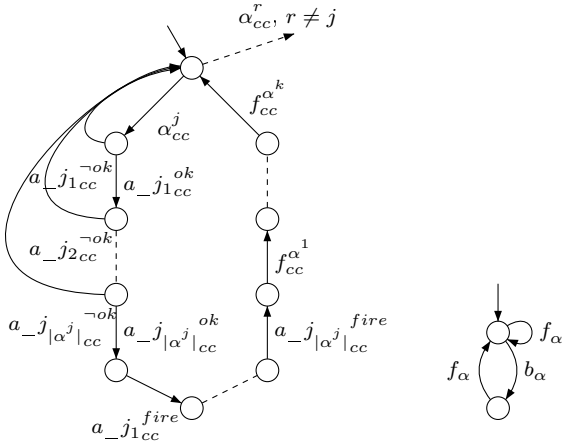
The extended version of the reduction still results in a linear interaction system because component *pro* only interacts with the component that models cell 0. Thus, deciding progress in linear interaction systems is PSPACE-

complete.

Remark 2.10:
in Section 2.6, our transformation to star-like systems does not preserve progress. Let α be an interaction that is not enabled in a global state q in the original system Sys. Let q' be the corresponding state in the associated star-like system Sys$'$. Starting in q' there is a sequence of transitions in Sys$'$ that corresponds to a test of whether or not all ports in α are enabled. This sequence can be repeated infinitely often. Let k be a component that may progress in Sys but does not participate in α. This is, starting in q' there is a run in Sys$'$ such that k does not participate in the interactions in this run, i.e., k may not progress in Sys$'$. The transformation can be extended such that progress is preserved. The idea is to exclude an interaction in Sys from being checked in Sys$'$ if it has been confirmed in Sys$'$ that this interaction is not enabled in the corresponding state in Sys. Clearly, this exclusion has to be revoked if a sequence of interactions was performed in Sys$'$ that corresponds to an interaction in Sys.

We extend Sys$'$ as follows. For every interaction α in Sys we introduce a component c_α with $A_{c_\alpha} = \{b_\alpha, f_\alpha\}$. The port b_α models that a check whether a sequence of interactions in Sys$'$ that corresponds to α shall be blocked and f_α revokes this block. The behavior T_{c_α} is depicted in Figure 2.13b. For each interaction α in Sys we extend the set of ports A_{cc} of component cc by a port f_{cc}^{α} that models that a block with respect to α shall be revoked. Let Int $= \{\alpha^1, \alpha^2, \ldots, \alpha^k\}$ be the set of interactions in Sys. Figure 2.13a depicts the extended behavior of component cc. Depicted is the part of the behavior that coordinates a check whether a sequence of interactions that corresponds to $\alpha^j = \{a_{j_1}, \ldots, a_{j_{|\alpha^j|}}\}$ for $j \in \{1, 2, \ldots, k\}$ is enabled in Sys$'$ and, if applicable, coordinates the execution of the respective ports. If T_{cc} successfully coordinated the interaction with each port in α^j, T_{cc} coordinates the unblocking of each component c_α for $\alpha \in$ Int.

The set of interactions in Sys$'$ is modified as follows. For $\alpha \in$ Int the port α_{cc}



(a) T_{cc} (b) T_{c_α}

Figure 2.13: Local behavior of component k_{α^j} for $1 \leq j \leq k$ and parts of the behavior of component cc.

of component cc shall interact with the port b_α of component c_α, i.e., α_{cc} can not interact again until c_α becomes unblocked. Thus, an interaction $\{\alpha_{cc}\}$ in Sys$'$ is replaced by an interaction $\{\alpha_{cc}, b_\alpha\}$. Furthermore, for $\alpha \in$ Int we introduce an interaction $\{f_{cc}^{\alpha}, f_\alpha\}$ that revokes a block.

It is easy to see that this extension of our reduction preserves progress, i.e., to decide progress in star-like interaction systems is PSPACE-complete.

2.8 Conclusion

We investigated complexity issues for component-based systems. In [CEP95] the reachability in 1-safe Petri nets was proven to be PSPACE-complete and in [MCM08c] this result was used to show the PSPACE-completeness of the reachability problem in general interaction systems. Here we restricted ourselves to systems with certain communication pattern, such as tree-like, star-like and linear communication structures, and showed that even in these classes deciding reachability is PSPACE-complete. Given these complexity issues it makes sense to look for conditions that can be tested in polynomial

time and guarantee a desired property that is related to reachability [Hoa85, BR91, BCD02, MCM08a, Lam09, HJK10].

Chapter 3

A Refinement Technique for Over-Approximations

3.1 Introduction

In this chapter we report about an approach to efficiently represent and refine over-approximations of the global behavior of cooperating systems [MCS13a]. An approach to circumvent complexity issues of checking various properties in cooperating systems, i.e., PSPACE-completeness results of the reachability problem and various other properties in interaction systems [MCM08c], is to investigate techniques based on sufficient conditions in order to establish those properties. In this context, an interesting subclass of system properties are safety properties which can be established in over-approximations of a cooperating system (this topic is discussed in detail in Chapter 4). Especially the system property of deadlock-freedom and the negated reachability problem (the question whether or not a certain state is not reachable) are safety properties and can be established in over-approximations of a cooperating system, i.e., if an over-approximation of a cooperating system is deadlock-free then the behavior of the system in consideration is deadlock-free as well. We introduce here a formal notion of

over-approximations of the global behavior of cooperating systems for the formalism of interaction systems that was introduced in Chapter 1. The global behavior of an interaction system is modeled by a labeled transition system, thus, our over-approximations are labeled transition systems as well. For complexity reasons these over-approximations are in general to large to be handled efficiently because an over-approximation of the global behavior of an interaction system suffers from the state space explosion problem just like the global behavior. Thus, we introduce a compact representation of an over-approximation that we call *abstract over-approximation.* An abstract over-approximation is a transition system that induces an over-approximation of the global behavior of an interaction system while it can be constructed in a way that it is of polynomial size in the size of the underlying interaction system.

If a safety property does not hold in an over-approximation of an interaction system then we can not conclude whether or not the underlying interaction system fulfills this property. In this case it might help to refine an over-approximation by which we here mean to remove sates and transitions such that the resulting object remains to be an over-approximation. This is, if an over-approximation of an interaction system is not deadlock-free but the refinement of this over-approximation results in an over-approximation where all transitions that lead to reachable deadlocks are removed then the underlying interaction system is deadlock-free. We introduce here a refinement technique that is based on an operator that we call Edge-Match. This operator compares pairs of abstract over-approximations and removes transitions such that the resulting transition system remains to be an abstract over-approximation.

A family of abstract over-approximations can be used to establish certain system properties in a cooperating system in polynomial time in the size of the underlying interaction system. This statement is discussed in Chapter 4 where we show how abstract over-approximations can be used to establish deadlock-freedom. In this chapter we treat exclusively the construction and

refinement of abstract over-approximations.

This chapter is organized as follows. In Section 3.2 we formally define over-approximations of the global behavior of interaction systems, our concept of abstract over-approximations and illustrate how an abstract over-approximation induces an over-approximation of the global behavior of an interaction system. Furthermore, we introduce the Edge-Match operator, a refinement operator that works on pairs of abstract over-approximations. In Section 3.3 we discuss preciseness aspects of the Edge-Match operator and argue that we can not expect to calculate "exact" abstract over-approximations by any refinement operator in polynomial time in the size of the underlying interaction system, i.e., abstract over-approximations where no further refinement is possible. In a second part of Section 3.3 we introduce an approach that refines a family of abstract over-approximations and argue under which assumptions this approach works in polynomial time. Section 3.4 concludes this chapter.

3.2 Abstract Over-Approximations and their Refinement

Here we introduce our concept of over-approximations of the reachable behavior of an interaction system Sys. As the global behavior of Sys is defined by a transition system we define an over-approximation of the global behavior of Sys as a transition systems that "includes" the reachable behavior of the global behavior of Sys, i.e., each reachable transition in the global behavior of Sys is included in an over-approximation. If T is the global behavior of Sys, then the size of the reachable behavior of T might be exponentially in the number of components in Sys, i.e., if Sys consists of a large number of components (e.g., 2.000 components where the local behavior of each component has 2 states) then it is not feasible to calculate the reachable behavior of T. Certainly, in this case it is not feasible to consider an

over-approximation of the reachable behavior as well. What we do is that we look at a transition system that "induces" an over-approximation of the reachable behavior.

We define an over-approximation of a transition system as follows.

Definition 3.1:
Let RT be the operator that yields all reachable transitions of a transition system. Let $R = (Q, A, \rightarrow_R, q^0)$ be a transition system with $\rightarrow_R \subseteq Q \times A \times Q$. A transition system $U = (Q', A', \rightarrow_U, q'^0)$ with $\rightarrow_U \subseteq Q' \times A' \times Q'$ is called an **over-approximation** (of the reachable behavior) of R if and only if $q^0 = q'^0$ and $RT(R) \subseteq RT(U)$. An over-approximation U of R is called **exact** over-approximation of R if and only if $RT(R) = RT(U)$.

We consider here a special type of over-approximations, i.e., over-approximations that are induced by an abstract over-approximation that is based on a subset of components. These abstract over-approximations are based on the following definition.

Definition 3.2:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\mathrm{IM} = (\mathrm{K}, \{A_i\}_{i \in \mathrm{K}}, \mathrm{Int})$ and $C \subseteq K$ with $C \neq \emptyset$. A transition system of the form $R = (Q_C, Int, \rightarrow_R, q_C^0)$ where

- $Q_C = \prod_{i \in C} Q_i$,
- $q_C^0 = (q_i^0)_{i \in C}$ and
- $\rightarrow_R \subseteq Q_C \times Int \times Q_C$

is called a **transition system with respect to** C. Let $T = (Q, Int, \rightarrow_T, q^0)$ be the global behavior of Sys. For $q \in Q$ the **projection** of q to the components in C is denoted by $q{\downarrow_C} \in Q_C$, i.e., if $q = (q_i)_{i \in K}$ then $q{\downarrow_C} = (q_i)_{i \in C}$.

In the following we define in which case a transition system with respect to C, i.e., a transition system of the form $R = (Q_C, Int, \rightarrow_R, q_C^0)$ is an abstract

over-approximation of the global behavior T of an interaction system Sys. In the immediately following lemma we justify the definition by showing that an induced transition system that is constructed by "extending" R is an over-approximation of T if and only if R is an abstract over-approximation of T.

Definition 3.3:
Let T be the global behavior of an interaction system Sys with components K and C a nonempty subset of K. Let $T' = (Q, Int, RT(T), q^0)$, i.e., the transition relation $\rightarrow_T$ of T restricted to reachable transitions. Let T'' be T' projected on the components in C, i.e., $T'' = (Q_C, Int, \rightarrow_{T''}, q_C^0)$ with $q_C \xrightarrow{\alpha}_{T''} q'_C$ if and only if there is a transition $q \xrightarrow{\alpha}_{T'} q'$ in T' with $q{\downarrow_C} = q_C$ and $q'{\downarrow_C} = q'_C$. We say a transition system $R = (Q_C, Int, \rightarrow_R, q_C^0)$ is an **abstract over-approximation of** T if and only if R is an over-approximation of T''.

In other words, R is an abstract over-approximation of T if and only if each reachable transition in the global behavior T projected on the components in C is reachable in R.

Lemma 3.1:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\mathrm{IM} = (K, \{A_i\}_{i \in K}, Int)$. Let $C \subseteq K$ be a nonempty subset of components and $R = (Q_C, Int, \rightarrow_R, q_C^0)$ a transition system with respect to C. Let $T = (Q, Int, \rightarrow_T, q^0)$ be the global behavior of Sys. The **global extension** of R is the transition system $E(R) = (Q, Int, \rightarrow_{E(R)}, q^0)$ such that for all $q, q' \in Q$ and all $\alpha \in Int$ the transition $q \xrightarrow{\alpha}_{E(R)} q'$ is in $E(R)$ if and only if a transition $q_C \xrightarrow{\alpha}_R q'_C$ is in R with $q{\downarrow_C} = q_C$ and $q'{\downarrow_C} = q'_C$. Then: R is an abstract over-approximation of T if and only if $E(R)$ is an over-approximation of T.

Proof. The proof can be found in Appendix A on Page 196. □

Remark 3.1:
If R is an abstract over-approximation of T then we call $E(R)$, defined as in

$$(q^1_{\mathrm{TER}_1}, q^2_{\mathrm{GS}}, q^1_{\mathrm{ADB}}, \underline{q^0_{\mathrm{TER}_2}, q^0_{\mathrm{DB}}}) \xrightarrow{send\_data}_{E(R)} (q^1_{\mathrm{TER}_1}, q^3_{\mathrm{GS}}, q^1_{\mathrm{ADB}}, \underline{q^1_{\mathrm{TER}_2}, q^1_{\mathrm{DB}}})$$

$$(q^1_{\mathrm{TER}_1}, q^2_{\mathrm{GS}}, q^1_{\mathrm{ADB}}, \underline{q^0_{\mathrm{TER}_2}, q^1_{\mathrm{DB}}}) \xrightarrow{send\_data}_{E(R)} (q^1_{\mathrm{TER}_1}, q^3_{\mathrm{GS}}, q^1_{\mathrm{ADB}}, \underline{q^1_{\mathrm{TER}_2}, q^0_{\mathrm{DB}}})$$

$$(q^1_{\mathrm{TER}_1}, q^2_{\mathrm{GS}}, q^1_{\mathrm{ADB}}, \underline{q^0_{\mathrm{TER}_2}, q^1_{\mathrm{DB}}}) \xrightarrow{send\_data}_{E(R)} (q^1_{\mathrm{TER}_1}, q^3_{\mathrm{GS}}, q^1_{\mathrm{ADB}}, \underline{q^1_{\mathrm{TER}_2}, q^1_{\mathrm{DB}}})$$

$$(q^1_{\mathrm{TER}_1}, q^2_{\mathrm{GS}}, q^1_{\mathrm{ADB}}, \underline{q^1_{\mathrm{TER}_2}, q^1_{\mathrm{DB}}}) \xrightarrow{send\_data}_{E(R)} (q^1_{\mathrm{TER}_1}, q^3_{\mathrm{GS}}, q^1_{\mathrm{ADB}}, \underline{q^1_{\mathrm{TER}_2}, q^0_{\mathrm{DB}}})$$

Figure 3.1: Transitions in a global extension $E(R)$ of R.

Lemma 3.1, the **induced over-approximation of T with respect to R.**

The following example shows some global extensions of a single transition.

Example 3.1:
Consider a subset of components $C = \{\mathrm{TER}_1, \mathrm{GS}, \mathrm{ADB}\}$ from the interaction system Sys presented in Examples 1.1 and 1.2 in Chapter 1. Let $R = (Q_C, Int, \rightarrow_R, q^0_C)$ be a transition system and

$$(q^1_{\mathrm{TER}_1}, q^2_{\mathrm{GS}}, q^1_{\mathrm{ADB}}) \xrightarrow{send\_data}_R (q^1_{\mathrm{TER}_1}, q^3_{\mathrm{GS}}, q^1_{\mathrm{ADB}})$$

a transition in R. The components TER_2 and DB are not in C. Figure 3.1 shows some (4 out of 16) transitions in $E(R)$ that are extensions of the above transition. For better readability, the local states of components not in C are underlined. Note that the considered interaction *send_data* includes the port *send_val* from component DB. The only transition labeled by *send_val* in DB is $q^1_{\mathrm{DB}} \xrightarrow{send\_val}_{\mathrm{DB}} q^0_{\mathrm{DB}}$, i.e., the only transitions in Figure 3.1 that occur in the global behavior of Sys are those with q^1_{DB} on the left hand side and q^0_{DB} on the right hand side.

$E(R)$ is an induced over-approximation that is never constructed in any of our methods as it can become exponentially large because of the state space explosion problem. The refinement of $E(R)$ is taking place on R. If $E(R)$ is an over-approximation of T then R can be seen as a compact representation of $E(R)$.

The next lemma shows that transition systems that are abstract over-approximations can be easily constructed from the specification of an interaction system. We use these transition systems as an initial point for our refinement technique.

Lemma 3.2:
Let Sys $= (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\text{IM} = (K, \{A_i\}_{i \in K}, Int)$ and $C \subseteq K$ a nonempty subset of components. Let $S_C = (Q_C, Int, \rightarrow_{S_C}, q_C^0)$ be the transition system with transition relation defined as follows: $q_C \xrightarrow{\alpha}_{S_C} q'_C$ if for all $i \in C$: if $\alpha \cap A_i = \{a_i\}$ then $q_i \xrightarrow{a_i}_i q'_i$ and if $\alpha \cap A_i = \emptyset$ then $q_i = q'_i$. Then S_C is an abstract over-approximation of the global behavior T of Sys.

Proof. The proof can be found in Appendix A on Page 198. □

Remark 3.2:
The definition of the transition relation of S_C is similar to the definition of the global behavior of an interaction system (see Definition 1.4 in Chapter 1). Actually, S_C equals the global behavior T if $C = K$.

Note further that for an interaction $\alpha \in \text{Int}$, where for all $i \in C$ we have $\alpha \cap A_i = \emptyset$, for each $q_C \in Q_C$ holds that $q_C \xrightarrow{\alpha}_{S_C} q_C$, i.e., interactions that do not include ports from a component in C label a self-loop in S_C. Actually, in S_C each state in Q_C has a self-loop for each interaction which does not include ports from a component in C.

In the following we discuss the refinement of abstract over-approximations. A refinement by the Cross-Checking operator, as discussed in [Min10], considers the refinement of over-approximations of the reachable global state space of interaction systems and amounts to the removal of states from the over-approximations. Here we consider the refinement of the reachable global behavior of interaction systems. This is, we do not remove states from abstract over-approximations but transitions. Of course, if we consider an abstract over-approximation of the reachable global behavior and remove

transitions then the reachable state space might become smaller. In our context we use the term "refinement" based on the following definition.

Definition 3.4:
Let $U = (Q', A', \rightarrow_U, q'^0)$ be an over-approximation (of the reachable behavior) of the transition system $R = (Q, A, \rightarrow_R, q^0)$. A transition system $\bar{U} = (\bar{Q}, \bar{A}, \rightarrow_{\bar{U}}, \bar{q}^0)$ is called a **refinement** of U (with respect to R) if U is an over-approximation of $\bar{U}$ and $\bar{U}$ is an over-approximation of R. This is, $RT(R) \subseteq RT(\bar{U}) \subseteq RT(U)$.

Remark 3.3:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\mathrm{IM} = (K, \{A_i\}_{i \in K}, Int)$ and $C \subseteq K$ a nonempty subset of components. Let $R = (Q_C, \mathrm{Int}, \rightarrow_R, q_C^0)$ be an abstract over-approximation of the global behavior T of Sys, i.e., R is an over-approximation of the transition system T'' (see Definition 3.3). Thus, according to Definition 3.4, we call an abstract over-approximation $R' = (Q_C, \mathrm{Int}, \rightarrow_{R'}, q_C^0)$ of T a refinement of R if R is an over-approximation of R'.

A transition system S_C with respect to a subset of components C that is constructed as in Lemma 3.2 can induce a relatively coarse over-approximation $E(S_C)$ of the global behavior T, i.e., there can be a great deal of reachable transitions in $E(S_C)$ that are not reachable in the global behavior T. In order to refine $E(S_C)$ we modify S_C by removing transitions from S_C which only induce transitions in $E(S_C)$ that are not reachable in T. These are exactly the reachable transitions in S_C that are not projections of reachable transitions in T. Transitions like these are called **artifacts**. Refining S_C, i.e., removing transitions that are artifacts can result in a smaller reachable state space in a refined version S'_C (and thus in $E(S'_C)$ as well). In the following, an abstract over-approximation with respect to a subset of components C that is constructed as in Lemma 3.2 is denoted by an uppercase S indexed by C, i.e., S_C.

The following example shows an abstract over-approximation of the global

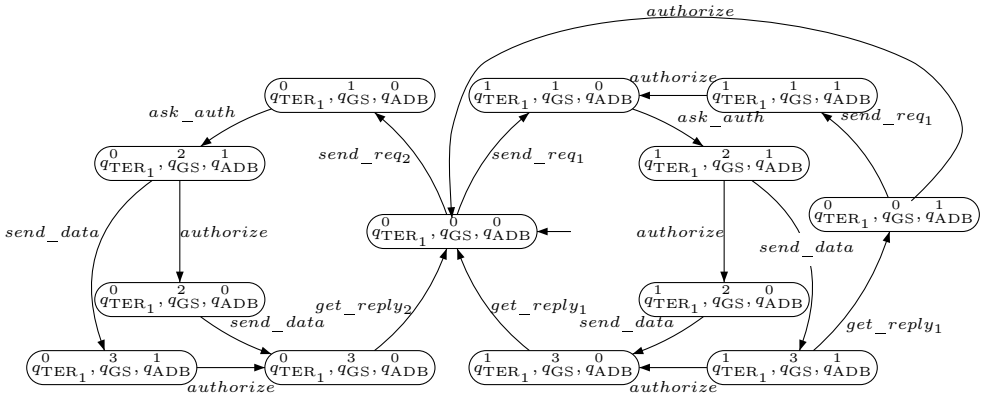


Figure 3.2: The transition system S in Example 3.2.

behavior of the interaction system that was introduced in Example 1.1 and 1.2.

Example 3.2:

Consider the interaction system Sys defined in Example 1.1 and 1.2 in Chapter 1 and the subset of components $C = \{\mathrm{TER}_1, \mathrm{GS}, \mathrm{ADB}\}$. Figure 3.2 shows a transition system $S = (Q_C, Int, \rightarrow_S, q_C^0)$. For the clarity of Figure 3.2, the transition system S consists of parts of the behavior of the abstract over-approximation S_C that is constructed as in Lemma 3.2. Omitted in S are

- all transitions that are not reachable in S_C,
- some, but not all, artifacts and
- all self-loops on the states that are labeled with interactions which do not need a component in C in order to participate.

Note that S is an abstract over-approximation if the self-loops are included.

We now give an example that demonstrates the idea of how a transition in an abstract over-approximation can be identified as an artifact by a comparison with another abstract over-approximation. Afterwards we formally define a refinement operator that is based on this idea.

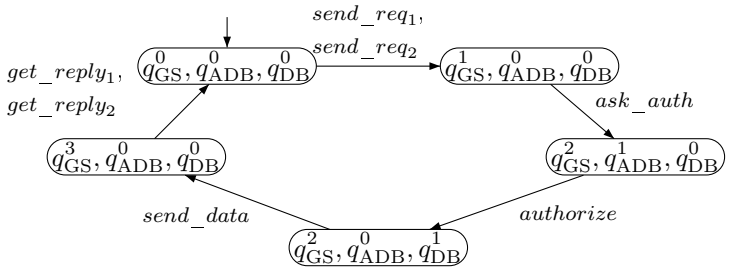


Figure 3.3: The transition system R in Example 3.3.

Example 3.3:
Consider the interaction system Sys defined in Examples 1.1 and 1.2 in Chapter 1 and the subset of components $D = \{\mathrm{GS}, \mathrm{ADB}, \mathrm{DB}\}$. Figure 3.3 shows a transition system $R = (Q_D, Int_D, \rightarrow_R, q_D^0)$. The transition system R consists of parts of the behavior of the abstract over-approximation S_D that is constructed as in Lemma 3.2. Omitted are all transitions that are not reachable and all self-loops on the states that are labeled by interactions which do not need a component in D in order to participate, i.e., R is an abstract over-approximation of the global behavior T of Sys if the self-loops are included.

In the following we assume that S (Example 3.2 respectively Figure 3.2) and R are abstract over-approximations of the global behavior T of Sys, i.e., there are self-loops on all states that are labeled by interactions which do not include ports from components in C and D respectively. We have to assume that each transition that is reachable in S is the projection of a transition that is reachable in the global behavior T of Sys. Consider the transition

$$(q^1_{\mathrm{TER}_1}, q^2_{\mathrm{GS}}, q^1_{\mathrm{ADB}}) \xrightarrow{send\_data}_S (q^1_{\mathrm{TER}_1}, q^3_{\mathrm{GS}}, q^1_{\mathrm{ADB}})$$

which is reachable in S and assume that this transition is the projection of a transition $q \xrightarrow{send\_data}_T q'$ that is reachable in T. As R is an abstract over-approximation as well, $q{\downarrow_D} \xrightarrow{send\_data}_R q'{\downarrow_D}$ has to be reachable in R. In q, the local behavior T_{GS} is in state q^2_{GS} and T_{ADB} is in state q^1_{ADB}. The only reachable state in R in which T_{GS} and T_{ADB} are in these local states

is $(q^2_{\text{GS}}, q^1_{\text{ADB}}, q^0_{\text{DB}})$. Thus, if $q{\downarrow_D} \xrightarrow{send\_data}_R q'{\downarrow_D}$ is reachable in R then $q{\downarrow_D}$ equals $(q^2_{\text{GS}}, q^1_{\text{ADB}}, q^0_{\text{DB}})$. This state does not have an outgoing transition labeled by *send_data* that leads to a state that agrees with $q'{\downarrow_C}$ on shared components. Thus, we can conclude that $q{\downarrow_C} \xrightarrow{send\_data}_S q'{\downarrow_C}$ is an artifact as this transition cannot be the projection of a transition that is reachable in T.

We now define the Edge-Match operator that removes transitions from a transition system S by a comparison with another transition system R. The subsequent theorem states that the result of the Edge-Match operator, if S and R are abstract over-approximations of the global behavior T of an interaction system Sys, is a refined version of S, i.e., only artifacts are removed from S.

Definition 3.5:
Let Sys be an interaction system with components K. Let C and D be nonempty subsets of K and $S = (Q_C, Int, \rightarrow_S, q^0_C)$ and $R = (Q_D, Int, \rightarrow_R, q^0_D)$ transition systems with respect to C respectively D. For a state $q_C \in Q_C$ the state $q_C{\downarrow_D} \in Q_{C \cap D}$ denotes the **projection** of q_C on the components in D, i.e., if $q_C = (q_i)_{i \in C}$ then $q_C{\downarrow_D} = (q_i)_{i \in C \cap D}$. Note that $q_C{\downarrow_D}$ yields the empty tuple if $C \cap D = \emptyset$. The **Edge-Match** operator EM on S and R yields a transition system $S' = EM(S, R)$ with $S' = (Q_C, Int, \rightarrow_{S'}, q^0_C)$ such that $q_C \xrightarrow{\alpha}_{S'} q'_C$ if and only if $q_C \xrightarrow{\alpha}_S q'_C$ is reachable in S and a transition $q_D \xrightarrow{\alpha}_R q'_D$ is reachable in R with $q_C{\downarrow_D} = q_D{\downarrow_C}$ and $q'_C{\downarrow_D} = q'_D{\downarrow_C}$.

Theorem 3.1:
Let Sys be an interaction system with components K and global behavior T. Let C and D be nonempty subsets of K and $S = (Q_C, Int, \rightarrow_S, q^0_C)$ and $R = (Q_D, Int, \rightarrow_R, q^0_D)$ abstract over-approximations of T, then $EM(S, R)$ is an abstract over-approximation of T.

Proof. The proof can be found in Appendix A on Page 198. □

Given an interaction system Sys with a set of components K we use the

Edge-Match operator to refine a family of abstract over-approximations of the global behavior T of Sys by a pairwise application. An abstract over-approximation is based on a subset C of components in K. Thus, a family of abstract over-approximations is indexed by a subset $\mathbb{C}$ of 2^K. We call this set a domain.

Definition 3.6:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\mathrm{IM} = (K, \{A_i\}_{i \in K}, Int)$. A set $\mathbb{C} \subseteq 2^K \setminus \{\emptyset\}$ of subsets of components is called a **domain** of IM.

The next example shows the result of the application of the Edge-Match operator on a family of abstract over-approximations from the running example.

Example 3.4:
Consider the interaction system Sys defined in Example 1.1 and 1.2 in Chapter 1. Let $\{S_C\}_{C \in \mathbb{C}}$ be the family of abstract over-approximations of T with respect to the domain

$$\begin{aligned}\mathbb{C} = \; & \{\{\mathrm{TER}_1, \mathrm{GS}, \mathrm{ADB}\}, \{\mathrm{GS}, \mathrm{ADB}, \mathrm{DB}\}, \{\mathrm{TER}_2, \mathrm{GS}, \mathrm{DB}\}, \\ & \{\mathrm{TER}_2, \mathrm{GS}, \mathrm{ADB}\}, \{\mathrm{TER}_2, \mathrm{GS}, \mathrm{TER}_1\}, \{\mathrm{GS}, \mathrm{TER}_1, \mathrm{DB}\}\}\end{aligned}$$

that are constructed as in Lemma 3.2. Let $\{R_C\}_{C \in \mathbb{C}}$ be the result of a sequence of applications of the Edge-Match operator on $\{S_C\}_{C \in \mathbb{C}}$. Figure 3.4 depicts parts of the abstract over-approximation R_C with $C = \{\mathrm{TER}_1, \mathrm{GS}, \mathrm{ADB}\}$. Omitted are all self-loops on the states that are labeled with interactions which do not need a component in C in order to participate. Note that R_C is a refined abstract over-approximation of the abstract over-approximation S described in Example 3.2. A total of 8 transitions that are in S are not included in R_C at which 4 reachable states in S become unreachable in R_C.

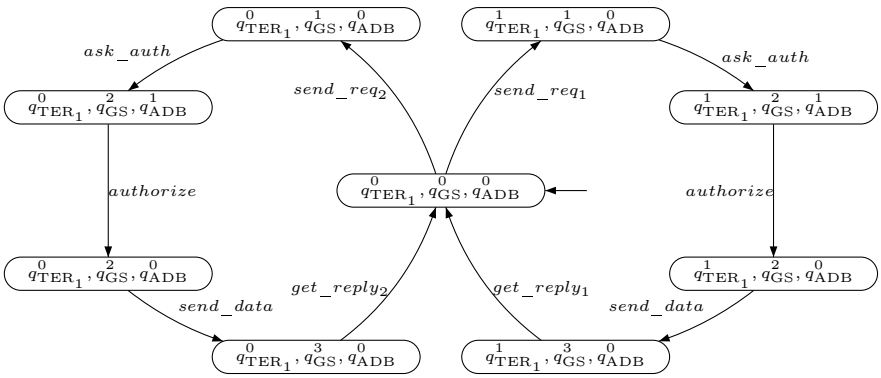


Figure 3.4: Part of the transition system R_C with $C = \{\mathrm{TER}_1, \mathrm{GS}, \mathrm{ADB}\}$ in Example 3.4.

3.3 Preciseness and Application

In this section we discuss the preciseness of abstract over-approximations that were refined by the Edge-Match operator, introduce how we apply our refinement technique on a family of abstract over-approximations and analyze the runtime of this approach.

An obvious question is, whether we can construct exact abstract over-approximations (see Definition 3.1). Unfortunately, we show here that we cannot expect to construct exact over-approximations by an algorithm that runs in polynomial time in the size of the underlying interaction system. We introduce here a weaker preciseness property of abstract over-approximations that we call legitimate. Roughly speaking, if we consider a family of abstract over-approximations, then a transition in an abstract over-approximation is legitimate if there is a transition in each abstract over-approximation such that all these transitions agree on shared components. Thus, a transition that is not legitimate is an artifact. A family of abstract over-approximations is legitimate if in all abstract over-approximations all transitions are legitimate. We argue that we cannot assume that a family of abstract over-approximations that was refined by the Edge-Match operator is legitimate. In Chapter 6 we show, by using results from the field of

relational databases, that for a certain subclass of families of abstract over-approximations holds that a refined by the Edge-Match operator results in a legitimate family of abstract over-approximations.

In a second part we formulate an algorithm that calculates a fixed-point of a family of abstract over-approximations with respect to an application of our refinement operator and give a detailed runtime analysis which describes under which circumstances our algorithm runs in polynomial time in the size of the underlying interaction system.

3.3.1 Preciseness

Here we discuss aspects regarding the preciseness of abstract over-approximations that were refined by the Edge-Match operator. By preciseness we mean whether or not there are artifacts in the abstract over-approximations. We consider here two questions:

1. Can we expect to construct exact abstract over-approximations in polynomial time in the size of the underlying interaction system, i.e., abstract over-approximations without artifacts?

2. If there are artifacts that cannot be detected by the Edge-Match operator, why does the operator fails here?

The following corollary shows that we cannot expect to generate exact abstract over-approximations by using a technique that runs in polynomial time in the size of the specification of an interaction system. This result is a direct conclusion from the fact that the decision problem whether there is a global state reachable where a fixed component is in a fixed local state is PSPACE-complete. This result was shown in Chapter 2 in Section 2.5.

Lemma 3.3:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\mathrm{IM} = (K, \{A_i\}_{i \in K}, Int)$. Let $C \in K$ be a nonempty subset of components.

Constructing an exact abstract over-approximation $S = (Q_C, Int, \rightarrow_S, q_C^0)$ can not be achieved in polynomial time in the size of the specification of Sys.

Proof. The proof can be found in Appendix A on Page 199. □

Lemma 3.3 states that we have to assume that an abstract over-approximation that was constructed and refined in polynomial time in the size of the specification of Sys contains artifacts. It is quite easy to characterize certain artifacts the detection of which is not covered by the Edge-Match operator. In the following we exemplify this claim and describe how these residual artifacts can be characterized.

Let Sys = (IM, $\{T_i\}_{i\in K}$) be an interaction system with interaction model IM = $(K, \{A_i\}_{i\in K}, Int)$, global behavior T and let R_D, R_{D_1} and R_{D_2} be abstract over-approximations of T. Let $q_D \xrightarrow{\alpha}_{R_D} q'_D$ be a reachable transition in the abstract over-approximation R_D and let $q_{D_1} \xrightarrow{\alpha}_{R_{D_1}} q'_{D_1}$ be the only transition that is reachable in R_{D_1} and $q_{D_2} \xrightarrow{\alpha}_{R_{D_2}} q'_{D_2}$ in R_{D_2} such that

$$q_D{\downarrow_{D_1}} = q_{D_1}{\downarrow_D}, q'_D{\downarrow_{D_1}} = q'_{D_1}{\downarrow_D}, q_D{\downarrow_{D_2}} = q_{D_2}{\downarrow_D} \text{ and } q'_D{\downarrow_{D_2}} = q'_{D_2}{\downarrow_D}.$$

Thus, $q_D \xrightarrow{\alpha}_{R_D} q'_D$ is in $EM(R_D, R_{D_1})$ and in $EM(R_D, R_{D_2})$. If now either $q_{D_1}{\downarrow_{D_2}} \neq q_{D_2}{\downarrow_{D_1}}$ or $q'_{D_1}{\downarrow_{D_2}} \neq q'_{D_2}{\downarrow_{D_1}}$ then it is easy to see that $q_D \xrightarrow{\alpha}_{R_D} q'_D$ is an artifact and that $q_D \xrightarrow{\alpha}_{R_D} q'_D$ is not removed by the application of the Edge-Match operator.

The following definition formalizes in which case a set of abstract over-approximations does not contain such artifacts. We define this property on families of abstract over-approximations.

Definition 3.7:
Let Sys = (IM, $\{T_i\}_{i\in K}$) be an interaction system with interaction model IM = $(K, \{A_i\}_{i\in K}, Int)$. Let $\mathbb{C}$ be a domain of IM and $\{R_C\}_{C\in\mathbb{C}}$ a family of abstract over-approximations. A transition $q_C \xrightarrow{\alpha}_{R_C} q'_C$ ($C \in \mathbb{C}$) is called **legitimate** if $q_C \xrightarrow{\alpha}_{R_C} q'_C$ is reachable in R_C and there exists a transition

$q \xrightarrow{\alpha}_T q'$ in the global behavior T such that $q{\downarrow_C} = q_C$ and $q'{\downarrow_C} = q'_C$ and for all $D \in \mathbb{C}$ $(D \neq C)$ holds that there exists a reachable transition $q_D \xrightarrow{\alpha}_{R_D} q'_D$ in R_D with $q{\downarrow_D} = q_D$ and $q'{\downarrow_D} = q'_D$.

R_C is called **legitimate** if all reachable transitions in R_C are legitimate. The family $\{R_C\}_{C \in \mathbb{C}}$ is called **legitimate** if all abstract over-approximations in $\{R_C\}_{C \in \mathbb{C}}$ are legitimate.

In other words, a reachable transition in an abstract over-approximation R_C is legitimate if we can find a reachable transition in each abstract over-approximation R_D for $D \in \mathbb{C}$ such that all these transitions agree on shared components. It follows that in this case there is a global transition that agrees with all these transitions on shared components as well, i.e., if a reachable transition in R_C is not legitimate then this transition is an artifact as there cannot be a reachable global transition which projection is reachable in R_C.

We argued that, in general, if $\{R_C\}_{C \in \mathbb{C}}$ resulted from a sequence of applications of the Edge-Match operator, we cannot assume that this family is legitimate. This motivates the consideration of additional approaches in order to identify and remove transitions in abstract over-approximations that are not legitimate or to analyze under which assumptions we can use our operator in order to generate a legitimate family of abstract over-approximations. In Chapter 6 we use a theorem from the field of relational database theory to show that a fixed-point with respect to the application of the Edge-Match operator is legitimate if $\mathbb{C}$ has a certain structure.

3.3.2 A Fixed-Point of a Family of Over-Approximations

A technique for establishing a system property that is based on a sufficient condition and uses abstract over-approximations is not able conclude that the property does not hold in the underlying interaction system. This is, if

the technique fails to establish the property then it is unknown whether or not the property holds. A refinement of the abstract over-approximations might help to establish the property, providing that the property actually holds in the interaction system. This is, we are interested in refining abstract over-approximations as much as possible by the Edge-Match operator. Given an interaction system Sys with a set of components K and a domain $\mathbb{C} \subseteq 2^K$ we want to calculate a fixed-point of the family of abstract over-approximations $\{S_C\}_{C\in\mathbb{C}}$ that are constructed as in Lemma 3.2, i.e., we want to apply a sequence of applications of the Edge-Match operator on $\{S_C\}_{C\in\mathbb{C}}$ such that no application of the Edge-Match operator on a pair of abstract over-approximations in the resulting family $\{R_C\}_{C\in\mathbb{C}}$ yields any refinement. After formally introducing in which case a family of abstract over-approximations is a fixed-point of the Edge-Match operator we proceed by showing that two fixed-points that result from different sequences of applications of the Edge-Match operator are identical, i.e., the quality of a fixed-point that results from the refinement by the Edge-Match operator is independent from the sequence of applications. We proceed by introducing an algorithm that calculates the fixed-point with respect to the Edge-Match operator of a family of abstract over-approximations and give a detailed runtime analysis. Our runtime analysis shows that a polynomial runtime of our algorithm completely depends on the choice of the domain of the family of abstract over-approximations. We propose a class of domains such that our algorithm runs in polynomial time on families that are based on these domains and we provide two interesting lemmas which show that we can, under certain conditions, modify a domain $\mathbb{C}$ while preserving the "information" in the fixed-point of the family $\{S_C\}_{C\in\mathbb{C}}$ that is constructed by the Edge-Match operator.

Definition 3.8:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i\in K})$ be an interaction system with interaction model IM $= (K, \{A_i\}_{i\in K}, Int)$ and $\mathbb{C}$ a domain of IM. Let $\{R_C\}_{C\in\mathbb{C}}$ be a family of transition systems of the form $R_C = (Q_C, \mathrm{Int}, \rightarrow_{R_C}, q_C^0)$. The family

$\{R_C\}_{C\in\mathbb{C}}$ is called a **fixed-point with respect to the Edge-Match operator** if $EM(R_C, R_D) = R_C$ for all $C, D \in \mathbb{C}$, i.e., no further application of the Edge-Match operator on a pair of transition systems yields any refinement.

First we show here that fixed-points with respect to the Edge-Match operator that result from different sequences of applications of the Edge-Match operator on pairs of transition systems in a family are identical. This result shows that we do not have to consider that there might be a sequence of refinement steps that leads to a more refined fixed-point.

Lemma 3.4:
Let Sys = (IM, $\{T_i\}_{i\in K}$) be an interaction system with interaction model IM = $(K, \{A_i\}_{i\in K}, Int)$ and $\mathbb{C}$ a domain of IM. Let $\{R_C\}_{C\in\mathbb{C}}$ be a family of transition systems of the form $R_C = (Q_C, Int, \rightarrow_{R'_C}, q^0_C)$. Let seq_1 and seq_2 be sequences that describe applications of the Edge-Match operator on $\{R_C\}_{C\in\mathbb{C}}$, i.e., sequences of tuples in $\mathbb{C} \times \mathbb{C}$. Let seq_1 result in the family $\{R'_C\}_{C\in\mathbb{C}}$ and seq_2 in the family $\{R''_C\}_{C\in\mathbb{C}}$. If $\{R'_C\}_{C\in\mathbb{C}}$ and $\{R''_C\}_{C\in\mathbb{C}}$ are fixed-points with respect to the Edge-Match operator then $\{R'_C\}_{C\in\mathbb{C}} = \{R''_C\}_{C\in\mathbb{C}}$.

Proof. The proof can be found in Appendix A on Page 199. □

Definition 3.9:
Let Sys = (IM, $\{T_i\}_{i\in K}$) be an interaction system with interaction model IM = $(K, \{A_i\}_{i\in K}, Int)$ and $\mathbb{C}$ a domain of IM. Let $\{R_C\}_{C\in\mathbb{C}}$ be a family of transition systems of the form $R_C = (Q_C, \mathrm{Int}, \rightarrow_{R_C}, q^0_C)$. If $\{R'_C\}_{C\in\mathbb{C}}$ is the fixed-point with respect to the Edge-Match operator that resulted from the application of a sequence of Edge-Match operations on $\{R_C\}_{C\in\mathbb{C}}$ then we call $\{R'_C\}_{C\in\mathbb{C}}$ the **Edge-Match fixed-point of** $\{R_C\}_{C\in\mathbb{C}}$.

Note that from Theorem 3.1 follows that the Edge-Match fixed-point of a family of abstract over-approximations is a family of abstract over-approximations.

An Algorithm for calculating the Edge-Match Fixed-Point

We now introduce an approach to calculate the Edge-Match fixed-point of the family of abstract over-approximations $\{S_C\}_{C\in\mathbb{C}}$ that is constructed as in Lemma 3.2. In our approach we successively apply the Edge-Match operator on all pairs of abstract over-approximations, that share a nonempty set of components, until no further application causes any refinement. Algorithm 2, called FP, describes this approach in pseudocode.

Algorithm 2 FP($\{S_C\}_{C\in\mathbb{C}}$)

1: $\{R_C\}_{C\in\mathbb{C}} \leftarrow \{S_C\}_{C\in\mathbb{C}}$
2: $\{R'_C\}_{C\in\mathbb{C}} \leftarrow$ **NIL**
3: **while** $\{R_C\}_{C\in\mathbb{C}} \neq \{R'_C\}_{C\in\mathbb{C}}$ **do**
4: $\quad\{R'_C\}_{C\in\mathbb{C}} \leftarrow \{R_C\}_{C\in\mathbb{C}}$
5: $\quad$**for** all $C, D \in \mathbb{C}$ with $C \neq D$ and $C \cap D \neq \emptyset$ **do**
6: $\quad\quad R_C \leftarrow EM(R_C, R_D)$
7: $\quad$**end for**
8: **end while**
9: **return** $\{R_C\}_{C\in\mathbb{C}}$

In the following we discuss the runtime of Algorithm 2 and argue under which assumption this runtime is polynomial in the size of the underlying interaction system. The runtime analysis shows that we can ensure that Algorithm 2 runs in polynomial time on $\{S_C\}_{C\in\mathbb{C}}$ if the domain $\mathbb{C}$ consists of a polynomial number of subsets and each subset consists of a number of components that is bounded by a constant. Later we propose a class of domains that fulfills these properties.

Runtime Analysis of Algorithm FP

Let Sys $= (\text{IM}, \{T_i\}_{i\in K})$ be an interaction system with interaction model IM $= (K, \{A_i\}_{i\in K}, Int)$ and $\mathbb{C}$ a domain of IM.

Let $c_{\max} = \max\{|C| \mid C \in \mathbb{C}\}$, i.e., the number of components in the largest set in $\mathbb{C}$ and let $m = \max\{|Q_i| \mid i \in K\}$, i.e., the size of the largest state space among all components in K. Let $C \in \mathbb{C}$ be an arbitrary set in the domain. The size of the state space Q_C of the abstract over-approximation S_C is bounded by $|Q_C| \leq m^{c_{\max}}$. Thus, the size of the transition relation $\rightarrow_{S_C} \subseteq Q_C \times Int \times Q_C$ is bounded by $| \rightarrow_{S_C} | \leq m^{2 \cdot c_{\max}} \cdot |Int|$.

The cost of a reachability analysis on S_C is bounded by $|Q_C| + |\rightarrow_{S_C}| \leq m^{c_{\max}} + m^{2 \cdot c_{\max}} \cdot |Int|$ (a modified BFS).

The application of the Edge-Match operator on a pair of abstract over-approximations S_C and S_D ($C, D \in \mathbb{C}$) consists of a comparison of each reachable transition in S_C with each reachable transition in S_D. Thus, the cost of an application corresponds to the costs of a reachability analysis on S_C and S_D plus $|\rightarrow_{S_C}| \cdot |\rightarrow_{S_D}|$. It follows that the costs of an application of the Edge-Match operator is bounded by

$$em = 2\left(m^{c_{\max}} + m^{2 \cdot c_{\max}} \cdot |Int|\right) + \left(m^{2 \cdot c_{\max}} \cdot |Int|\right)^2.$$

where the first summand describes the worst case costs of the reachability analysis on S_C and S_D and the second summand an upper bound for the number of pairs of transitions in S_C and S_D.

There are $|\mathbb{C}|$ abstract over-approximations in $\{S_C\}_{C \in \mathbb{C}}$. The for-loop in Line 5 to Line 7 performs an application of the Edge-Match operator on each pair of abstract over-approximations, i.e., the costs of this process are bounded by $|\mathbb{C}|^2 \cdot em$.

The number of all transitions in $\{S_C\}_{C \in \mathbb{C}}$ is bounded by $|\mathbb{C}| \cdot m^{2 \cdot c_{\max}} \cdot |Int|$. In the while-loop in Line 3 to Line 8 at least one transition is removed by the Edge-Match operator in each iteration. It follows that the runtime costs of Algorithm 2 are altogether

$$|\mathbb{C}| \cdot m^{2 \cdot c_{\max}} \cdot |Int| \cdot |\mathbb{C}|^2 \cdot em.$$

This runtime bound is polynomial in the size of the specifications of Sys if

1. $c_{\max}$ is a constant, i.e., the number of components in each set in $\mathbb{C}$ is bounded by a constant and
2. $|\mathbb{C}|$ is of polynomial size in the size of the specifications of Sys.

Thus, the requirement for Algorithm 2 to run in polynomial time in the size of the specifications of Sys is completely depending on the choice of the domain.

The size of a domain $\mathbb{C} \subseteq 2^K$ is bounded by $2^{|K|}$, i.e., the number of abstract over-approximations in a family can be exponentially in the number of components in an interaction system. Sure enough, a family consisting of an exponential number of abstract over-approximations requires an exponential number of applications of the Edge-Match operator in order to calculate the fixed-point. In the following we propose a domain such that Algorithm 2 runs in polynomial time in the size of the specifications of Sys on a family of abstract over-approximations that is based on this domain.

Let Sys be an interaction system with interaction model IM and a set of components K. We assume here that the interaction graph G of IM is connected. Consider the domain $\mathbb{C} \subseteq 2^K$ that consists of all subsets of the set of components K that are of a fixed constant size $d \ll |K|$, i.e., all subsets of the same constant size d which is considerable smaller than $|K|$. For a domain like this holds that $|\mathbb{C}| = \binom{|K|}{d} \leq |K|^d$ and each set in $\mathbb{C}$ is of constant size, i.e., Algorithm 2 runs in polynomial time in the size of the specifications of Sys on the family of abstract over-approximations $\{S_C\}_{C \in \mathbb{C}}$. In the following we argue that, depending on the structure of the interaction graph G, we can neglect certain sets in $\mathbb{C}$.

A Sophisticated Domain

Let Sys be an interaction system with a set of components K and $\mathbb{C} \subseteq 2^K$ a domain. Let $\{S_C\}_{C \in \mathbb{C}}$ be the family of abstract over-approximations that are constructed as in Lemma 3.2. Let $\{R_C\}_{C \in \mathbb{C}}$ be the Edge-Match fixed-point

of $\{S_C\}_{C\in\mathbb{C}}$. The following lemmas show that, under certain conditions, we can modify $\mathbb{C}$ by replacing subsets in $\mathbb{C}$ against others, that are not included in $\mathbb{C}$, or even removing entire subsets such that, roughly spoken, the information in $\{R_C\}_{C\in\mathbb{C}}$ is preserved in the Edge-Match fixed-point of the family that is based on the modified domain.

The first lemma exploits the structure of the interaction graph of an interaction model (see Definition 2.1 in Chapter 2) and shows that we can restrict ourselves domains consisting of subsets of components such that the interaction graph restricted to a subset is connected in a graph theoretic sense.

Definition 3.10:
Let $G = (V, E)$ be an undirected graph. A set of nodes $V' \subseteq V$ is called **connected** in G if any two nodes in V' are connected by a path and no node in V' is connected by an edge to a node in $V \setminus V'$. We say $G = (V, E)$ is connected if V is connected in G.

Let $\mathbb{C} \subseteq 2^K$ be a domain with respect to an interaction system Sys with a set of components K. Let $G = (K, E)$ be the interaction graph of the interaction model of Sys and $D \in \mathbb{C}$. We show that we can replace D by a partition of D that consists of the connected subsets of components in G restricted to the components in D.

Lemma 3.5:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i\in K})$ be an interaction system with interaction model IM $= (K, \{A_i\}_{i\in K}, Int)$ and $\mathbb{C}$ a domain of IM. Let $D \in \mathbb{C}$ where D is the disjoint union of the sets $D_1, D_2, \ldots, D_k$ such that D_i $(1 \leq i \leq k)$ is connected in the interaction graph G of IM restricted to the components in D.

Let $\mathbb{C}' = \mathbb{C}\setminus\{D\}$ and $\tilde{\mathbb{C}} = \mathbb{C}'\cup\{D_1, D_2, \ldots, D_k\}$. Let $\{R_C\}_{C\in\mathbb{C}}$ be the Edge-Match fixed-point of the family $\{S_C\}_{C\in\mathbb{C}}$ and $\{R'_C\}_{C\in\tilde{\mathbb{C}}}$ be the Edge-Match fixed-point of the family $\{S_C\}_{C\in\tilde{\mathbb{C}}}$.

Then the following two properties hold for the two families:

1. $R_C = R'_C$ for $C \in \mathbb{C}'$ and
2. a transition $q_D \xrightarrow{\alpha}_{R_D} q'_D$ is reachable in R_D if and only if the transition $q_{D_i} \xrightarrow{\alpha}_{R'_{D_i}} q'_{D_i}$ with $q_D{\downarrow_{D_i}} = q_{D_i}$ and $q'_D{\downarrow_{D_i}} = q'_{D_i}$ is reachable in R'_{D_i} for each $1 \leq i \leq k$.

Proof. The proof can be found in Appendix A on Page 200. □

The next lemma shows that we can remove a subset in a domain that is included in another subset in the domain. Roughly spoken, the reason for not including a subset in a domain is because the information that this subset of components contributes to the refinement process is covered by another subset of components already included in the domain.

Lemma 3.6:
Let $\text{Sys} = (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\text{IM} = (K, \{A_i\}_{i \in K}, Int)$ and $\mathbb{C}$ a domain of IM. Let $D_1, D_2 \in \mathbb{C}$ with $D_1 \subsetneq D_2$ and $\mathbb{C}' = \mathbb{C} \setminus \{D_1\}$. Let $\{R_C\}_{C \in \mathbb{C}}$ be the Edge-Match fixed-point of the family $\{S_C\}_{C \in \mathbb{C}}$ and $\{R'_C\}_{C \in \mathbb{C}'}$ be the Edge-Match fixed-point of the family $\{S_C\}_{C \in \mathbb{C}'}$. Then we can conclude that

1. $R_C = R'_C$ for $C \in \mathbb{C}'$ and
2. a transition $q_{D_1} \xrightarrow{\alpha}_{R_{D_1}} q'_{D_1}$ is reachable in R_{D_1} if and only if a transition $q_{D_2} \xrightarrow{\alpha}_{R'_{D_2}} q'_{D_2}$ with $q_{D_2}{\downarrow_{D_1}} = q_{D_1}$ and $q'_{D_2}{\downarrow_{D_1}} = q'_{D_1}$ is reachable in R'_{D_2}.

Proof. The proof can be found in Appendix A on Page 202. □

Let Sys be an interaction system with interaction model IM and a set of components K and consider the domain $\mathbb{C} \subseteq 2^K$ that consists of all subsets of the set of components K that are of a fixed constant size $d \ll |K|$.

Consider the domain $\mathbb{C}'$ that results from $\mathbb{C}$ by removing all $D \in \mathbb{C}$ where G restricted to D is not connected and adding the connected sets in G restricted to D, i.e., for each $C \in \mathbb{C}'$ holds that G restricted to C is connected. Let

$\{R_C\}_{C\in\mathbb{C}}$ respectively $\{R'_C\}_{C\in\mathbb{C}'}$ be the Edge-Match fixed-points which are calculated by Algorithm 2 from $\{S_C\}_{C\in\mathbb{C}}$ respectively $\{S_C\}_{C\in\mathbb{C}'}$. By Lemma 3.5 follows that for each $D \in \mathbb{C}$, if G restricted to D is connected, that $R_D = R'_D$ and, if G restricted to D is not connected, that we can construct R_D from the respective abstract over-approximations in $\{R'_C\}_{C\in\mathbb{C}'}$ that are based on the connected sets in G restricted to D.

Consider now the domain $\mathbb{C}''$ that results from $\mathbb{C}'$ by removing all $D_1 \in \mathbb{C}'$ if there is $D_2 \in \mathbb{C}'$ with $D_1 \subsetneq D_2$. Let $\{R''_C\}_{C\in\mathbb{C}''}$ be the Edge-Match fixed-point of $\{S_C\}_{C\in\mathbb{C}''}$ that was calculated by Algorithm 2. By Lemma 3.6 follows that for each $D \in \mathbb{C}' \cap \mathbb{C}''$ holds that $R'_D = R''_D$ and for each $D_1 \in \mathbb{C}' \setminus \mathbb{C}''$ that we can construct R'_{D_1} by projecting the transitions in an abstract over-approximation R''_{D_2} on D_1 where D_2 is a superset of D_1.

The domain $\mathbb{C}''$ consists of all subsets C of K with d components such that the interaction graph G restricted to C is connected. This domain is a subset of the domain that consists of all subsets of size d, i.e., our runtime analysis shows that Algorithm 2 runs in polynomial time in the size of the specifications of Sys on a family of abstract over-approximations that is based on this domain. Thus, this is the domain that we propose to use as a basis for Algorithm 2.

Example 3.5:
The interaction graph G of the interaction model in Example 1.1 is depicted in Figure 2.1 on Page 36. The Interaction model contains the set of components $K = \{\mathrm{TER}_1, \mathrm{TER}_2, \mathrm{GS}, \mathrm{ADB}, \mathrm{DB}\}$, i.e., K consists of 5 components. A domain $\mathbb{C}$ that consists of all subsets of size 3 includes $\binom{5}{3} = 10$ subsets.

The domain

$$\begin{aligned}\mathbb{C}' = \quad & \{\{\mathrm{TER}_1, \mathrm{GS}, \mathrm{ADB}\}, \{\mathrm{GS}, \mathrm{ADB}, \mathrm{DB}\}, \{\mathrm{TER}_2, \mathrm{GS}, \mathrm{DB}\}, \\ & \{\mathrm{TER}_2, \mathrm{GS}, \mathrm{ADB}\}, \{\mathrm{TER}_2, \mathrm{GS}, \mathrm{TER}_1\}, \{\mathrm{GS}, \mathrm{TER}_1, \mathrm{DB}\}\}\end{aligned}$$

consists of all subsets $C \subseteq K$ such that $|C| = 3$ and the interaction graph G restricted to C is connected. This domain only consists of 6 subsets. If we consider an interaction model with $s \geq 2$ terminals then a respective domain

$\mathbb{C}$ that consists of all subsets of size 3 includes $\binom{s+3}{3}$ subsets (s terminals and the 3 additional components GS, ADB and DB). If we consider a domain that consists of all subsets $C \subseteq K$ with $|C| = 3$ and the interaction graph G restricted to C is connected then it is easy to see that each set consists of the component GS and two arbitrary additional components. This is, there are $\binom{s+2}{2}$ sets in this domain. In other words, in this example, there are $\binom{s+2}{3}$ less sets in this domain compared to the domain that consists of all subsets of size 3.

Let Sys be an interaction systems with a set of components K. In summary, we propose to use a domain $\mathbb{C} \subseteq 2^K$ as a basis for a family of abstract over-approximations in Algorithm 2 that consists of all subsets C of K such that

1. C consists of d components for a constant $d \ll |K|$ and
2. the interaction graph G with respect to the interaction model in consideration restricted to the components in C is connected.

There is no set in $\mathbb{C}$ that can be neglected by Lemma 3.5 or Lemma 3.6 and each subset of K that consists of d components and is not in $\mathbb{C}$ is redundant for the calculation of the fixed-point of a family of abstract over-approximations that is based on $\mathbb{C}$.

A domain like that consists in the worst case of $\binom{|K|}{d} \leq |K|^d$ subsets and each abstract over-approximation is of polynomial size in the size of the specifications of Sys. By our runtime analysis of Algorithm 2 follows that the algorithm runs in polynomial time in the size of the specifications of Sys on a family of abstract over-approximations that is based on such a domain.

3.4 Conclusion and Related Work

In this chapter we presented a formal concept of over-approximations of the global behavior of interaction systems. Such over-approximations suffer

from the state space explosion problem just like the global behavior that is approximate. In order to circumvent this issue, we introduced abstract over-approximations that are based on a subset of components of an interaction system and discussed that these abstract over-approximations induce over-approximations of the global behavior of an interaction system while they, depending on the size of the underlying subset of components, are of polynomial size in the size of the specifications of the interaction system. Furthermore, we described the Edge-Match operator that compares pairs of abstract over-approximations in order to refine them and proposed an algorithm that computes the fixed-point of a family of abstract over-approximations with respect to an application of this operator. We showed under which assumptions our algorithm runs in polynomial time in the size of the underlying interaction system. The assumptions relate to the number of abstract over-approximations in a family and their individual size, i.e., the requirement for our algorithm to run in polynomial time is completely depending on the domain of the family in consideration. We proposed a subclass of families of abstract over-approximations that guarantees that we can calculate the fixed-point in polynomial time.

The work most related to our abstract over-approximations which induce over-approximations of a cooperating system modeled by the formalism of interaction systems is [Min10]. Similar to our approach, [Min10] considers not necessarily disjoint subsets of components of an interaction system and introduces subsystems that are based on these subsets and the glue-code of the system. The reachable state space of each subsystem is interpreted as a compact representation of an over-approximation of the reachable state space of the global behavior of the interaction system. In the following we call these sets *abstract state over-approximations* . Thus, the approach deals with over-approximations of the set of reachable global states and not with over-approximations of the reachable global behavior, i.e., reachable global transitions. The abstract state over-approximations are refined by a technique called Cross-Checking that, similar to the Edge-Match operator,

compares pairs of abstract state over-approximations and removes a subsystem state q_C if there is an abstract state over-approximation with no state that agrees with q_C on shared components. This is, our refinement approach uses a similar course of action while we use the information that is provided by the glue-code in our techniques. Thus, it is easy to see that if we interpret the sets of states in the Cross-Checking approach as abstract over-approximations in our setting then our approach results in more refined abstract over-approximations. In Chapter 5 we introduce several examples and point out the advantages of our approach over the Cross-Checking approach. In addition [Min10] introduces an approach to establish deadlock-freedom in interaction systems in polynomial time by an approach that is based on the analysis of abstract state over-approximations. The approach is briefly introduced in Chapter 4 in Section 4.3 as the waiting chain approach. We show by examples that our approach to establish deadlock-freedom and the waiting chain approach are incomparable, i.e., there are deadlock-free systems where our approach succeeds to establish deadlock-freedom and the waiting chain approach fails and vice versa.

[CHM+93] considers a partitioning of all subsystems in a cooperating system and performs reachability analyses on, so called, "subautomata" that are based on a set in this partition. Like in our approach, these subautomata are compact representations of over-approximations. The subautomata are based on pairwise disjunct subsets of subsystems, i.e., there is no way to apply a refinement approach based on the Cross-Checking or the Edge-Match operator. An approach that comes close to our approach is introduced in [GDHH98] where the reachable state space of synchronous hardware that is modeled by *Mealy machines* [Mea55] is approximated. The Mealy machine formalism comes very close to the formalism of interaction systems. The approach deals exclusively with the construction of compact representations of over-approximations. Similarly to our approach and the approach in [Min10], the authors consider not necessarily disjoint subsets of subsystems and interpret the reachable state space of a system based on one of these

subsets as a compact representation of an over-approximation of the reachable state space of the Mealy machine in consideration. Thus, the approach constructs abstract state over-approximations of a Mealy machine. In contrast to [Min10] the reachable state spaces are constructed simultaneously and the refinement of the abstract state over-approximations is achieved by cofactoring with the so far explored state spaces. Similarly to [Min10] and in contrast to our refinement approach, the approach in [GDHH98] considers the state transitions only for state space explorations and not for the refinement process of the abstract state over-approximations, i.e., the approach considers only over-approximations of the reachable state space and not over-approximations of the reachable transitions. In [GD99] the authors use this approach in order to establish invariants (a subclass of safety properties) by a combined forward reachability analysis from the initial state of a Mealy machine and backward reachability analysis from states that harm an invariant in consideration.

Introduced in [AC05] is a technique that is based on a sufficient condition in order to establish deadlock-freedom in *finite state concurrent programs* in polynomial time. The approach checks a condition that guarantees that there is an over-approximation of the global behavior of a system in consideration such that every time a subsystem changed its state it is ensured that this subsystem either does not block any cooperation or can participate in a cooperation. If the initial state is deadlock-free then this condition guarantees deadlock-freedom in a system. The condition is checked by an analysis of all abstract over-approximations that are based on subsets of 3 subsystems such that the interaction graph restricted to a subset is connected, i.e., in contrast to our approach the authors do not consider abstract over-approximations based on $d \neq 3$ subsystems. Similarly to the waiting chain approach in [Min10] the analysis of the abstract over-approximations is based on the refutation of certain waiting conditions among local states of the subsystems.

Chapter 4

Establishing Deadlock-Freedom

4.1 Introduction

Verifying properties of a system is a crucial part in the process of system design. Given system specifications, in system design a model in a formal language is constructed that should meet the specifications. Deciding various properties in interaction systems is PSPACE-complete [MCM08c], i.e., for certain system properties, we can not expect that there is a technique that decides this property in time polynomial in the size of an input interaction system. Particularly, deciding the system property of deadlock-freedom is PSPACE-complete, i.e., the property that states that there is no global state reachable in the global behavior of an interaction system that is a deadlock. Deadlock-freedom is an important and desirable system property in cooperating systems. Especially systems that are critically required to answer to unexpected or dangerous situations are crucially required to be deadlock-free, e.g., a control component in a power plant that is in a deadlock during an earthquake can block important safety precautions.

It is well known that the class of safety properties, i.e., properties that state that "something bad does never happen" [Lam77, LS85], can be established

in a system by checking these properties in an over-approximation of the system. This is, if "something bad does never happen" in an over-approximation of a system then it especially does never happen in the behavior of the system. In particular, the property of deadlock-freedom is a safety property. Thus, if an over-approximation of an interaction system is deadlock-free then the there is no deadlock reachable in the global behavior as well. In Chapter 3 we introduced a concept of over-approximations of the global behavior of an interaction system and abstract over-approximations that are based on a subset of components and induce over-approximations of the global behavior. In this chapter we describe how a family of abstract over-approximations can be used in order to establish deadlock-freedom in interaction systems in time polynomial in the size of an input interaction system. The general idea of establishing an arbitrary system property is based on [Min10]. Let Sys be an interaction system with a set of components K, $\mathbb{C} \subseteq 2^K$ a domain and $\{R_C\}_{C \in \mathbb{C}}$ a family of abstract over-approximations of the global behavior T of Sys. Let P be a system property and P' a predicate on abstract over-approximations such that from $P'(R_C)$ is true for all $C \in \mathbb{C}$ follows that P holds in Sys and the test whether $P'(R_C)$ is true for all $C \in \mathbb{C}$ can be achieved in polynomial time in the size of Sys. If P is a safety property, i.e., a property that states that "something bad will never happen" then we can add the following intermediate step. If we can conclude from $P'(R_C)$ is true for all $C \in \mathbb{C}$ that there must be an over-approximation T' of T such that P holds for T' then we can conclude that P holds in T as well.

This chapter is organized as follows. In Section 4.2 we give a brief compendium of definitions regarding linear time properties and safety properties. Section 4.3 introduces an approach for establishing deadlock-freedom in interaction systems by analyzing a family of abstract over-approximations. Furthermore, in Section 4.3 we compare our approach to an approach for establishing deadlock-freedom in interaction systems that was introduced in [Min10]. Section 4.4 concludes this chapter.

4.2 Safety Properties and Over-Approximations

In this section we provide a brief compendium of the definitions of linear time properties and the subclass of safety properties. We recapitulate the well known theorem that states that safety properties can be established in systems by testing these properties in over-approximations of the system and we exemplify how a family of abstract over-approximations can be used to guarantee that a safety property has to hold in an over-approximation of a system. The notations in this section are based on [BK08][1].

For a given transition system that describes the behavior of a system, we have to define a labeling function that assigns sets of so called "atomic" propositions to the states of the transition system, e.g., a system that models an ATM has states to which we could assign atomic propositions like "a user is logged in", "a wrong pin was entered" or "ERROR 37 occurred". These are propositions that are guaranteed in a fixed state of the system and are independent from states being visited previously or afterwards. The paths of a system induce sequences of sets of atomic propositions. A linear time property is defined by a set of sequences of sets of atomic propositions. A path in a system fulfills a linear time property if the induced sequence of propositions is included in the linear time property and a system fulfills a linear time property if the induced sequence of each path is included in the linear time property. In the following we formally introduce this concept.

Definition 4.1:
Let $S = (Q, A, \rightarrow_S, q^0)$ be a transition system with transition relation $\rightarrow_S \subseteq Q \times A \times Q$. A **finite path** in S is a finite sequence of states $\pi = q^0 q^1 q^2 \ldots, q^k$ such that for each $0 \leq i \leq k-1$ there is a transition $q^i \xrightarrow{a} q^{i+1}$ in S. Analogously, an **infinite path** in S is an infinite sequence of states $\pi' = q^0 q^1 q^2 \ldots$ such that for $i \geq 0$ there is a transition $q^i \xrightarrow{a} q^{i+1}$ in S. Note that a finite respectively infinite path starts in the initial state q^0. A finite path

[1] In contrast to [BK08] we allow a transition systems to contain reachable deadlocks, i.e., in some points the following definitions differ from the concepts in [BK08].

is called **maximal** if it ends in a deadlock.

Let $Path_{fin}(S)$ be the set of all finite paths in S and $Path(S)$ the set of all infinite and maximal finite paths in S.

If there are atomic propositions assigned to the states of a transition system then a finite path in the system induces a sequence of sets of atomic propositions. These sequences are called *traces*. A linear time property is defined by a set of sequences of sets of atomic propositions and a transition systems fulfills a linear time property if all traces of the system are included in the property.

Definition 4.2:
Let $S = (Q, A, \rightarrow_S, q^0)$ be a transition system, AP a set of atomic propositions and $L_S : Q \rightarrow 2^{AP}$ a labeling function that assigns a set of atomic propositions to each state in Q. Let $\pi = q^0 q^1 q^2 \ldots, q^k \in Paths_{fin}(S)$ be a finite path in S, then the **trace** of π is the sequence

$$trace(\pi) = L_S(q^0)L_S(q^1)L_S(q^2)\ldots L_S(q^k).$$

Let $Traces_{fin}(S) = \{trace(\pi)|\pi \in Paths_{fin}(S)\}$ be the set of all traces of finite paths in S. Analogously, for an infinite path $\pi' = q^0 q^1 q^2 \ldots$ let $trace(\pi') = L_S(q^0)L_S(q^1)L_S(q^2)\ldots$ and $Traces(S) = \{trace(\pi)|\pi \in Paths(S)\}$, i.e., the set of all traces of all infinite and maximal finite paths in S.

A **linear time property** (**LT-property**) P_{lt} is a subset of $(2^{AP})^\infty$ where the ∞ operator yields all finite and infinite concatenations of elements in 2^{AP}. A transition system S **satisfies** an LT-property P_{lt} if and only if $Traces(S) \subseteq P_{lt}$. We denote S satisfies P_{lt} as $S \models P_{lt}$.

A well investigated subclass of LT-properties is the subclass of *safety* properties. Roughly spoken, an LT-property is a safety property if we can confirm that the trace of a path in a transition system violates the property by only examine a finite prefix of the trace, i.e., even if the trace in consideration is of infinite length. A state in a system that is reached if we follow a path

that corresponds to a prefix like that is often interpreted as a "bad situation" because following this path violates the safety property. Thus, a safety property states that "something bad does never happen" [Lam77, LS85].

Definition 4.3:
Let AP be a set of atomic propositions and $P_{lt} \subseteq (2^{AP})^\infty$ an LT-property. The property P_{lt} is called a **safety property** if for each infinite word σ in $(2^{AP})^\infty \setminus P_{lt}$ (i.e., a word that does not belong to P_{lt}) there is a finite prefix σ' of σ such that all words σ'' in $(2^{AP})^\infty$ where σ' is a prefix of σ'' do not belong to P_{lt} as well.

Remark 4.1:
In other words, if σ is an infinite trace in a transition system S such that σ is not in a safety property P_{lt} then only a finite prefix of σ has to be examined in order to refute that $S \models P_{lt}$.

In the following example we illustrate the negated reachability problem in interaction systems in the presented notation of LT-properties and argue why this property is in fact a safety property. The negated reachability problem consists of the question, given an interaction system Sys and a global state q, whether q is not reachable in the global behavior T of Sys.

Example 4.1:
Let Sys $= (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with a set of components $K = \{1, 2, \ldots, n\}$ and interaction model $\text{IM} = (K, \{A_i\}_{i \in K}, Int)$. We assume that the local state spaces of the local behaviors are pairwise disjoint. Let $T = (Q, \text{Int}, \rightarrow_T, q^0)$ be the global behavior of Sys and $q = (q_1, q_2, \ldots, q_n) \in Q$ a global state in T. We want to establish whether q is not reachable in T. Let AP be the set of atomic propositions that consists of all states of the local behaviors in Sys, i.e., $AP = \cup_{i \in K} Q_i$. Let $L_T : Q \rightarrow 2^{AP}$ be a labeling function with $L_T(q') = \{q'_1, q'_2, \ldots, q'_n\}$ for $q' = (q'_1, q'_2, \ldots, q'_n) \in Q$, i.e., the atomic propositions that hold in a global state coincide with the respective local states. The LT-property

$$P = \left\{ \sigma | \sigma \in \left(2^{AP} \setminus \{q_1, q_2, \ldots, q_n\} \right)^\infty \right\}$$

consists of all finite and infinite sequences σ of subsets of AP where the set $\{q_1, q_2, \ldots, q_n\}$ does not occur in σ. Clearly,

- P is a safety property because a sequence that is not in P has a finite prefix that ends in $\{q_1, q_2, \ldots, q_n\}$ and
- $T \models P$ if and only if q is not reachable in T.

It is well known that a safety property can be established in a system by checking this property in an over-approximation of the system. This statement is based on the following theorem.

Theorem 4.1:
Let $S = (Q_S, A_S, \rightarrow_S, q_S^0)$ and $T = (Q_T, A_T, \rightarrow_T, q_T^0)$ be transition systems, AP a set of atomic propositions and $L_T : Q_T \rightarrow 2^{AP}$ and $L_S : Q_S \rightarrow 2^{AP}$ labeling functions. Then $Traces_{fin}(T) \subseteq Traces_{fin}(S)$ if and only if for each safety property P holds $S \models P \Rightarrow T \models P$.

Proof. A proof can be found in [BK08]. □

Let $T = (Q, A, \rightarrow_T, q^0)$ and $S = (Q, A, \rightarrow_S, q^0)$ be transition systems such that S is an over-approximation of T. Furthermore, let AP be a set of atomic propositions and $L : Q \rightarrow 2^{AP}$ a labeling function, then it is easy to see that $Traces_{fin}(T) \subseteq Traces_{fin}(S)$. This is because each finite path in T is also a finite path in S and thus each trace of a finite path in T is included in the set of finite traces of S as well. This means, if one can show that an arbitrary safety property P holds in S then P holds in T as well. Clearly, if P does not hold in S then we do not know whether P does or does not hold in T.

Note, if S is an over-approximation of T, that we can not conclude that T satisfies an arbitrary LT-property P which is not a safety property if $S \models P$, i.e., if $Traces(S) \subseteq P$. A maximal finite path in T the trace of which is not included in P might be a proper prefix of a maximal finite or infinite path in S. This is, we can not assume that $Traces(T) \subseteq Traces(S)$ and that

$Traces(T) \subseteq P$ if P is not a safety property.

We consider abstract over-approximations of an interaction system Sys which induce over-approximations of the global behavior T of Sys. Because of the state space explosion problem we want to avoid to analyze the over-approximation that is induced by an abstract over-approximation in order to ensure a safety property P in Sys. What we do is that we formulate a predicate P' on a family of abstract over-approximations such that if P' holds on the family then we can conclude that there is an over-approximation S of T with $S \models P$. By Theorem 4.1 it follows then that $T \models P$.

The following example illustrates, based on Example 4.1, how we can use abstract over-approximations in order to conclude that there is an over-approximation of the global behavior of an interaction systems where a certain global state is not reachable.

Example 4.2:
Let Sys $= (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\text{IM} = (K, \{A_i\}_{i \in K}, Int)$ and global behavior T. Let P be the safety property that was described in Example 4.1 that holds in T if and only if the global state $q = (q_1, q_2, \ldots, q_n) \in Q$ is not reachable in T.

Let R_C be an abstract over-approximation of T with respect to the subset of components $C \subseteq K$. If q is reachable in T then it follows from Definition 3.3 that $q{\downarrow_C}$ is reachable in R_C. Note, if $q{\downarrow_C}$ is reachable in R_C then we cannot conclude whether or not q is reachable in T. If $q{\downarrow_C}$ is not reachable in R_C then we can conclude that q is not reachable in the global extension $E(R_C)$ of R_C (see Lemma 3.1), i.e., for the over-approximation $E(R_C)$ of T holds that $E(R_C) \models P$. It follows by Theorem 4.1 that $T \models P$ as well.

Clearly, if $q{\downarrow_C}$ is reachable in R_C then a refinement by the Edge-Match operator, with other abstract over-approximations, might result in a refined version R'_C where $q{\downarrow_C}$ is not reachable, i.e., an abstract over-approximation that suffice to establish that q is not reachable in T.

4.3 An Approach to Establish Deadlock-Freedom

In this section we discuss how we can establish whether there is no deadlock (see Definition 1.4) reachable in an interaction system by considering a family of abstract over-approximations, i.e., we want to establish whether an interaction system is deadlock-free.

Our approach deals with the identification of states in abstract over-approximations that cannot be projections of reachable deadlocks in the global behavior T of an interaction system Sys. If there is one abstract over-approximation R of T where no state can be the projection of a reachable deadlock then we can conclude that there must be an over-approximation T' of T where no reachable state is a deadlock. This is because the respective projection of each reachable state in T is reachable in R. Deadlock-freedom is a safety property, i.e., we can continue to conclude that T is deadlock-free as well.

In the following we define the system property of deadlock-freedom in interaction systems as an LT-property. Let $\text{Sys} = (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\text{IM} = (K, \{A_i\}_{i \in K}, Int)$ and global behavior $T = (Q, \text{Int}, \rightarrow_T, q^0)$. Let $AP = \text{Int}$ be a set of atomic propositions and $L : Q \rightarrow 2^{AP}$ with

$$L(q) = \{\alpha \in \text{Int} \mid \forall_{i \in K} i(\alpha) \neq \emptyset \Rightarrow i(\alpha) \subseteq en(q_i)\},$$

i.e., $\alpha \in L(q)$ if and only if in the global behavior T the interaction α is enabled in q. The state q is a deadlock if and only if no interaction is enabled in q, i.e., if and only if $L(q) = \emptyset$. The interaction system Sys is deadlock-free if there is no deadlock reachable in T. Thus, deadlock-freedom of Sys can be specified as an LT-property by

$$P_{dlf} = \left\{ \sigma \mid \sigma \in \left(2^{AP} \setminus \{\emptyset\}\right)^{\infty} \right\},$$

i.e., all sequences of subsets of AP where $\emptyset$ does not occur in the sequence. Clearly, P_{dlf} is a safety property, because a sequence of subsets of AP violates

P_{def} if and only if this sequence has a finite prefix that ends in the empty set. Thus, if S is an over-approximation of T and $S \models P_{dlf}$ then it follows by Theorem 4.1 that $T \models P_{def}$, i.e., we can test whether P_{def} holds in T by checking whether P_{def} holds in an over-approximation S of T. Given an abstract over-approximation $R = (Q_C, \text{Int}, \rightarrow_R, q_C^0)$ of T, we can construct the induced over-approximation $E(R)$ and check a safety property in $E(R)$. Clearly, this course of action is not feasible if T, and thus $E(R)$ as well, is a complex transition system. We want to check P_{dlf} efficiently on an abstract over-approximation R of T. Each reachable state in R could be the projection of a deadlock that is reachable in T. Let $q_C \in Q_C$ be a reachable state in R and $E(q_C) \subseteq Q$ be all states $q \in Q$ with $q{\downarrow_C} = q_C$. We can conclude that q_C cannot be the projection of a reachable deadlock in T if there is no $q \in E(q_C)$ with $L(q) = \emptyset$. Clearly, Sys is deadlock-free if this property holds for each reachable state in an abstract over-approximation. This is a rather strict and naive approach to establish deadlock-freedom because a deadlock in $E(q_C)$ is not necessarily reachable in T. In the following we describe how this approach can be improved by comparing different abstract over-approximations in a family of abstract over-approximations.

We illustrate our approach on the example of the Dining Philosophers Problem and show additionally how we can use the information that our approach collects in order to remove possible deadlocks in this example in a non-automatic way.

Example 4.3:
In the remainder we demonstrate our approach to establish deadlock-freedom on variations of the well known *Dining Philosophers Problem* that was introduced by E. Dijkstra [Dij02]. The Dining Philosophers Problem is used to describe parallel processes which share a bounded number of resources. Basic version of this problem are not deadlock-free, i.e., the system might reach a state where no further activity is possible.

The problem is described as follows. We have $n \geq 2$ philosophers, numbered consecutively, sitting in an anticlockwise order around a table. Each philosopher has a plate of food in front of himself. Placed between two philosophers is a fork which has to be shared between these neighboring philosophers. A philosopher can think or eat. If a philosopher thinks he does not have any forks in his hands. If a philosopher wants to eat he needs both forks, left and right, i.e., if a philosopher eats then the two neighboring philosophers on his left and right cannot eat because the shared forks are in use. If a philosopher already took one fork, either its left or right, then he will not put it back on the table until he took the respective other fork and finished eating.

If the philosophers are allowed to choose nondeterministically which fork they take first (provided the respective fork lies on the table) then it is easy to see that a model that is based on these specifications is not deadlock-free. If all philosophers on the table are holding either their left respectively right fork then each philosopher waits to take his right respectively left fork.

In the following we model the Dining Philosophers Problem with n philosophers by an interaction system Sys_n. Afterwards we describe abstract over-approximations of the model and show which states indicate that the global behavior T of the Sys_n is not deadlock-free. Later we describe how we can use these states in the abstract over-approximations in order to modify the model such that it becomes deadlock-free.

Let $\mathrm{Sys}_n = (\mathrm{IM}, \{T_i\}_{i \in K})$ with $n \geq 2$ be an interaction system with interaction model $\mathrm{IM} = (K, \{A_i\}_{i \in K}, \mathrm{Int})$. Let

$$K = \{Phil_0, Phil_1, \ldots, Phil_{n-1}, Fork_0, Fork_1, \ldots, Fork_{n-1}\}$$

where $Phil_i$ models philosopher i and $Fork_i$ models fork i for $0 \leq i < n$. We assume that fork i is placed on the right of philosopher i. The set of ports A_{Phil_i} respectively A_{Fork_i} for philosopher i respectively fork i with $0 \leq i < n$ is specified as follows.

$$
\begin{aligned}
A_{Phil_i} = \{ \quad & take\_left_i, && // \text{ take the left fork} \\
& take\_right_i, && // \text{ take the right fork} \\
& put\_left_i, && // \text{ put the left fork on the table} \\
& put\_right_i && // \text{ put the right fork on the table}\}
\end{aligned}
$$

$$
\begin{aligned}
A_{Fork_i} = \{ \quad & take_i, && // \text{ take this fork} \\
& put_i && // \text{ put this fork back on the table}\}
\end{aligned}
$$

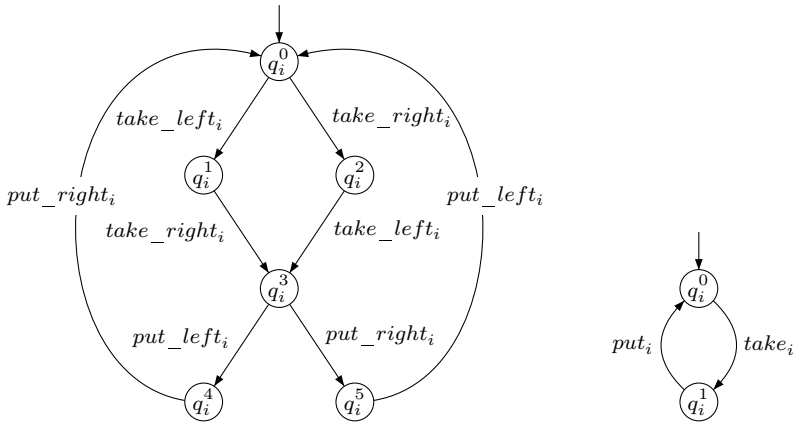
Each philosopher can take the fork on his left respectively right and put these forks back on the table. The following interactions model this cooperation between philosopher $Phil_i$ and the fork on his left $Fork_{i-1}$ and the fork on his right $Fork_i$ for $0 \leq i < n$. Note that we assume a modulo n arithmetic, i.e., if $i-1 = -1$ then $i-1$ refers to $n-1$ and if $i+1 = n$ then $i+1$ refers to 0.

$$
\begin{aligned}
tl_i &= \{take\_left_i, take_{i-1}\} \\
tr_i &= \{take\_right_i, take_i\} \\
pl_i &= \{put\_left_i, put_{i-1}\} \\
pr_i &= \{put\_right_i, put_i\}
\end{aligned}
$$

Let Int $= \{tl_i, tr_i, pl_i, pr_i | 0 \leq i < n\}$ be the interaction set of Sys_n. Figure 4.1 shows the local behavior of philosopher $Phil_i$ respectively $Fork_i$ for $0 \leq i < n$. The local behavior T_{Phil_i} is depicted in 4.1a and T_{Fork_i} in 4.1b. Note, for ease of presentation, we subscript the states in Figure 4.1 by i instead of $Phil_i$ respectively $Fork_i$.

By this, Sys_n is fully specified. Figure 4.2 shows the interaction graph for the interaction model of Sys_8, i.e., for a model of the Dining Philosophers Problem with 8 philosophers.

In the following we examine a family of abstract over-approximations of Sys_n that is constructed and refined as described in Chapter 3. Let $\mathbb{C}$ be a domain that consist of all subsets of K with 3 components where the interaction graph restricted to these three components is connected. There are two



(a) T_{Phil_i} (b) T_{Fork_i}

Figure 4.1: Local behavior of component $Phil_i$ and $Fork_i$ $(0 \leq i < n)$ in Example 4.3.

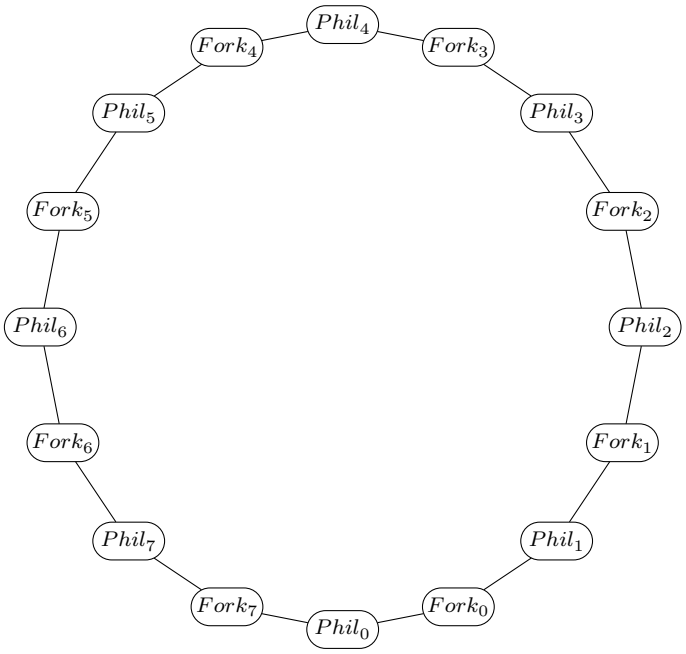


Figure 4.2: Interaction graph for the interaction model of Sys_8.

forms of subsets in $\mathbb{C}$, these are $C_i = \{Fork_i, Phil_{i+1}, Fork_{i+1}\}$ and $D_i = \{Phil_i, Fork_i, Phil_{i+1}\}$ for $0 \leq i < n$, i.e., two forks and one philosopher and one fork and two philosophers. Let $\{R_C\}_{C \in \mathbb{C}}$ be the family of abstract over-approximations that was constructed by Algorithm 2 (Chapter 3 on Page 89) from the family $\{S_C\}_{C \in \mathbb{C}}$ (see Lemma 3.2), i.e., $\{R_C\}_{C \in \mathbb{C}}$ is the Edge-Match fixed-point of $\{S_C\}_{C \in \mathbb{C}}$. Figure 4.3 shows R_{C_i} and Figure 4.4 R_{D_i} for $0 \leq i < n$. Not depicted are transitions that are labeled by interactions in which no component in C_i respectively D_i participates. The local states in the abstract over-approximation R_{C_i} respectively R_{D_i} are depicted in the order $Fork_i, Phil_{i+1}, Fork_{i+1}$ respectively $Phil_i, Fork_i, Phil_{i+1}$.

As mentioned, there are exactly two reachable deadlocks in the global behavior T of Sys_n. These are the states where all philosophers have taken their fork on the left respectively right. Certainly, the projections of these states are reachable in the abstract over-approximations. In Figure 4.3 these are the states $(q_i^1, q_{i+1}^1, q_{i+1}^1)$ (marked red) respectively $(q_i^1, q_{i+1}^2, q_{i+1}^1)$ (marked blue). In Figure 4.4 these are $(q_i^1, q_i^1, q_{i+1}^1)$ (marked red) respectively $(q_i^2, q_i^1, q_{i+1}^2)$ (marked blue). In the following we describe how we can automatically exclude that some of the states in the abstract over-approximations are projections of reachable deadlocks in the global behavior of Sys_n.

4.3.1 Projected Deadlocks

Given an interaction system Sys with a set of components K and a family of abstract over-approximations $\{R_C\}_{C \in \mathbb{C}}$ based on a domain $\mathbb{C} \subseteq 2^K \setminus \{\emptyset\}$ we describe here an approach to identify states in the abstract over-approximations that cannot be projections of reachable deadlocks in the global behavior T of Sys. If for one abstract over-approximation R_C ($C \in \mathbb{C}$) each reachable state $q_C \in Q_C$ was ruled out by this approach then we can conclude that Sys is deadlock-free. This conclusion is justified by the fact that each reachable state $q \in Q$ in T projected on C is reachable in R_C. Thus, if none

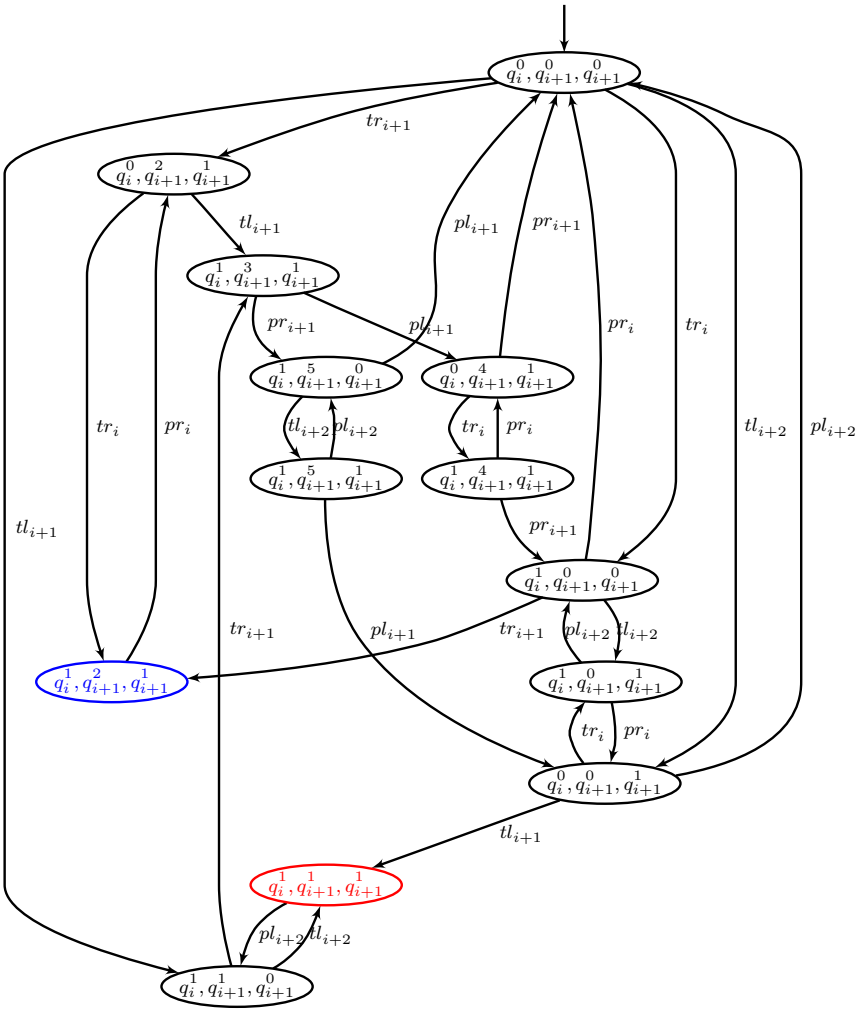


Figure 4.3: The abstract over-approximation R_{C_i} $(0 \leq i < n)$. Local states are depicted in the order $Fork_i, Phil_{i+1}, Fork_{i+1}$.

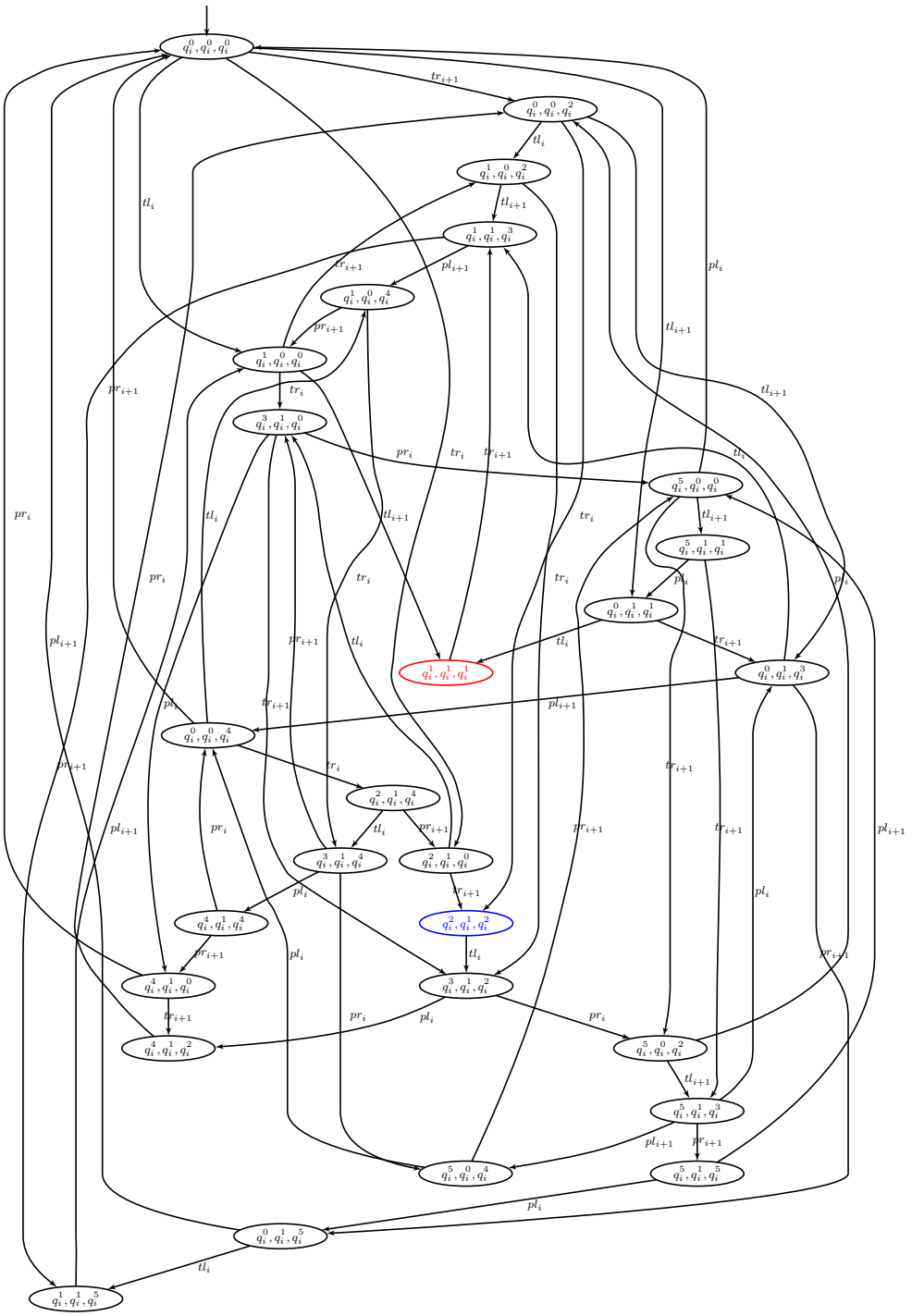


Figure 4.4: The abstract over-approximation R_{D_i} $(0 \leq i < n)$. Local states are depicted in the order $Phil_i, Fork_i, Phil_{i+1}$.

of the reachable states in R_C are projections of reachable deadlocks then there is an over-approximation S of T which is deadlock-free and thus, as of Theorem 4.1, T is deadlock-free as well.

The basic idea behind this approach is based on the following three observations. First we need one additional definition.

Definition 4.4:
Let Sys = (IM, $\{T_i\}_{i\in K}$) be an interaction system with interaction model IM = $(K, \{A_i\}_{i\in K}, Int)$, C a nonempty subset of K and $q_C \in Q_C$. Let

$$enC(q_C) = \{\alpha \in \text{Int} \mid \forall_{i\in K} i(\alpha) \neq \emptyset \Rightarrow (i \in C \wedge i(\alpha) \subseteq en(q_i))\}.$$

We say that an interaction $\alpha \in enC(q_C)$ is **complete** in q_C.

In other words, let $q_C \in Q_C$ be a state then $enC(q_C)$ consists of all interactions $\alpha \in Int$ in which only components in C participate and each component that participates in α enables its respective port. Note that each global state that projected on C equals q_C is assured to enable all $\alpha \in enC(q_C)$.

Observation 4.1:
If $q_C \in Q_C$ is a state that is not reachable in R_C, then q_C cannot be the projection of a reachable deadlock q in T because a state q with $q{\downarrow_C} = q_C$ is not reachable in T in the first place (see Definition 3.3).

Observation 4.2:
If $q_C \in Q_C$ is reachable in R_C and $enC(q_C) \neq \emptyset$ then q_C cannot be the projection of a reachable deadlock in T. Let $\alpha \in enC(q_C)$ then each port in α is enabled by the respective components in q_C. These ports are enabled in each global state q with $q{\downarrow_C} = q_C$ as well, i.e., $\alpha \in en(q)$.

Observation 4.3:
If $q_C \in Q_C$ is reachable in R_C and there is another abstract over-approximation R_D with $D \neq C$ such that for each reachable state $q_D \in Q_D$ in R_D with $q_C{\downarrow_D} = q_D{\downarrow_C}$ holds that either

- it is confirmed that q_D cannot be the projection of a reachable deadlock in T or

- for the state $q_{C\cup D} \in Q_{C\cup D}$ with $q_{C\cup D}{\downarrow_C} = q_C$ and $q_{C\cup D}{\downarrow_D} = q_D$ holds $enC(q_{C\cup D}) \neq \emptyset$

then q_C cannot be the projection of a reachable global deadlock as well. This is because, for each state q reachable in T with $q{\downarrow_C} = q_C$ holds that either $q{\downarrow_D}$ is not the projection of a deadlock or q is assured to enable an interaction in which only components in $C \cup D$ participate.

Based on the above observations we can formulate an approach that identifies states in a family of abstract over-approximations that can not be the projection of a reachable deadlock in the global behavior of an interaction system. Observation 4.1 and 4.2 affect only individual abstract over-approximations, i.e., there is no cross reference between pairs of abstract over-approximations. We use these two observations to construct an initial set of states for each abstract over-approximation that we want to refine by Observation 4.3. Observation 4.3 considers pairs of abstract over-approximations. Algorithm 3, named $CRIT$, describes an approach in pseudocode, that applies the above observations on a family of abstract over-approximations in order to construct a set of states for each abstract over-approximation in the family that consists of states for which we cannot exclude that they are projections of reachable deadlocks. The family of abstract over-approximations can be constructed as in Algorithm 2 (Chapter 3).

Observation 4.1 is applied in Line 1 in Algorithm 3 where $ReachStates(R_C)$ returns all reachable states in the abstract over-approximation R_C. The second observation is applied in Line 2 where $complete(H_C)$ returns all states q_C in H_C for which $enC(q_C) \neq \emptyset$. The third observation is used in the while-loop in Line 4 to 20 where pairs of sets of states are compared. The for-loop in Line 6 to 19 runs through all pairs $C, D \in \mathbb{C}$. For each state $q_C \in H_C$ we assume that this state is not the projection of a reachable deadlock – in Line 8 this is indicated by assigning *true* to the variable *notCritical*. The state q_C is then compared to each state $q_D \in H_D$, i.e., we only consider states q_C and q_D that can be projections of reachable deadlocks. In Line 12 the

algorithm checks whether there is an interaction that is complete in a state $q_{C\cup D}$ with $q_{C\cup D}{\downarrow_C} = q_C$ and $q_{C\cup D}{\downarrow_D} = q_D$. If there is one such state where no interaction is complete then we have to assume that q_C is the projection of a reachable deadlock – this is marked in Line 12 where *false* is assigned to the variable *notCritical*.

The while-loop terminates if there is no new state found that cannot be the projection of a reachable deadlock.

Algorithm 3 CRIT($\{R_C\}_{C\in\mathbb{C}}$)

```
1:  {H_C}_{C∈ℂ} ← {ReachStates(R_C)}_{C∈ℂ}
2:  {H_C}_{C∈ℂ} ← {H_C \ complete(H_C)}_{C∈ℂ}
3:  {H'_C}_{C∈ℂ} ← NIL
4:  while {H'_C}_{C∈ℂ} ≠ {H_C}_{C∈ℂ} do
5:    {H'_C}_{C∈ℂ} ← {H_C}_{C∈ℂ}
6:    for C, D ∈ ℂ do
7:      for q_C ∈ H_C do
8:        notCritical ←true
9:        for q_D ∈ H_D do
10:         if q_C↓_D = q_D↓_C then
11:           Let q_{C∪D} ∈ Q_{C∪D} with q_{C∪D}↓_C = q_C and q_{C∪D}↓_D = q_D
12:           notCritical ← notCritical ∧ enC(q_{C∪D}) = ∅
13:         end if
14:       end for
15:       if notCritical then
16:         H_C ← H_C \ {q_C}
17:       end if
18:     end for
19:   end for
20: end while
21: return {H_C}_{C∈ℂ}
```

In Chapter 3 we discussed under which assumptions the fixed-point of a

family of abstract over-approximations with respect to an application of the Edge-Match operator can be calculated in polynomial time. This is the case if the family consists of polynomial many abstract over-approximations and each abstract over-approximation is based on a number of components which is bounded by a constant d. This implies that the number of states in each abstract over-approximation is bounded polynomially in d. This means, if we assume that a family of abstract over-approximations is conform to this assumptions then Algorithm 3 runs in polynomial time. This is because we start with polynomially many states that we consider to be possible projections of deadlocks and in each execution of the while-loop in Line 4 to 20 at least one of these states is removed.

Remark 4.2:
In the following we say a state in an abstract over-approximation is marked as **critical** if we have not yet ruled out that this state is the projection of a reachable deadlock in the global behavior of the interaction system in consideration.

Example 4.4:
If we apply Algorithm 3 on the family of abstract over-approximations of our model of the Philosophers problem, described in Example 4.3, then we start with 13 reachable states in an abstract over-approximation R_{C_i} and 27 reachable states in an abstract over-approximation R_{D_i} for $0 \leq i < n$ (see Line 1), i.e., H_{C_i} contains 13 and H_{D_i} contains 27 critical states. After removing all states from H_{C_i} respectively H_{D_i} for $0 \leq i < n$ with $enC(q_{C_i}) \neq \emptyset$ respectively $enC(q_{D_i}) \neq \emptyset$ the refined sets contain 3 respectively 7 remaining critical states (see Line 2). After the while loop from Line 4 to Line 20 was executed, i.e., the refinement described in Observation 3 was applied, the updated set H_{C_i} contains 3 critical states and H_{D_i} 4 critical states $(0 \leq i < n)$.

The returned family of sets of states consists of n sets of size 3 and n sets of size 4. Remember that there are two reachable deadlocks in the global behavior of a model of the Dining Philosophers problem – the states where

each philosopher picked up the fork on his left respectively right. This is, we know that each abstract over-approximation contains exactly two states that are actually projections of reachable deadlocks, i.e., this example shows that we cannot expect to rule out all states that are not projections of reachable deadlocks by this approach.

In summary, we presented an approach that can be used to establish deadlock-freedom in interaction systems by analyzing a family of abstract over-approximations of the system. In order for the approach to run in polynomial time we propose to use a family of abstract over-approximations that are constructed as described in Section 3.3 in Chapter 3. We exemplified our approach on a model of the Dining Philosophers problem. The global behavior of this model is not deadlock-free. The example shows that our approach is able to exclude a great amount of states from being critical, i.e., the excluded states can not be projections of reachable global deadlocks. Nevertheless, there are states in the example, that our approach has marked as critical, which are not projections of reachable global deadlocks. However, we can use the obtained information, i.e., the family of sets of critical states, in order to modify the system such that our approach is successfully, i.e., such that there is at least one abstract over-approximation where we can conclude that no state could be the projection of a reachable deadlock. This approach can make sense even if all states marked as critical are actually false-positives, i.e., the system in consideration is deadlock-free. Our intention is, if the system in consideration is far to complex to be verified by exact techniques as LTL or CTL model checking then we modify the system such that the modifications preserve the initial design specifications sufficiently while our approach succeeds to establish deadlock-freedom.

The following example illustrates how we can modify our model of the Dining Philosophers in order to establish deadlock-freedom by our approach.

Example 4.5:
Consider the family of sets of critical states $\{H_C\}_{C \in \mathbb{C}}$ from Example 4.4, i.e.,

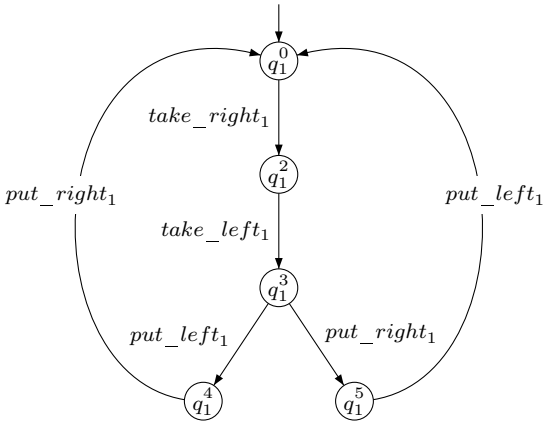


Figure 4.5: Modified behavior of philosopher $Phil_1$.

the output of Algorithm 3 applied on the family of abstract over-approximations described in Example 4.3. In our example $\{H_C\}_{C\in\mathbb{C}}$ consists of sets H_{C_i} and H_{D_i} for $0 \leq i < n$. Included in H_{C_0} ($C_0 = \{Fork_0, Phil_1, Fork_1\}$) is the state (q_0^1, q_1^1, q_1^1), i.e., the state where $Fork_0$ and $Fork_1$ are taken and $Phil_1$ has taken only the fork on his left (this state is marked red in Figure 4.3). This state is the projection of the reachable global deadlock where each philosopher took the fork on his left and waits to take the fork on his right. We want to modify the system such that this state becomes unreachable in R_{C_0}, i.e., such that this state does not appear in H_{C_0}. In R_{C_0} the state (q_0^1, q_1^1, q_1^1) can only be reached if philosopher $Phil_1$ took the fork $Fork_0$ on his left side and the fork $Fork_1$ on his right side was obtained by philosopher $Phil_2$ (who is not included in C_0). In order to prevent this situation we can modify the behavior of philosopher $Phil_1$ such that he takes the fork on his left only if he already holds the fork on his right, i.e., we modify the behavior of philosopher $Phil_1$ as depicted in Figure 4.5.

Roughly spoken, this modification prevents the deadlock in the global behavior where each philosopher took the fork on his left and waits for the fork on his right, i.e., the deadlock where each philosopher took the fork on his right and waits for the fork on his left should still be reachable in the

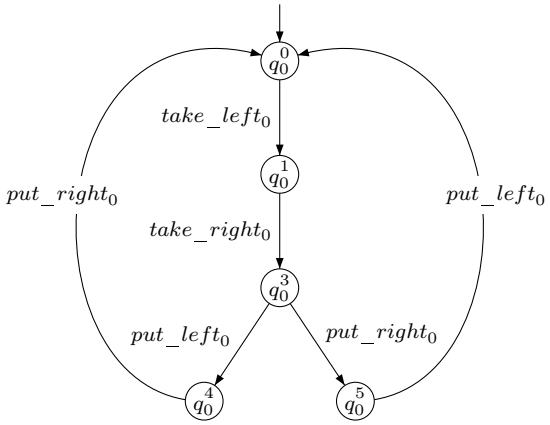


Figure 4.6: Modified behavior of philosopher $Phil_0$.

global behavior of this modified interaction system. Actually, if Algorithm 3 is applied on this modified system (where the family of abstract over-approximations is constructed just like in the original system) then each abstract over-approximation contains exactly one critical state. This state is the projection of the deadlock where each philosopher took the fork on his left and waits for the fork on his right. The reason for this is that the state (q_0^1, q_1^1, q_1^1) becomes unreachable in the abstract over-approximation R_{C_0} and other states that remained marked as critical in the original version are now confirmed to be not projections of deadlocks with respect to Observation 3.

In the same manner we can modify, e.g., the behavior of philosopher $Phil_0$ such that he only waits for the fork $Fork_0$ on his right if he already took the fork $Fork_{n-1}$ on his left, i.e., we modify the behavior of philosopher $Phil_0$ as depicted in Figure 4.6. This modification should prevent that the projection of the (former) reachable global deadlock where each philosopher took the fork on his right and waits for the fork on his left is reachable in the abstract over-approximation with respect to C_0 of the modified system.

It is easy to see that the modified system is deadlock-free. And, as expected, after our approach applied to this modified system there is no state marked as critical in any abstract over-approximation, i.e., the modified system is

confirmed to be deadlock-free. Note that projections of the reachable global deadlock where each philosopher holds the fork on his left respectively right in the original system are still reachable in respective abstract over-approximations of the modified system. Nevertheless, our approach confirmed that these projections can not be projections of reachable global deadlocks.

4.3.2 Comparison to the Waiting Chain Approach

In Chapter 3 in Section 3.4 we mentioned that our introduced refinement approach is similar to an approach that was introduced in [Min10] where objects are considered that are similar to abstract over-approximations. In addition an approach for testing whether an interaction system is deadlock-free is described in [Min10]. The approach exploits a waiting structure between local states in a deadlock and attempts to refute that there is a state reachable in the global behavior that exhibits a certain waiting structure by analyzing respective waiting structures in abstract over-approximations. In the following we give a brief and merely informal description of the approach in [Min10] and provide two examples that show that our approach and the approach in [Min10] are incomparable, i.e., there are systems where our approach succeeds and the approach in [Min10] fails and vice versa. Thus, we argue that one approaches can be applied if the other fails in order to establish deadlock-freedom in an interaction system. In the following we refer to the approach in [Min10] as the **waiting chain approach**.

The waiting chain approach is based on the following waiting structure on global states in an interaction system.

Definition 4.5:
Let $\text{Sys} = (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\text{IM} = (\text{K}, \{A_i\}_{i \in \text{K}}, \text{Int})$ and global behavior $T = (Q, \text{Int}, \rightarrow_T, q^0)$. Let $q \in Q$ be a global state. The **waiting graph of** q is a directed graph $G(q) = (V, E)$ with $V = \{q_1, q_2, \ldots, q_n\}$ for $q = (q_1, q_2, \ldots, q_n)$ and $(q_i, q_j) \in E$ if and only if there is $\alpha \in \text{Int}$ with

- $i(\alpha) \neq \emptyset$ and $j(\alpha) \neq \emptyset$ and
- $i(\alpha) \subseteq en(q_i)$ and $j(\alpha) \nsubseteq en(q_j)$.

This is, q_i enables a port that is included in an interaction α. Component j features a port that is included in α as well, but q_j does not enable this port. This is interpreted as q_i *waits* on q_j.

Let Sys $= (\mathrm{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\mathrm{IM} = (\mathrm{K}, \{A_i\}_{i \in \mathrm{K}}, \mathrm{Int})$ and global behavior $T = (Q, \mathrm{Int}, \rightarrow_T, q^0)$. We assume here that for each component $i \in K$ each local state $q_i \in Q_i$ enables at least one port, this is, $en(q_i) \neq \emptyset$. Let $q \in Q$ be a deadlock then it is easy to see that $G(q)$ contains a directed cycle. Each local state in q enables a port that is included in at least one interaction and, as q is a deadlock, no interaction is enabled in q because at least one local state does not enable its corresponding port, i.e., each local state waits on at least one other state. Note that it is possible that a global state $q \in Q$ is not a deadlock even if $G(q)$ contains a directed cycle.

The waiting chain approach, interpreted in our setting and our notations, works as follows. Let $\mathbb{C}$ be a domain that consists of all $C \subseteq K$ with $|C| = d$ for $d \ll |K|$ and $\{R_C\}_{C \in \mathbb{C}}$ a family of abstract over-approximations. Let $q \in Q$ be a global state that is a deadlock, i.e., $G(q)$ contains at least one directed cycle. Assume that q is reachable in the global behavior T of Sys and one directed cycle in $G(q)$ consists of local states of the components in $D \subseteq K$. The approach now distinguishes two cases $|D| \leq d$ and $|D| > d$, i.e., there are less or equal d local states involved in the cyclic waiting relation and there are more than d local states involved.

If $|D| \leq d$ then there must be $C \in \mathbb{C}$ with $D \subseteq C$ and there is $q_C \in Q_C$ reachable in R_C such that $q{\downarrow_D} = q_C{\downarrow_D}$. Thus, the directed cycle in $G(q)$ can be found in a respective representation of the waiting structure in q_C. It follows that, if in each abstract over-approximation R_C with $C \in \mathbb{C}$ there is no reachable state where local states are in a cyclic waiting relation then

there can not be a state reachable in the global behavior T where $\leq d$ local states are in a cyclic waiting relation.

If $|D| > d$ then for each chain of d local states on the cycle there must be an abstract over-approximation, based on the respective components, where the projection of q on these components is reachable. The chaining waiting relation is apparent in this projection. A first conclusion is, if for each $C \in \mathbb{C}$ there is no state $q_C \in Q_C$ reachable in R_C such that the local states in q_C are in a chaining waiting relation then there can not be a deadlock reachable in the global behavior T of Sys where more than d components are involved in a cyclic waiting relation. This observation can be strengthened as follows. Let $C_1, C_2 \in \mathbb{C}$ and $C_1, C_2 \subseteq D$ such that $q{\downarrow_{C_1}}$ and $q{\downarrow_{C_2}}$ are in a chaining waiting relation then it is clear that $q{\downarrow_{C_1}}$ and $q{\downarrow_{C_2}}$ agree on shared components. Thus, if a reachable state $q_C \in Q_C$ in the abstract over-approximation R_C is in a chaining waiting relation and there is no abstract over-approximation $R_{C'}$ for $C' \in \mathbb{C}$ where a state $q_{C'} \in Q_{C'}$ is reachable such that the local states in $q_{C'}$ are in a chaining waiting relation and q_C and $q_{C'}$ agree on shared components then the local states in q_C can not occur in a waiting chain in a cyclic waiting relation of a global state. This observation can be used in order to exclude a state in an abstract over-approximation from being involved in a cyclic waiting relation in a global state q.

The waiting chain approach attempts to exclude separately that there are reachable global states where less or equal d local states or more than d local states are in a cyclic waiting relation. If this succeeds then it is clear that there can not be a reachable global deadlock, i.e., the system in consideration is deadlock-free.

The waiting chain approach and our approach, that was introduced in Section 4.3, are incomparable. This is, there are interaction systems that are deadlock-free where the waiting chain approach fails to establish deadlock-freedom and our approach succeeds and vice versa. We show this claim by providing two simple examples. Example 4.6 introduces an interaction

system the global behavior of which is deadlock-free where our approach succeeds to establish deadlock-freedom and the waiting chain approach fails. On the other hand, in Example 4.7 we describe a deadlock-free interaction system where our approach fails to establish deadlock-freedom and the waiting chain approach succeeds. The consequence is that one approach can be applied on an interaction system if the other fails. Nevertheless, we recommend to apply our approach first. This is because the output of our approach consists of a family of states from abstract over-approximations that could be projections of reachable global deadlocks. States that are not in this family can not be projections of reachable deadlocks, i.e., there is no need to apply the waiting chain approach on these states.

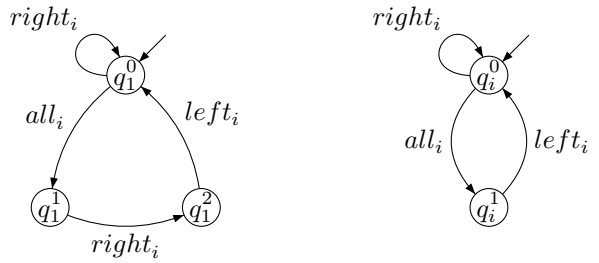
Example 4.6:
Let IM $= (\mathrm{K}, \{A_i\}_{i\in\mathrm{K}}, \mathrm{Int})$ be an interaction model with $\mathrm{K} = \{1,2,3,4\}$ and $A_i = \{left_i, right_i, all_i\}$ for $i \in \mathrm{K}$. We assume that the components are arranged circularly in an anticlockwise order. For $i \in \mathrm{K}$ the port $left_i$ models a communication with the component on the left, $right_i$ models a communication with the component on the right and the port all_i models a cooperation among all components in K. Thus, let Int consists of the following interactions:

- $comm_i = \{right_i, left_{i+1}\}$ for $i \in \mathrm{K}$ where $i + 1 = 5$ refers to 1 and
- $all = \{all_i | i \in \mathrm{K}\}$.

Let Sys $= (\mathrm{IM}, \{T_i\}_{i\in\mathrm{K}})$ be an interaction system. The local behavior of the components in Sys is depicted in Figure 4.7. 4.7a depicts the local behavior T_1 of component 1 and 4.7b depicts the local behavior T_i of component i for $i \in \{2,3,4\}$.

Figure 4.8 depicts the global behavior $T = (Q, \mathrm{Int}, \rightarrow_T, q^0)$ of Sys restricted to reachable transitions. Apparently, T is deadlock-free.

Let $\mathbb{C}$ be the domain that consists of all subsets of K of size 3, i.e., $\mathbb{C} =$



(a) T_1 (b) T_i with $i \in \{2,3,4\}$

Figure 4.7: Local behavior of the components in Example 4.6.

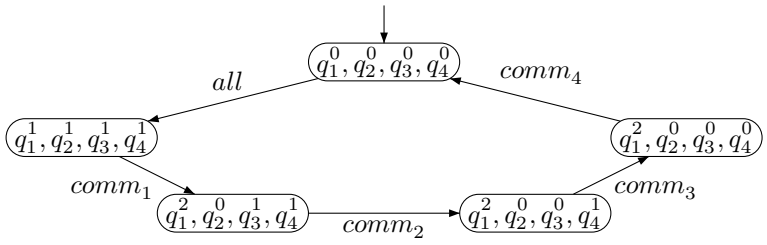


Figure 4.8: Global behavior T of Sys in Example 4.6.

$\{C_1, C_2, C_3, C_4\}$ with

$$C_1 = \{1,2,3\},$$
$$C_2 = \{1,2,4\},$$
$$C_3 = \{1,3,4\} \text{ and}$$
$$C_4 = \{2,3,4\}.$$

Let $\{S_C\}_{C \in \mathbb{C}}$ be the family of abstract over-approximations that is constructed as in Lemma 3.2 and $\{R_C\}_{C \in \mathbb{C}}$ be the family of abstract over-approximations that resulted from our refinement approach that is described in Chapter 3.

The waiting chain approach does not work on this example, i.e., the approach is not able to conclude whether or not the global behavior T of Sys is deadlock-free. The waiting graph $G(q^0)$ of q^0 contains a directed cycle because the state q_1^0 is waiting on q_2^0 with respect to the interaction $comm_1$, q_2^0 is waiting on q_3^0 with respect to the interaction $comm_2$ and so on. The

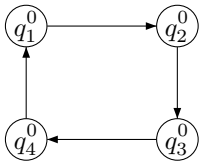
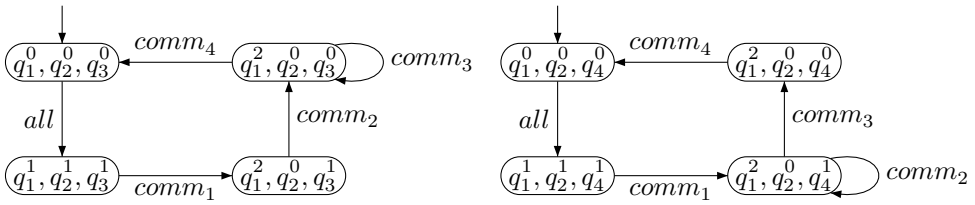


Figure 4.9: Waiting graph $G(q^0)$ in Example 4.6.



(a) R_{C_1} (b) R_{C_2}

Figure 4.10: The abstract over-approximations R_{C_1} and R_{C_2} in Example 4.6.

waiting graph $G(q^0)$ is depicted in Figure 4.9. The initial state q^0 is a reachable state in T, this is, all combinations of three states that are in a chaining waiting relation in $G(q^0)$ are reachable in the respective abstract over-approximation in $\{S_C\}_{C\in\mathbb{C}}$ (and in all abstract over-approximations that resulted from a refinement by the Edge-Match operator). Thus, the respective states in the abstract over-approximations contain a chaining waiting relation and the waiting chain approach can not exclude that any of these states appears in a cyclic waiting relation that involves more than three local states in a reachable global state.

Our approach on the other hand applied to this example concludes that the global behavior T of Sys is deadlock-free. Figure 4.10 depicts the behavior of the abstract over-approximations R_{C_1} and R_{C_2}.

Let us take a closer look at the abstract over-approximation R_{C_1}. The state (q_1^1, q_2^1, q_3^1) enables the interaction $comm_1$ and the state (q_1^2, q_2^0, q_3^1) the interaction $comm_2$. Both interactions only consists of ports from the components in C_1, i.e., $comm_1$ is complete in (q_1^1, q_2^1, q_3^1) and $comm_2$ is complete in (q_1^2, q_2^0, q_3^1). Thus, this two states can not be the projections of reachable

global deadlocks.

The only reachable states in R_{C_2} that agree with $(q_1^2, q_2^0, q_3^0) \in Q_{C_1}$ on shared components are (q_1^2, q_2^0, q_4^1) and (q_1^2, q_2^0, q_4^0). The state (q_1^2, q_2^0, q_4^0) can not be the projection of a reachable deadlock because the interaction $comm_4$ is complete in (q_1^2, q_2^0, q_4^0). There is no interaction complete in (q_1^2, q_2^0, q_4^1) but the interaction $comm_3$ is complete in $(q_1^2, q_2^0, q_3^0, q_4^1)$. Thus, (q_1^2, q_2^0, q_3^0) can not be the projection of a reachable global deadlock.

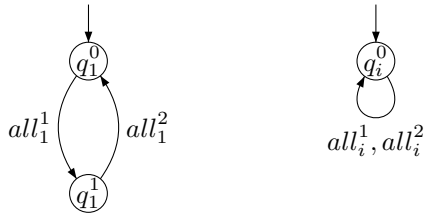
The only reachable state in R_{C_2} that agrees with $(q_1^0, q_2^0, q_3^0) \in Q_{C_1}$ on shared components is (q_1^0, q_2^0, q_4^0). There is no interaction complete in (q_1^0, q_2^0, q_4^0), however, the interaction all is complete in $(q_1^0, q_2^0, q_3^0, q_4^0)$, i.e., (q_1^0, q_2^0, q_4^0) can not be the projection of a reachable global deadlock.

This is, all reachable states in the abstract over-approximation R_{C_1} can not be projections of reachable global deadlocks, i.e., T is deadlock-free.

Example 4.7:
Let IM $= (\mathrm{K}, \{A_i\}_{i\in\mathrm{K}}, \mathrm{Int})$ be an interaction model with $\mathrm{K} = \{1, 2, \ldots, 7\}$ and $A_i = \{all_i^1, all_i^2\}$ for $i \in \mathrm{K}$. For component $i \in \mathrm{K}$, all_i^1 and all_i^2 are ports for a communication among all components in K. Thus, let Int consists of the two interactions $all^1 = \{all_i^1 | i \in \mathrm{K}\}$ and $all^2 = \{all_i^2 | i \in \mathrm{K}\}$. Note that the interaction graph G of IM is complete because both interactions in Int involve all components in K.

Let Sys $= (\mathrm{IM}, \{T_i\}_{i\in\mathrm{K}})$ be an interaction system. The local behavior of the components is depicted in Figure 4.11. 4.11a depicts the local behavior T_1 of component 1 and 4.11b depicts the local behavior T_i of component i with $i \in \{2, 3, \ldots, 7\}$.

It is easy to see that the global behavior $T = (Q, \mathrm{Int}, \rightarrow_T, q^0)$ of Sys has no reachable deadlock. In the global initial state q^0 only the interaction all^1 is enabled. If all^1 was performed then the only enabled interaction is all^2 which, if performed, leads back to the global initial state. Thus, T has exactly the two reachable states $(q_1^0, q_2^0, \ldots, q_7^0)$ and $(q_1^1, q_2^0, \ldots, q_7^0)$.



(a) T_1 (b) T_i with $i \in \{2, 3, \ldots, 7\}$

Figure 4.11: Local behavior of the components in Example 4.7.

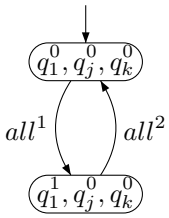


Figure 4.12: Behavior of the abstract over-approximation $S_{\{1,j,k\}}$ in Example 4.7.

Let $\mathbb{C}$ be the domain that consists of all subsets of K of size 3 and $\{S_C\}_{C \in \mathbb{C}}$ be the family of abstract over-approximations that is constructed as in Lemma 3.2. Let $i \neq j \neq k$ be components in K. If $1 \neq i, j, k$ then $S_{\{i,j,k\}}$ only consists of the initial state (q_i^0, q_j^0, q_k^0) with a self-loop that is labeled by all^1 and all^2. For $1 \neq j, k$ the abstract over-approximation $S_{\{1,j,k\}}$ is depicted in Figure 4.12.

For the two interactions in Sys holds that $|all^1| = |all^2| = 7$, i.e., there is no reachable state $q_C \in Q_C$ in any abstract over-approximation S_C with $C \in \mathbb{C}$ that enables an interaction that is complete in q_C. Furthermore, for $C_1, C_2 \in \mathbb{C}$ with $C_1 \neq C_2$ there is no pair of states $q_{C_1} \in Q_{C_1}$, $q_{C_2} \in Q_{C_2}$ such that the state $q_{C_1 \cup C_2} \in Q_{C_1 \cup C_2}$ with $q_{C_1 \cup C_2}{\downarrow_{C_1}} = q_{C_1}$ and $q_{C_1 \cup C_2}{\downarrow_{C_2}} = q_{C_2}$ enables an interaction that is complete in $q_{C_1 \cup C_2}$. This is because for all $C_1, C_2 \in \mathbb{C}$ holds that $|C_1 \cup C_2| \leq 6$ and both interactions in Int need all 7 components to participate. Thus, our approach fails on this system because

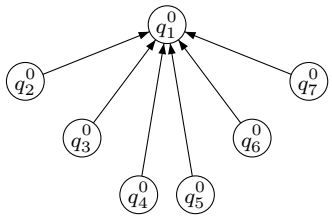


Figure 4.13: Waiting graph $G(q^0)$ of the global initial state q^0 in Example 4.7.

the system is deadlock-free and we can not exclude any state in the abstract over-approximations in $\{S_C\}_{C\in\mathbb{C}}$ from being the projection of a reachable global deadlock.

The waiting chain approach on the other hand, applied on the family of abstract over-approximations succeeds. Figure 4.13 depicts the waiting graph of the global initial state $q^0 = (q_1^0, q_2^0, \ldots, q_7^0)$ of T. The local states $q_2^0, \ldots, q_7^0$ are waiting on the state q_1^0 because they enable their respective port for interaction all^2 and q_1^0 does not enable the port all_1^2.

The waiting graph for the state $(q_1^1, q_2^0, \ldots, q_7^0)$ has the same structure as the graph in Figure 4.13 because the states $q_2^0, \ldots, q_7^0$ are waiting on the state q_1^1 with respect to the interaction all^1. All projections of these two graphs on a subset of three local states contain neither a cyclic waiting relation nor a chaining waiting relation that involves the respective local states. Thus, the waiting chain approach, applied on this example, establishes deadlock-freedom in the global behavior of Sys.

4.4 Conclusion

In this chapter we introduced an approach that can be used in order to establish deadlock-freedom in interaction systems by analyzing a family of abstract over-approximations. Our approach attempts to conclude that there

is an over-approximation of the global behavior of an interaction system that is deadlock-free. As deadlock-freedom is a safety property it follows that the global behavior of the system in consideration is deadlock-free as well. We argued that our approach runs in polynomial time under certain assumptions on the family of abstract over-approximations. Moreover, we described, using a version of the Dining Philosophers problem (which is not deadlock-free), how we can use the information that is calculated by our approach in order to modify a system such that we can guarantee deadlock-freedom. Additionally, we provided two examples which show that our approach and the waiting chain approach that is described in [Min10] are incomparable, i.e., if one approach fails to establish deadlock-freedom in an interaction system the other approach can be applied.

Chapter 5

Results

5.1 Introduction

We implemented our approach to refine abstract over-approximations and our approach to establish deadlock-freedom in a tool. In our tool we model the local behavior of components and abstract over-approximations by BDDs [Bry86]. BDDs offer the possibility to accomplish reachability analyses and applications of the Edge-Match operator efficiently by operations on BDDs. The symbolic representation of finite automata by BDDs is also the basis of the symbolic model checking approach [BCM$^+$92] where operations on BDDs are used to model check complex systems. Our tool takes as input an interaction system Sys in a description language and a parameter $d > 1$ where d is the number of components in an abstract over-approximation. An example for a system given in the description language and the methods that implement the Edge-Match operator and the reachability analysis can be found in Appendix B. The tool constructs all "reasonable" over-approximations with d components as described in Section 3.3.2. The calculation of all subsets C with d components where these components are connected in the interaction graph G of Sys restricted to C is achieved by a function that is based on an algorithm that is described in [MN06]. The following

additional software is used in the tool.

- JavaCC/JJTree, a Java compiler compiler is used to parse our description language of interaction systems[1].
- BuDDy, an efficient BDD library written in C/C++ and developed by Jørn Lind-Nielsen[2].
- Java Native Access (JNA), provides an interface to the native library BuDDy[3].
- Graphviz layout programs, for the visualization of transition systems and graph structures[4].
- `GraphViz.java`, a simple API to call dot from Java programs by Laszlo Szathmary[5].

In the following we introduce several parameterized examples of interaction systems and present results of our approach regarding the refinement of abstract over-approximations that is described in Chapter 3 and our approach to establish deadlock-freedom that is described in Chapter 4. The experiments were made on a computer with a dual-core 2.53GHz CPU and 4GiB RAM.

After each introduced example, we give a table that summarizes benchmarks regarding the application of our tool on various model instances. The instances differ on parameters that influence the number of components and the sizes of the local state spaces of the model. Additionally, we provide results from the application of the Cross-Checking operator [Min10] on the respective example in order to compare this refinement approach with our Edge-Match operator. The columns are labeled by parameters that where used in a specific model instance. The rows are labeled as follows.

[1] see `http://javacc.java.net/`

[2] see `http://sourceforge.net/projects/buddy/`

[3] see `https://github.com/twall/jna/`

[4] see `http://www.graphviz.org/Documentation.php`

[5] see `http://www.loria.fr/~szathmar/off/projects/java/GraphVizAPI/index.php`

- $|K|$ denotes the number of components in the model instance.
- $|\,\mathrm{Int}\,|$ denotes the number of interactions in the model instance.
- $|Q|$ is the size of the state space of the model instance. This includes reachable and unreachable states, i.e., the size of the Cartesian product of the local state spaces.
- $|\mathbb{C}|$ is the number of abstract over-approximations in the family on which we apply our refinement technique. The domain of our initial family of abstract over-approximations is constructed as described in Chapter 3 in Section 3.3.2. The size of the individual sets in the domain is indicated in the respective model description.
- $\Sigma|Q_C|$ denotes the sum over the sizes of all state spaces of the abstract over-approximations. This number includes initially reachable and unreachable states.
- $\Sigma Reach$ is the sum over the number of reachable states of all initial abstract over-approximations, i.e., abstract over-approximations that are constructed as in Chapter 3 Lemma 3.2. This is the number of states in all abstract over-approximations on which we have to apply our approach to establish deadlock-freedom if we do not use any refinement techniques.
- ΣCC denotes the sum over all states that remain in the fixed-point with respect to an application of the Cross-Checking operator on the family of abstract over-approximations.
- ΣEM denotes the sum over all states that remain in the fixed-point with respect to an application of the Edge-Match operator on the family of abstract over-approximations.
- % is the percentage of states from $\Sigma|Q_C|$ that is not reachable in the fixed-point with respect to the Edge-Match operator. This number indicates the strength of our refinement approach as attempts to es-

tablish deadlock-freedom (or another safety property) by analyzing abstract over-approximations do not have to deal with states that become unreachable by our refinement approach. This is, our approach to establish deadlock-freedom applied to the reachable states of the abstract over-approximations is less possible to produce a false-negative if there are less artifacts in the abstract over-approximations, i.e., if there are less states reachable.

- *time* is the time milliseconds that it takes to calculate the fixed-point with respect to the Edge-Match operator.
- *crit* is the sum over the number of states that remain critical in all abstract over-approximations in the fixed-point with respect to the Edge-Match operator after an application of our approach to establish deadlock-freedom. Note, if this numbers equals zero then our approach ensures deadlock-freedom for the respective model instance.

5.2 Measurement-Grid

This example describes a grid of data storages that allow to store measurement results from adjacent neighboring measurement station.

We consider an $m \times n$ $(m, n \geq 1)$ grid of data storages (DS). Two vertically respectively horizontally adjacent DSs share a measurement station V respectively H that is placed in between the two storages – the border of the grid is surrounded by vertical respectively horizontal measurement stations that are each used by only one data storage. A storage DS can decide to compare measurement values of their horizontally respectively vertically adjacent stations. If so, DS waits for both stations to connect. When the connections are established, then DS performs l $(l \geq 1)$ work-steps with each station. This is, our example is parametrized by l, m and n. After the work-steps are completed, both stations disconnect from DS.

Interaction Model

As an instance of the Measurement Grid example we consider a 2×2 grid G with $l = 1$ work-steps during a connection between a data storage and measurement stations. Let $\mathrm{IM}_G = (K, \{A_i\}_{i \in K}, Int)$ be an interaction model.

The set of components K is given by.

$$\begin{aligned} K \;=\; & \{DS_{i,j} | i = 0,1 \wedge j = 0,1\} \cup \; //\text{data storages} \\ & \{V_{i,j} | i = 0,1 \wedge j = 0,1,2\} \cup \; //\text{vertical measurement stations} \\ & \{H_{i,j} | i = 0,1,2 \wedge j = 0,1\} \; //\text{horizontal measurement stations} \end{aligned}$$

So far we did not describe, how a data storage DS obtains access to two adjacent vertical or horizontal measurement stations. This is regulated by a coordination between adjacent data storages. A data storage DS may access two vertically (horizontally) adjacent measurement stations if there is no conflict with the access of any of these stations by some other data storages.

For $i = 0,1$ and $j = 0,1$ the set of ports $A_{DS_{i,j}}$ for $DS_{i,j}$ consists of the following ports.

$vp_{i,j}$: obtain vertical priority
$hp_{i,j}$: obtain horizontal priority
$vc_{i,j}$: a vertical station connects
$hc_{i,j}$: a horizontal station connects
$if_{i,j}$: this storage is idling
$vw_{i,j}$: work-step with a vertical station
$hw_{i,j}$: work-step with a horizontal station
$vd_{i,j}$: a vertical stations disconnects
$hd_{i,j}$: a horizontal stations disconnects

For $i = 0,1$ and $j = 0,1,2$ the set of ports $A_{V_{i,j}}$ for $V_{i,j}$ consists of the

following ports.

$cv_{i,j}$: connect to a data storage
$wv_{i,j}$: work with a data storage
$dv_{i,j}$: disconnect from a data storage

The sets of ports $A_{H_{i,j}}$ for $H_{i,j}$ ($i = 0, 1, 2$ and $j = 0, 1$) are specified analogously.

The interactions of IM_G are specified for $i = 0, 1$ and $j = 0, 1$ as follows.

$$
\begin{aligned}
getPriorV_{i,j} &= \{vp_{i,j}, if_{i,j-1}, if_{i,j+1}\}^1 \\
getPriorH_{i,j} &= \{hp_{i,j}, if_{i-1,j}, if_{i+1,j}\}^1 \\
connLeft_{i,j} &= \{hc_{i,j}, ch_{i,j}\} \\
connRight_{i,j} &= \{hc_{i,j}, ch_{i+1,j}\} \\
connUp_{i,j} &= \{vc_{i,j}, cv_{i,j}\} \\
connDown_{i,j} &= \{vc_{i,j}, cv_{i,j+1}\} \\
workLeft_{i,j} &= \{hw_{i,j}, wh_{i,j}\} \\
workRight_{i,j} &= \{hw_{i,j}, wh_{i+1,j}\} \\
workUp_{i,j} &= \{vw_{i,j}, wv_{i,j}\} \\
workDown_{i,j} &= \{vw_{i,j}, wv_{i,j+1}\} \\
disConnLeft_{i,j} &= \{hd_{i,j}, dh_{i,j}\} \\
disConnRight_{i,j} &= \{hd_{i,j}, dh_{i+1,j}\} \\
disConnUp_{i,j} &= \{vd_{i,j}, dv_{i,j}\} \\
disConnDown_{i,j} &= \{vd_{i,j}, dv_{i,j+1}\}
\end{aligned}
$$

Note: [1]: $if_{i,j-1}$ is not included if $j-1 < 0$. Same goes for $if_{i,j+1}$ if $j+1 = n$, $if_{i-1,j}$ if $i-1 < 0$ and $if_{i+1,j}$ if $i+1 = n$.

Let Int be the set that consists of these interactions for $i = 0, 1$ and $j = 0, 1$.

Figure 5.1 shows the components of a 2×2 grid.

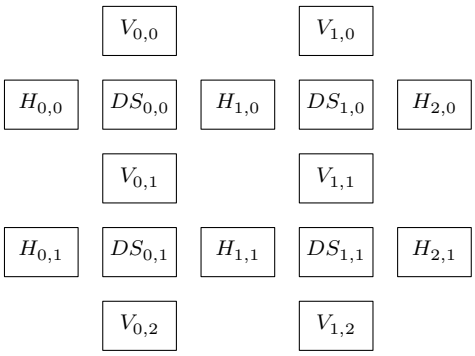


Figure 5.1: Components of a 2×2 grid.

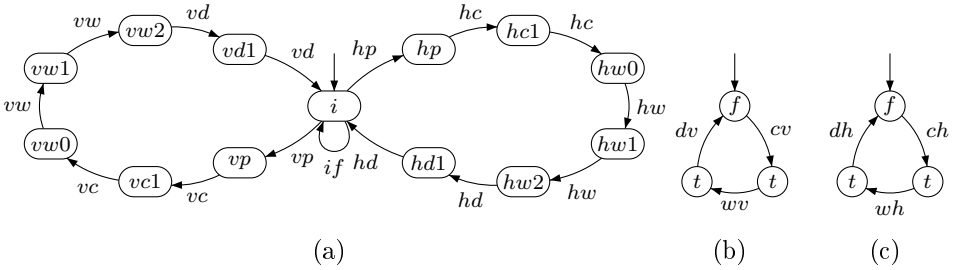


Figure 5.2: Behavior of the components of Sys_G.

Interaction System

Let $\mathrm{Sys}_G = (\mathrm{IM}_G, \{T_i\}_{i \in K})$ be the interaction system where the local behavior of the components is specified as follows. Figure 5.2 shows the behavior of the components of Sys_G. For better readability, we omitted the subscripts in the port names. Figure 5.2a shows the behavior $T_{DS_{i,j}}$ for $i = 0, 1$ and $j = 0, 1$, Figure 5.2b the behavior $T_{V_{i,j}}$ for $i = 0, 1$ and $j = 0, 1, 2$ and Figure 5.2c the behavior $T_{H_{i,j}}$ for $i = 0, 1, 2$ and $j = 0, 1$.

Results & Discussion

Table 5.1 shows benchmarks of the Measurement-Grid Example for $d = 3$, i.e., for each instance we considered a domain where all subsets of components are of size $d = 3$. The columns are labeled by (n, m, l), i.e., an

instance is an $n \times m$ grid with l working steps during a connection to a data storage. For each family of abstract over-approximations holds, after an application of our approach to establish deadlock-freedom, that there is no critical state in any abstract over-approximation, this is, our approach succeeds and guarantees that there is no reachable deadlock in the global behavior of our model instances. Our refinement approach has an advantage over the Cross-Checking approach for each instance. In fact, there is no state removed from the initial reachable states by the Cross-Checking operator, i.e., the Cross-Checking approach does not cause any refinement at all.

System	$(2,2,1)$	$(5,5,5)$	$(10,4,6)$	$(7,7,5)$	$(10,10,5)$
$\|K\|$	16	85	134	161	320
$\|\,\mathrm{Int}\,\|$	56	350	560	686	1,400
$\|Q\|$	$2^{33.82}$	$2^{289.89}$	$2^{483.78}$	$2^{552.46}$	$2^{1103.42}$
$\|\mathbb{C}\|$	56	946	1,610	2,150	4,856
$\Sigma\|Q_C\|$	25,792	10,005,754	25,837,020	24,443,462	57,657,104
$\Sigma Reach$	19,024	7,862,298	20,339,164	19,395,286	45,999,808
ΣCC	19,024	7,862,298	20,339,164	19,395,286	45,999,808
ΣEM	11,488	6,585,744	17,217,682	16,764,484	40,456,624
%	55.46	34.18	33.36	31.42	29.83
time	353	39,734	107,576	118,406	395,530
crit	0	0	0	0	0

Table 5.1: Benchmarks of the Measurement-Grid example for $d = 3$.

5.3 Tanenbaum's Philosophers

In [Tan07] Tanenbaum describes a solution that guarantees deadlock-freedom for the Dining Philosophers problem. The original problem and a model was introduced in Chapter 4 Example 4.3 in order to describe our approach to establish deadlock-freedom by analyzing abstract over-approximations. For

each philosopher Tanenbaum suggests to add a semaphore to a model of the system. A semaphore is binary and has the two states *locked* and *unlocked*. The idea is that a semaphore that is associated with a philosopher can only become locked if the semaphores associated with the neighboring philosophers are unlocked. A philosopher can only start his eating process, i.e., taking his forks, eating and putting the forks back on the table if his semaphore is locked. The semaphore becomes unlocked if the philosopher put both forks back on the table. This approach guarantees that a philosopher who already took one fork will never wait for the second fork, i.e., a philosopher who starts his eating process is assured to eat and put both forks back on the table.

Interaction Model

The specifications of this system differ from the system described in Example 4.3 in the following points.

- Before a philosopher can take a fork he has to gain the right to do so. In order to do this he interacts with his semaphore and the semaphores that are associated with his neighboring colleagues. If the neighboring semaphores are unlocked then his semaphore becomes locked and he is able to take either his left or right fork.
- after a philosopher finished eating, he can put both forks at once back on the table. In this process his semaphore becomes unlocked.

Let $\mathrm{IM}_n = (K, \{A_i\}_{i \in K}, \mathrm{Int})$ be an interaction model with $n \geq 2$. Let K be the set of components

$$\begin{aligned} K \;=\; & \{Phil_0, Phil_1, \ldots, Phil_{n-1}\} \cup \\ & \{Sem_0, Sem_1, \ldots, Sem_{n-1}\} \cup \\ & \{Fork_0, Fork_1, \ldots, Fork_{n-1}\} \end{aligned}$$

where $Phil_i$ models philosopher i, Sem_i Semaphore i and $Fork_i$ models fork i for $0 \leq i < n$. The set of ports A_{Phil_i} for philosopher i with $0 \leq i < n$

consists of the following ports.

get_prior_i : gain the right to take a fork
$take\_left_i$: take the left fork
$take\_right_i$: take the right fork
put_forks_i: put both forks back on the table

The set of ports A_{Sem_i} for semaphore i with $0 \leq i < n$ consists of the following ports.

$is\_unlocked_i$: the semaphore is unlocked
$lock_i$: lock this semaphore
$unlock_i$: unlock this semaphore

The set of ports A_{Fork_i} for fork i with $0 \leq i < n$ consists of the following ports.

$take_i$: take this fork
put_i : put this fork back on the table

Each philosopher can gain the right to pick up his forks on his left and his right by locking their respective semaphore, if the respective neighboring semaphores are unlocked. Furthermore, each philosopher can take his fork on the left respectively right and put these forks back on the table. The following interactions model these cooperations between philosopher $Phil_i$, semaphore Sem_i and the fork on his left $Fork_i$ and the fork on his right $Fork_{i-1}$ for $0 \leq i < n$. Note that we assume (similarly to Example 4.3) a modulo n arithmetic, i.e., if $i - 1 = -1$ then $i - 1$ refers to $n - 1$ and if $i + 1 = n$ then $i + 1$ refers to 0.

$$\begin{aligned} pr_i &= \{get\_prior_i, is\_unlocked_{i-1}, lock_i, is\_unlocked_{i+1}\} \\ tl_i &= \{take\_left_i, take_{i-1}\} \\ tr_i &= \{take\_right_i, take_i\} \\ pu_i &= \{put\_forks_i, put_{i-1}, put_i, unlock_i\} \end{aligned}$$

Let Int $= \{pr_i, tl_i, tr_i, pu_i | 0 \leq i < n\}$.

Figure 5.3 depicts the interaction graph G of the interaction model IM_n for $n = 8$.

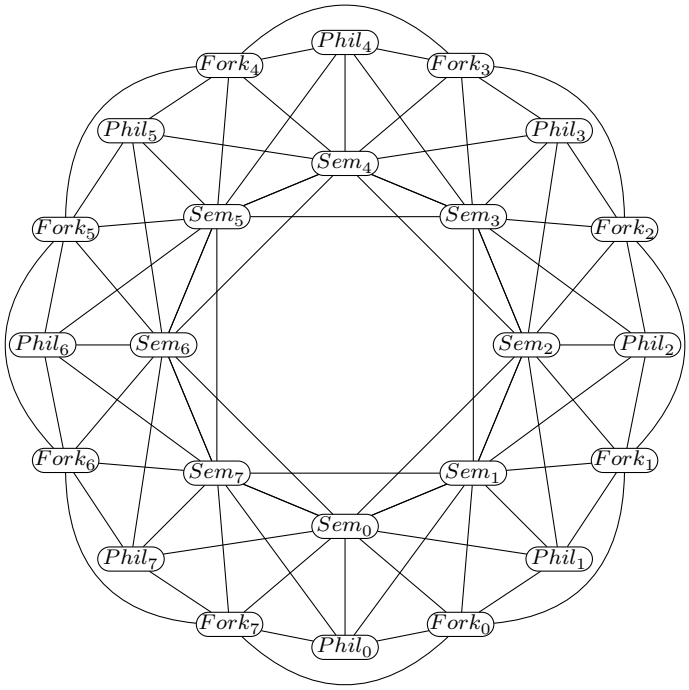


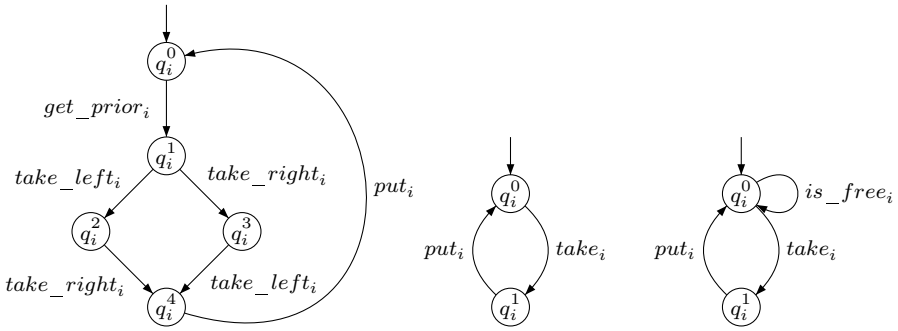
Figure 5.3: Interaction graph G of IM_8 in Tanenbaum's Philosophers example.

Interaction System

In the following we specify the interaction system $\mathrm{Sys}_n = (\mathrm{IM}_n, \{T_i\}_{i \in K})$ for $n \geq 2$ that provides the local behavior for each component described in the interaction model IM_n. Figure 5.4 shows the local behavior T_{Phil_i}, T_{Fork_i} respectively T_{Sem_i} for $0 \leq i < n$.

Results & Discussion

Table 5.2 shows results from our tool for instances of our model of Tanenbaum's solution of the Dining Philosophers problem. We considered instances with $n = 5$, 10, 20, 100, 200 and 400 philosophers and constructed abstract over-approximations based on all reasonable subsets with $d = 3$ components. If we apply our approach to establish deadlock-freedom then there is no state in any abstract over-approximation that remains to be a



(a) T_{Phil_i} (b) T_{Fork_i} (c) T_{Sem_i}

Figure 5.4: Local behavior of the components $Phil_i$, $Fork_i$ and Sem_i in Sys_n for $0 \leq i < n$.

possible projection of a reachable global deadlock, i.e., our approach to establish deadlock-freedom succeeds for all instances. We want to point out, that the waiting chain approach ([Min10] and informally described in Chapter 4) is not able to establish deadlock-freedom for instances of this system on abstract over-approximations that consists of $d = 3$ components.

5.4 A Chain of Components

This example provides a rather abstract parameterized interaction system that consists of components that are connected such that the corresponding interaction graph forms a chainlike structure of components. This interaction system has the property that the Cross-Checking approach (adapted to our context), that is described in [Min10], does not have any refinement effect on abstract over-approximations of instances of this system for $d = 3$.

Let $n \geq 2$. The system consists of three kinds of components C_i^a for $0 \leq i < n$ and C_i^b, C_i^c for $0 \leq i < n-1$. The components model certain processes. In the following we will not distinguish between a component and the process that is modeled by this component. We refer to a process that is modeled by a component of the form C_i^a, C_i^b respectively C_i^c as an a-process, b-process respectively c-process. For $0 \leq i < n-1$ C_i^a can connect to C_i^b if C_i^c is not

System	5	10	20	100	200	400
$\|K\|$	15	30	60	300	600	1,200
$\|\,\mathrm{Int}\,\|$	20	40	80	400	800	1,600
$\|Q\|$	$2^{21.61}$	$2^{43.22}$	$2^{86.44}$	$2^{432.19}$	$2^{864.39}$	$2^{1728.77}$
$\|\mathbb{C}\|$	185	390	780	3,900	7,800	15,600
$\Sigma\|Q_C\|$	3,520	7,200	14,400	72,000	144,000	288,000
$\Sigma Reach$	2,680	5,500	11,000	55,000	110,000	220,000
ΣCC	1,660	3,460	6,920	34,600	69,200	138,400
ΣEM	1,635	3,410	6,820	34,100	68,200	136,400
%	53.55	52.64	52.64	52.64	52.64	52.64
time	356	783	1,873	16,476	47,438	207,212
crit	0	0	0	0	0	0

Table 5.2: Benchmarks of Tanenbaum's Philosophers for $d = 3$.

connected to another process and vice versa. We call C_i^b and C_i^c the front processes with respect to C_i^a. Analogously, for $1 \leq i < n$ C_i^a can connect to C_{i-1}^b if C_{i-1}^c is not connected and vice versa and these processes are called the back processes with respect to C_i^a. If C_i^a $(0 \leq i < n)$ is connected to a process then the two processes will perform simultaneously $l \geq 0$ working steps. After two processes completed their working steps they disconnect.

Interaction Model

Let $\mathrm{IM}_n = (K, \{A_i\}_{i \in K}, \mathrm{Int})$ (with $n \geq 2$) be the interaction model that is specified as follows. The set of components K is given by

$$K = \{C_i^a, C_i^b, C_i^c | 0 \leq i < n-1\} \cup \{C_{n-1}^a\}.$$

The set of ports $A_{C_i^a}$ for a component C_i^a with $1 \leq i < n-1$ consists of the following ports.

$conn_i^f$: connect to a front process
$conn_i^b$: connect to a back process
$work_i^f$: do a working step with a front process
$work_i^b$: do a working step with a back process
dis_i^f : disconnect from a front process
dis_i^b : disconnect from a back process

The port set $A_{C_0^a}$ respectively $A_{C_{n-1}^a}$ is specified analogously without the ports that model a communication with a back respectively a front process.

The set of ports $A_{C_i^b}$ for a component C_i^b with $0 \leq i < n-1$ consists of the following ports.

$conn\_b_i$: connect to an a-process
$work\_b_i$: do a working step with an a-process
dis_b_i : disconnect from an a-process
$free\_b_i$: this process is not connected to any process

Analogously, the set of ports $A_{C_i^c}$ for a component C_i^c with $0 \leq i < n-1$ consists of the following ports.

$conn\_c_i$: connect to an a-process
$work\_c_i$: do a working step with an a-process
dis_c_i : disconnect from an a-process
$free\_c_i$: this process is not connected to any process

Let the interaction set Int of IM_n consist of the following interactions.

For $0 \leq i < n-1$ the component C_i^a

- connects to its front b-process: $\{conn_i^f, conn\_b_i, free\_c_i\}$
- connects to its front c-process: $\{conn_i^f, conn\_c_i, free\_b_i\}$
- does a working step with its front b-process: $\{work_i^f, work\_b_i\}$

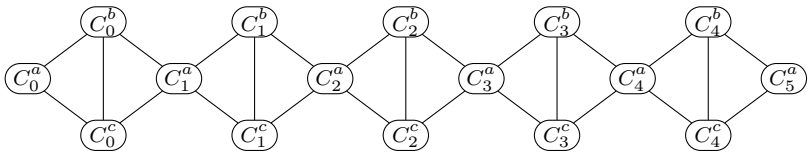


Figure 5.5: Interaction graph G of IM_6 in the chaining components example.

- does a working step with its front c-process: $\{work_i^f, work\_c_i\}$
- disconnects from its front b-process: $\{dis_i^f, dis\_b_i\}$
- disconnects from its front c-process: $\{dis_i^f, dis\_c_i\}$

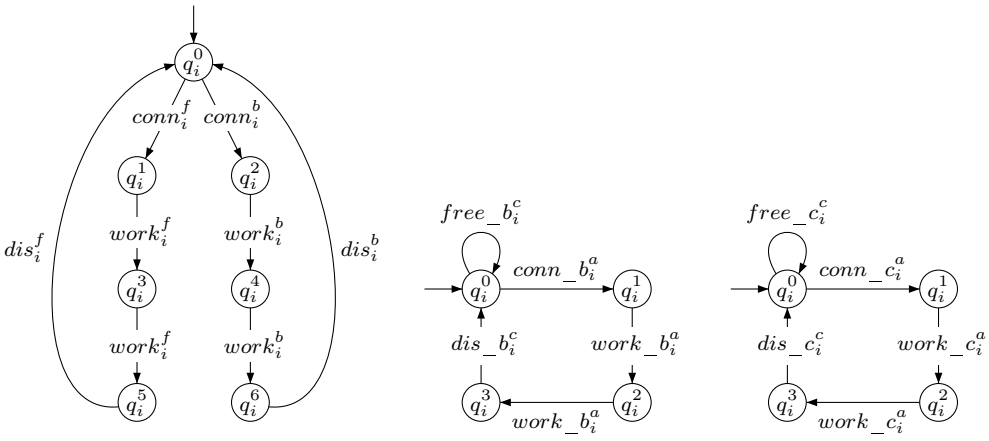
For $1 \leq i < n$ the component C_i^a

- connects to its back b-process: $\{conn_i^b, conn\_b_{i-1}, free\_c_{i-1}\}$
- connects to its back c-process: $\{conn_i^b, conn\_c_{i-1}, free\_b_{i-1}\}$
- does a working step with its back b-process: $\{work_i^b, work\_b_{i-1}\}$
- does a working step with its back c-process: $\{work_i^b, work\_c_{i-1}\}$
- disconnects from its back b-process: $\{dis_i^b, dis\_b_{i-1}\}$
- disconnects from its back c-process: $\{dis_i^b, dis\_c_{i-1}\}$

Figure 5.5 depicts the interaction graph of the interaction model IM_6.

Interaction System

Let $n \geq 2$ and $\mathrm{IM}_n = (K, \{A_i\}_{i \in K}, \mathrm{Int})$ be the interaction model for the chaining components example. Let $l \geq 0$ and $\mathrm{Sys}_n^l = (\mathrm{IM}_n, \{T_i\}_{i \in K})$ be the interaction system with the local behaviors of the components in K. Figure 5.6 depicts the respective local behaviors for $l = 2$. 5.6a depicts $T_{C_i^a}$ for $1 \leq i < n-1$ and 5.6b $T_{C_i^b}$, 5.6c $T_{C_i^c}$ for $0 \leq i < n-1$. It is easy to see that the global behavior T of Sys_n^l is deadlock-free for any $n \geq 2$ and $l \geq 0$. Note that $T_{C_0^a}$ respectively $T_{C_{n-1}^a}$ are structured analogously without



(a) $T_{C_i^a}$ (b) $T_{C_i^b}$ (c) $T_{C_i^c}$

Figure 5.6: Local behavior of the components C_i^a for $1 \leq i < n-1$ and C_i^b, C_i^c in Sys_n^l for $0 \leq i < n-1$ and $l = 2$.

the transitions that are labeled by ports that model a cooperation with a back respectively front process.

Results & Discussion

Table 5.3 respectively table 5.4 shows results of our tool on instances of the chaining components example. The columns are labeled by (n, l), i.e., we considered systems of the form Sys_n^l. We considered families of abstract over-approximations based on $d = 3$ (Table 5.3) and $d = 4$ (Table 5.4) components.

We want to point out two observations that are visible in the tables. In Table 5.3 the Cross-Checking approach [Min10] does not have any refinement effect on the abstract over-approximations. In Chapter 3 we already mentioned that our Edge-Match operator is stronger that the Cross-Checking operator because, in contrast to a refinement approach based on the Cross-Checking operator, we comprise the behavior of an interaction system in our refinement process. This example shows that there are even systems where the Cross-Checking does not refine abstract over-approximations, whereas

System	$(5,2)$	$(10,4)$	$(20,5)$	$(50,5)$	$(100,7)$
$\|K\|$	13	28	58	148	298
$\|\,\mathrm{Int}\,\|$	48	108	228	588	1,188
$\|Q\|$	$2^{28.42}$	$2^{79.37}$	$2^{178.9}$	$2^{458.36}$	$2^{1034.56}$
$\|\mathbb{C}\|$	28	68	148	388	788
$\Sigma\|Q_C\|$	3,376	31,188	112,252	297,892	1,321,236
$\Sigma Reach$	2,894	26,128	93,570	249,030	1,090,204
ΣCC	2,894	26,128	93,570	249,030	1,090,204
ΣEM	2,302	19,028	66,076	175,396	738,692
%	31.81	38.99	41.14	41.12	44.09
time	54	334	991	3,524	18,732
crit	378	2,498	7,752	20,592	70,838

Table 5.3: Benchmarks of the chaining components example for $d = 3$.

an application of the Edge-Match operator results in a great amount of unreachable states. The other observation we want to point out is that our approach to establish deadlock-freedom fails for all instances in Table 5.3, i.e., if we consider abstract over-approximations based on subsets of $d = 3$ components then our approach fails for all examined instances. On the other hand our approach succeeds for the same instances if we base our analysis on subsets of $d = 4$ components (see Table 5.4). This is, if our approach fails on a family of abstract over-approximations then considering another parameter d might establish a property in consideration.

5.5 A Circle of Components

The following example describes an abstract parameterizes interaction system. The interaction system is similar to the interaction system described in Section 5.4 and provides a system where the Cross-Checking approach (adapted to our context), that is described in [Min10] has a significant

System	$(5,2)$	$(10,4)$	$(20,5)$	$(50,5)$	$(100,7)$
$\|K\|$	13	28	58	148	298
$\|\,\mathrm{Int}\,\|$	48	108	228	588	1,188
$\|Q\|$	$2^{28.42}$	$2^{79.37}$	$2^{178.9}$	$2^{458.36}$	$2^{1034.56}$
$\|\mathbb{C}\|$	40	105	235	625	1,275
$\Sigma\|Q_C\|$	23,968	374,220	1,632,631	4,403,581	25,418,043
$\Sigma Reach$	18,292	279,740	1,215,679	3,292,789	18,727,547
ΣCC	5,212	45,180	159,439	433,429	1,754,427
ΣEM	5,212	45,180	159,439	433,429	1,754,427
%	78.25	87.93	90.23	90.16	93.1
time	188	1,339	4,192	14,035	75,385
crit	0	0	0	0	0

Table 5.4: Benchmarks of the chaining components example for $d = 4$.

impact regarding the refinement of abstract over-approximations and the Edge-Match operator creates even better refined abstract over-approximations. The corresponding interaction graph G of a model instance forms a circle-like structure.

Let $n \geq 2$. Similarly as in the example in Section 5.4, the system consists of three kinds of components C_i^a, C_i^b and C_i^c for $0 \leq i < n$. The components model certain processes. In the following we will not distinguish between a component and the process that is modeled by this component. We refer to a process that is modeled by a component of the form C_i^a, C_i^b respectively C_i^c as an a-process, b-process respectively c-process. For $0 \leq i < n$ C_i^a can connect to C_i^b if C_i^c is not connected to another process and vice versa. We call C_i^b and C_i^c the front processes with respect to C_i^a. Analogously, for $0 \leq i < n$ C_i^a can connect to C_{i-1}^b if C_{i-1}^c is not connected and vice versa and these processes are called the back processes with respect to C_i^a. The term $i-1$ refers to $n-1$ if $i = 0$. If C_i^a $(0 \leq i < n)$ is connected to a process then the two processes will perform simultaneously $l \geq 1$ working steps. The last

working step corresponds to a disconnection of the processes. For $0 \leq i < n$ the component C_i^a is able to synchronize with its front processes C_i^b and C_i^c if both components are connected to an a-process.

Interaction Model

Let $\mathrm{IM}_n = (K, \{A_i\}_{i \in K}, \mathrm{Int})$ be an interaction model with $n \geq 2$. The set of components K is given by

$$K = \{C_i^a, C_i^b, C_i^c | 0 \leq i < n\}.$$

The set of ports $A_{C_i^a}$ for a component C_i^a with $0 \leq i < n$ consists of the following ports.

$conn_i^f$: connect to a front process
$conn_i^b$: connect to a back process
$work_i^f$: do a working step with a front process
$work_i^b$: do a working step with a back process
$sync_i$: synchronize with the front processes

The set of ports $A_{C_i^b}$ for a component C_i^b with $0 \leq i < n-1$ consists of the following ports.

$conn\_b_i$: connect to an a-process
$work\_b_i$: do a working step with an a-process
$sync\_b_i$: synchronize with an a-process
$free\_b_i$: this process is not connected to any process

Analogously, the set of ports $A_{C_i^c}$ for a component C_i^c with $0 \leq i < n-1$ consists of the following ports.

$conn\_c_i$: connect to an a-process
$work\_c_i$: do a working step with an a-process
$sync\_c_i$: synchronize with an a-process
$free\_c_i$: this process is not connected to any process

Let the interaction set Int of IM_n consist of the following interactions.

For $0 \leq i < n$ the component C_i^a

- connects to its front b-process: $\{conn_i^f, conn\_b_i, free\_c_i\}$
- connects to its front c-process: $\{conn_i^f, conn\_c_i, free\_b_i\}$
- does a working step with its front b-process: $\{work_i^f, work\_b_i\}$
- does a working step with its front c-process: $\{work_i^f, work\_c_i\}$
- synchronizes with its front processes: $\{sync_i, sync\_b_i, sync\_c_i\}$
- connects to its back b-process: $\{conn_i^b, conn\_b_{i-1}, free\_c_{i-1}\}$
- connects to its back c-process: $\{conn_i^b, conn\_c_{i-1}, free\_b_{i-1}\}$
- does a working step with its back b-process: $\{work_i^b, work\_b_{i-1}\}$
- does a working step with its back c-process: $\{work_i^b, work\_c_{i-1}\}$

Note that $i-1$ refers to $n-1$ if $i = 0$.

Figure 5.7 depicts the interaction graph of the interaction model IM_{10}.

Interaction System

Let $n \geq 2$ and $\mathrm{IM}_n = (K, \{A_i\}_{i \in K}, \mathrm{Int})$ be the interaction model for the circle-like components example. Let $l \geq 1$ and $\mathrm{Sys}_n^l = (\mathrm{IM}_n, \{T_i\}_{i \in K})$ be the interaction system with the local behaviors of the components in K. Figure 5.8 depicts the respective local behaviors for $l = 2$. 5.8a depicts $T_{C_i^a}$, 5.8b $T_{C_i^b}$ and 5.8c $T_{C_i^c}$ for $0 \leq i < n$. It is easy to see that the global behavior T of Sys_n^l is deadlock-free for any $n \geq 2$ and $l \geq 1$.

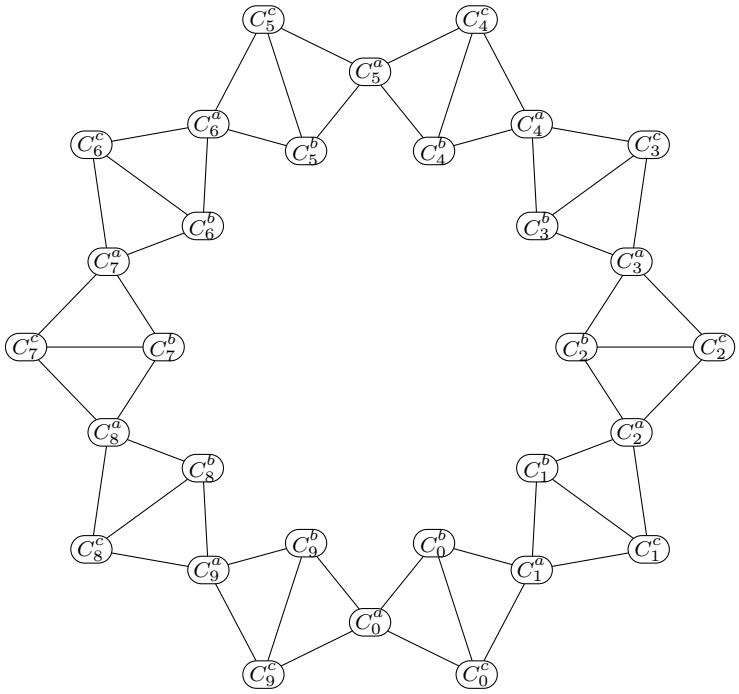
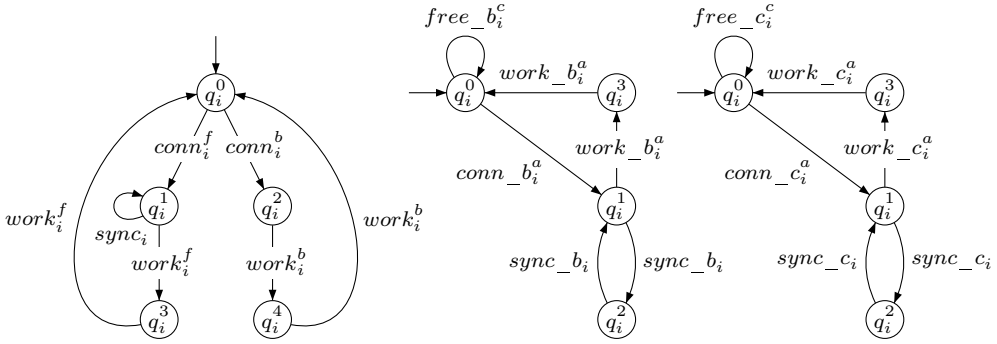


Figure 5.7: Interaction graph G of IM_{10} in the circle-like components example.



(a) $T_{C_i^a}$ (b) $T_{C_i^b}$ (c) $T_{C_i^c}$

Figure 5.8: Local behavior of the components C_i^a, C_i^b and C_i^c in Sys_n^l for $0 \leq i < n$ and $l = 2$.

System	$(5,2)$	$(10,4)$	$(20,5)$	$(50,5)$	$(100,7)$
$\|K\|$	15	30	60	150	300
$\|\,\mathrm{Int}\,\|$	45	90	180	450	900
$\|Q\|$	$2^{31.61}$	$2^{83.4}$	$2^{181.48}$	$2^{453.71}$	$2^{1024.67}$
$\|\mathbb{C}\|$	40	80	160	400	800
$\Sigma\|Q_C\|$	3,400	29,160	98,560	246,400	1,134,000
$\Sigma Reach$	2,580	22,420	76,040	190,100	878,600
ΣCC	1,890	18,520	64,760	161,900	777,800
ΣEM	1,640	15,040	51,560	128,900	602,800
%	51.76	48.42	47.69	47.69	46.84
time	99	464	1,333	4,483	22,504
crit	280	1,760	5,200	13,000	47,600

Table 5.5: Benchmarks of the circle-like components example for $d = 3$.

Results & Discussion

We applied our approach to various instances of the chaining-like components example with abstract over-approximations based on subsets of $d = 3$ (Table 5.5) and $d = 4$ (Table 5.6) components. The columns are labeled by (n, l), i.e., we considered systems of the form Sys_n^l. Even though the obvious similarity between this system and the chaining components system introduced in Section 5.4, Table 5.5 shows that the Cross-Checking operator has a significant refinement effect on the instances. The Edge-Match operator however produces even more refined abstract over-approximations on the considered instances with $d = 3$. If we consider abstract over-approximations based on subsets of size $d = 4$ then the Edge-Match operator has no advantage over the Cross-Checking operator, i.e., both approaches produce abstract over-approximations with the same reachable state space.

System	$(5,2)$	$(10,4)$	$(20,5)$	$(50,5)$	$(100,7)$
$\|K\|$	15	30	60	150	300
$\|\,\mathrm{Int}\,\|$	45	90	180	450	900
$\|Q\|$	$2^{31.61}$	$2^{83.4}$	$2^{181.48}$	$2^{453.71}$	$2^{1024.67}$
$\|\mathbb{C}\|$	65	130	260	650	1,300
$\Sigma\|Q_C\|$	24,400	340,200	1,369,060	3,422,650	20,776,500
$\Sigma Reach$	15,585	222,970	904,520	2,261,300	13,880,200
ΣCC	4,505	36,850	120,560	301,400	1,309,600
ΣEM	4,505	36,850	120,560	301,400	1,309,600
%	81.54	89.17	91.19	91.19	93.7
time	279	1,500	4,350	14,169	71,478
crit	0	0	0	0	0

Table 5.6: Benchmarks of the circle-like components example for $d = 4$.

5.6 Production Cell

The *Production Cell* is a small system that describes the automatic processing of metal blanks. The system includes a feed belt, a rotating table, a robot unit with two arms, arm one and arm two, which are assembled on one swivel and can only move simultaneously, a press that processes the metal blanks, a deposit belt and a crane. The feed belt can transport a metal blank to the rotating table. The table rotates such that arm one can lift the blank into the press. After the press processed the blank, arm two withdraws the product and moves it to the deposit belt that transports the product into the scope of the crane. The crane can lift the product back on the feed belt where it is used again as a metal blank. Thus, one metal blank can be processed infinitely often. The System is described in detail in [LL95]. Here we model an abstract version of the system by an interaction system. Our model is partly based on a Petri net model that is described in [HD95]. We provide here merely a brief description of the system and our model. See [LL95] and [HD95] for further details.

We model each unit of the system as a component with the exception that the robot unit is modeled by three components – the two arms and the swivel are modeled separately. Between two neighboring units in the processing cycle there is a connecting area that is modeled by a component as well. An area models the three cases that the next unit in the process is busy, i.e., the area is blocked, the next unit is waiting and there is no metal blank/product available, i.e., the area is free and the case that the unit is waiting and a metal blank/product is available. Note that this system is not parameterized by a parameter that affects the number of components.

Interaction Model

Let IM $= (K, \{A_i\}_{i \in K}, \text{Int})$ be an interaction model with the set of components $K = K_{units} \cup K_{areas}$ where

$K_{areas} = \{FT, TA1, A1P, PA2, A2D, DC, CF, \}$ and
$K_{units} = \{feedBelt, table, arm1, swivel, press, arm2, depositBelt, crane\}$.

Note that the names of the components that model the areas consists of the first letters of the units that are connected by the area, e.g., the component FT models the area that connects the feed belt and the table and $PA2$ models the area that connects the press and arm two.

In the following we specify the set of ports for each component. Various components that model units and areas in the model share a similar behavior, i.e., they exhibit a similar set of ports. For $i \in \{feedBelt, depositBelt\}$ let A_i consist of the following ports.

$occupy_i$: the belt becomes occupied by a metal blank
$transport_i$: the belt transports a blank to the output area
$empty_i$: a metal blank becomes unloaded from the belt
$goidle_i$: the belt goes into an idle state

For $i \in \{table, press\}$ let A_i consist of the following ports.

$moveUnload_i$: an available blank moves to the unload position
$readyUnload_i$: a blank becomes unloaded
$moveLoad_i$: move to the loading position
$ready_i$: become available for input

For $i \in \{arm1, arm2\}$ let A_i consist of the following ports.

$gowait1_i$: wait for the swivel to rotate towards a blank
$load_i$: load a blank
$goSwivel1_i$: rotate toward the output area
$store_i$: a loaded blank waits for becoming unloaded
$gowait2_i$: wait for the swivel to rotate into the output area
$unload_i$: become ready for unloading
$goSwivel2_i$: turn into the output area and unload a blank
$free_i$: wait for a new blank

The set of ports A_{swivel} consists of the following ports.

$take_{swivel}$: an arm wants to use the swivel in order to rotate
put_{swivel} : an arm finished a rotating process

The set of ports A_{crane} of the component that models the crane consists of the following ports.

$load_{crane}$: load an available blank
$store_{crane}$: a loaded blank waits for being moved
$unload_{crane}$: unload a blank into the output area
$free_{crane}$: become available for a new blank

For $i \in K_{areas}$ let the set of ports A_i consist of the following ports.

$lockInput_i$: the area is blocked by a blank
$unlockInput_i$: a blank was processed by the next unit
$lockOutput_i$: previous unit wants to load a blank into this area
$unlockOutput_i$: loads a blank from the previous unit

The following interactions describe the cooperation between each unit and its input respectively output area. Let Int consists of exactly these interactions.

The component that models the crane interacts with the areas DC and CF:

$$\begin{aligned} lockInputDC &= \{lockInput_{DC}, load_{crane}\} \\ unlockInputDC &= \{unlockInput_{DC}, store_{crane}\} \\ lockOutputCF &= \{lockOutput_{CF}, unload_{crane}\} \\ unlockOutputCF &= \{unlockOutput_{CF}, free_{crane}\} \end{aligned}$$

Component $feedBelt$ interacts with the areas CF and FT:

$$\begin{aligned} lockInputCF &= \{lockInput_{CF}, occupy_{feedBelt}\} \\ unlockInputCF &= \{unlockInput_{CF}, empty_{feedBelt}\} \\ lockOutputFT &= \{lockOutput_{FT}, transport_{feedBelt}\} \\ unlockOutputFT &= \{unlockOutput_{FT}, goidle_{feedBelt}\} \end{aligned}$$

The component $depositBelt$ that models the deposit belt interacts with the components $A2D$ and DC:

$$\begin{aligned} lockInputA2D &= \{lockInput_{A2D}, occupy_{depositBelt}\} \\ unlockInputA2D &= \{unlockInput_{A2D}, empty_{depositBelt}\} \\ lockOutputDC &= \{lockOutput_{DC}, transport_{depositBelt}\} \\ unlockOutputDC &= \{unlockOutput_{DC}, goidle_{depositBelt}\} \end{aligned}$$

The table interacts with the areas FT and $TA1$:

$$\begin{aligned} lockInputFT &= \{lockInput_{FT}, moveUnload_{table}\} \\ unlockInputFT &= \{unlockInput_{FT}, goready_{table}\} \\ lockOutputTA1 &= \{lockOutput_{TA1}, moveLoad_{table}\} \\ unlockOutputTA1 &= \{unlockOutput_{TA1}, readyUnload_{table}\} \end{aligned}$$

Component $press$ interacts with the areas $A1P$ and $PA2$:

$$\begin{aligned} lockInputA1P &= \{lockInput_{A1P}, moveUnload_{press}\} \\ unlockInputA1P &= \{unlockInput_{A1P}, goready_{press}\} \\ lockOutputPA2 &= \{lockOutput_{PA2}, moveLoad_{press}\} \\ unlockOutputPA2 &= \{unlockOutput_{PA2}, readyUnload_{press}\} \end{aligned}$$

The component $arm1$ that models the robot arm one interacts with the areas $TA1$ and $A1P$ and with the component that models the swivel:

$$\begin{aligned}
lockInputTA1 &= \{lockInput_{TA1}, gowait1_{arm1}\} \\
unlockInputTA1 &= \{unlockInput_{TA1}, goSwivel1_{arm1}\} \\
lockOutputA1P &= \{lockOutput_{A1P}, gowait2_{arm1}\} \\
unlockOutputA1P &= \{unlockOutput_{A1P}, goSwivel2_{arm1}\} \\
takeSwivel1Arm1 &= \{take_{swivel}, load_{arm1}\} \\
putSwivel1Arm1 &= \{put_{swivel}, store_{arm1}\} \\
takeSwivel2Arm1 &= \{take_{swivel}, unload_{arm1}\} \\
putSwivel2Arm1 &= \{put_{swivel}, free_{arm1}\}
\end{aligned}$$

The component $arm2$ that models the robot arm two interacts with the areas $PA2$ and $A2D$ and with the component that models the swivel:

$$\begin{aligned}
lockInputPA2 &= \{lockInput_{PA2}, gowait1_{arm2}\} \\
unlockInputPA2 &= \{unlockInput_{PA2}, goSwivel1_{arm2}\} \\
lockOutputA2D &= \{lockOutput_{A2D}, gowait2_{arm2}\} \\
unlockOutputA2D &= \{unlockOutput_{A2D}, goSwivel2_{arm2}\} \\
takeSwivel1Arm2 &= \{take_{swivel}, load_{arm2}\} \\
putSwivel1Arm2 &= \{put_{swivel}, store_{arm2}\} \\
takeSwivel2Arm2 &= \{take_{swivel}, unload_{arm2}\} \\
putSwivel2Arm2 &= \{put_{swivel}, free_{arm2}\}
\end{aligned}$$

Figure 5.9 depicts the interaction graph of the interaction model IM.

Interaction System

Let Sys = $(\text{IM}, \{T_i\}_{i \in K})$ be the interaction system that consists of the interaction model of the Production Cell example and the local behavior of the components that is depicted in the following figures. Note that the local initial state of the behavior of components in $\{DC, CF, FT, A1P, A2D\}$ is *avail*. This state models that a blank is currently in this area which is ready for further processing. The components in $\{PA2, TA1\}$ exhibit

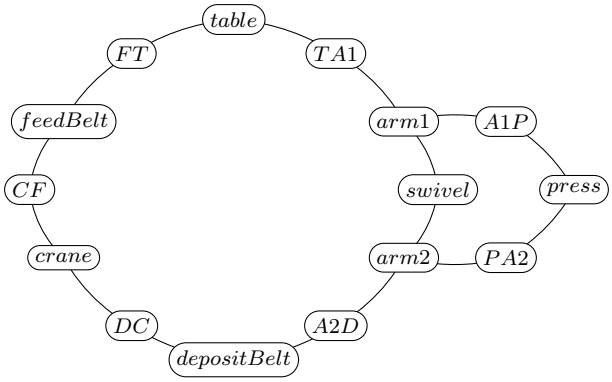
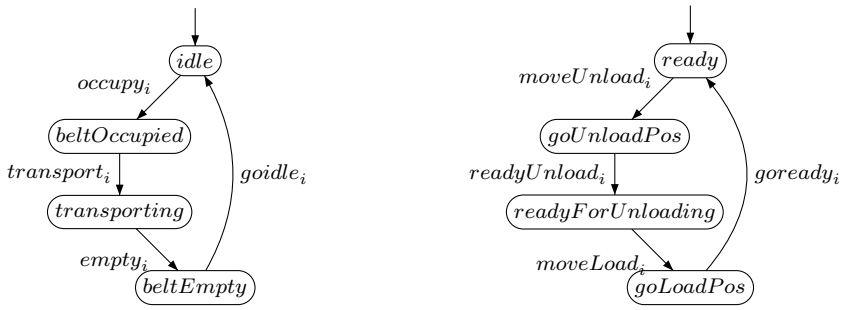


Figure 5.9: Interaction graph G of IM in the Production Cell example.

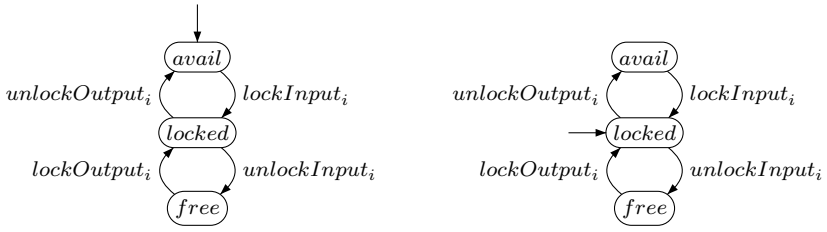


(a) T_i, $i \in \{feedBelt, depositBelt\}$ (b) T_i, $i \in \{table, press\}$

Figure 5.10: Local behavior of the components that model the feed belt, the deposit belt, the table and the press in the Production Cell example.

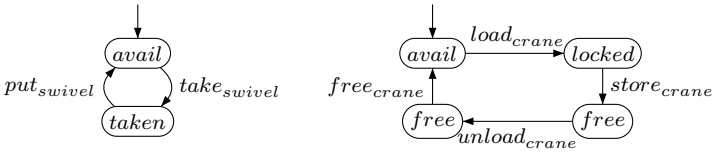
the initial state *locked* which models that these areas are ready to receive a blank from the previous unit. This is, there are initially 5 blanks in the system that are in the areas that are modeled by the components in $\{DC, CF, FT, A1P, A2D\}$.

Figure 5.10 depicts the local behavior of the components that model the feed belt, the deposit belt, the table and the press, Figure 5.11 the local behavior of the areas, Figure 5.12 the local behavior of the component that models the swivel and the crane and Figure 5.13 depicts the local behavior of the components that model the robot arm one and two.



(a) T_i, $i \in \{DC, CF, FT, A1P, A2D\}$ (b) T_i, $i \in \{PA2, TA1\}$

Figure 5.11: Local behavior of the components that model areas in the Production Cell example.



(a) T_{swivel} (b) T_{crane}

Figure 5.12: Local behavior of the components that model the swivel and the crane in the Production Cell example.

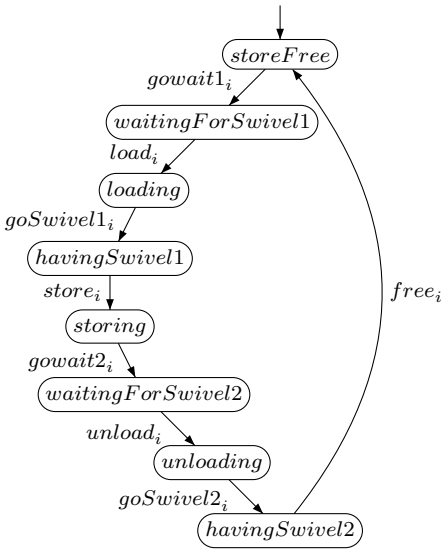


Figure 5.13: Local behavior T_i, $i \in \{arm1, arm2\}$, of the components that model arm one and arm two in the Production Cell example.

Results & Discussion

Our model of the Production Cell example is not parameterized and, in comparison to instances in our other examples, relatively small – there are only 15 components in the model. Table 5.7 shows the results of our refinement approach and our approach to establish deadlock-freedom applied to the Production Cell example. The columns are labeled by the parameter d that we used in our experiments. We considered families of abstract over-approximations based on subsets consisting of 3, 6, 9, 12 and 15 components. Note that there is only one abstract over-approximation if we consider $d = 15$ and that this abstract over-approximation corresponds to the global behavior of the system (thus, the required time to calculate the Edge-Match fixed-point equals 0 milliseconds). Furthermore the table shows that the Edge-Match operator produces only slightly more refined abstract over-approximations in comparison with the Cross-Checking operator and that our approach to establish deadlock-freedom succeeds for an analysis with $d = 12$, i.e., our model is deadlock-free.

5.7 Conclusion

In this chapter we presented results of a tool that implements our approach to refine abstract over-approximations and our approach to establish deadlock-freedom by an analysis of these abstract over-approximations. Furthermore, we provided a comparison between our refinement approach and a refinement approach that is based on the Cross-Checking operator [Min10]. The results are calculated from various complex and parametrized examples.

The results present the strength of our refinement approach and reflect that there are complex systems where our approach is able to conclude in a reasonable amount of time that a great amount of states in the initial abstract over-approximations are not projections of reachable global states. Furthermore, the results show that we can establish deadlock-freedom in interaction

System	3	6	9	12	15
$\|K\|$	15	15	15	15	15
$\|\,\mathrm{Int}\,\|$	36	36	36	36	36
$\|Q\|$	$2^{28.09}$	$2^{28.09}$	$2^{28.09}$	$2^{28.09}$	$2^{28.09}$
$\|\mathbb{C}\|$	19	39	60	70	1
$\Sigma\|Q_C\|$	1,172	138,240	10,153,728	523,542,528	286,074,857
$\Sigma Reach$	652	18,390	276,104	3,050,172	13,107
ΣCC	524	11,418	156,617	1,310,519	13,107
ΣEM	521	11,418	155,873	1,310,291	13,107
%	55.55	91.74	98.46	99.75	100.0
time	63	708	5,596	52,427	0
crit	60	187	120	0	0

Table 5.7: Benchmarks of the Production Cell example for various values of the parameter d.

systems with a large number of components.

As already argued in Chapter 3, our refinement approach is at least as strong as the Cross-Checking approach. Our results show that there are systems where our approach produces significantly more refined abstract over-approximations. Particularly in the chaining components example (Section 5.4), our approach produces for $d = 3$ in the system Sys^7_{100} (see Table 5.3) abstract over-approximations where the number of all reachable states is considerably smaller in comparison to the abstract over-approximations constructed by the Cross-Checking approach. In fact, in this example, our approach produces abstract over-approximations with more than 30% less reachable states. Furthermore, the results depicted in Table 5.3 show that there are systems where the Cross-Checking approach does not have any refinement effect at all, whereas our approach results in abstract over-approximations where a great amount of initial reachable states become unreachable. Presented in [Min10] is a prototype tool *PrInSESSA* that im-

plements a fixed-point calculation of abstract over-approximations based on the Cross-Checking operator. *PrInSESSA* is implemented in pure Java and does not use BDDs as an underlying data structure. Thus, *PrInSESSA* is in comparison with our tool considerably more slowly and can only handle considerably less complex systems.

There are, best to our knowledge, no other tools that implement a comparable approach. We just want to mention that all *PROMELA* models of the Philosophers Problem that we found can only be analyzed in an acceptable runtime in the famous LTL model checker SPIN [Hol97] for a considerably smaller number of philosophers. However, this observation can not be used or extended to a relevant comparison because SPIN analyzes (without additional adjustments) the entire reachable state space of a system.

Chapter 6

A Connection to Relational Algebraic Operators

6.1 Introduction

In this chapter we present a connection between our approach to refine abstract over-approximations by the Edge-Match operator, that we introduced in Chapter 3, and the theory of relational algebra and its operators [MCS13a]. We show that a family of abstract over-approximations can be modeled as relations in a database on a relational database scheme and model our refinement operator by relational algebraic operators. We use this connection to derive a proposition regarding the "preciseness" of our refinement technique that was introduced in Chapter 3. For this purpose we consider acyclic relational database schemata. These schemata form an important subclass in the theory of relational databases as they fulfill various interesting properties and several operations become decomposable, i.e., an operation on a general database the application of which requires an expensive calculation that involves all tables in the database can be decomposed to less expensive operations on subsets or even pairs of tables (see, e.g., [Yan81] for efficient algorithms on databases that are based on

acyclic database schemata). We show here that the fixed-point of a family of abstract over-approximations with respect to the Edge-Match operator is legitimate (see Definition 3.7) if the hypergraph that is based on the domain of the family is acyclic. Results from the theory of relational databases have been exploited in other fields as well, e.g., a generalization of these concepts in the field of set theory can be found in [Heg91]. A relational algebraic approach for establishing system properties is used in [LL88] where the state space of cooperating protocols is modeled as one relation and various properties are checked by relational queries. In [KK96] a Petri net is modeled by a relation for markings and one for the places and transitions. Properties can be checked by algorithms on these relations that make use of relational algebraic operators.

In the first part of this chapter, that consists of Section 6.2 and 6.3, we describe how abstract over-approximations can be interpreted in terms of relations and how the Edge-Match operator can be modeled by operations from the relational algebra. Particularly the semijoin on a relation r with a relation s is used for this purpose, that (roughly speaking) restricts a relation r to tuples t for which there is a tuple t' in s such that t and t' coincide on their shared attributes. We start the first part by repeating the notions of the relational algebra that we need in the remainder.

The second part of this chapter, Section 6.4, uses the result of the first part in order to derive a proposition that states that the fixed-point of a family of abstract over-approximations is legitimated if the domain on which the family is based has a certain structure.

Section 6.5 concludes this chapter.

6.2 Relational Algebra

Here we give a brief compendium of notions and operators from the relational algebra (see, e.g., [Mai83]) that are used in the remainder of this chapter.

Definition 6.1:
A **relational scheme** $\mathsf{R} = \{\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_p\}$ is a finite set of **attribute names** (**attributes** for short). The **domain** of an attribute A_i $(1 \leq i \leq p)$ is a set $\mathsf{D}_i = \mathrm{dom}(\mathsf{A}_i)$. Let $\mathsf{D}(\mathsf{R}) = \mathsf{D}_1 \cup \cdots \cup \mathsf{D}_p$ then a **relation** $\mathsf{r}(\mathsf{R})$ on R is a set of mappings $\mathsf{r}(\mathsf{R}) = \{\mathsf{t}_1, \mathsf{t}_2, \ldots, \mathsf{t}_k\}$ from R to $\mathsf{D}(\mathsf{R})$ such that for each $\mathsf{t} \in \mathsf{r}(\mathsf{R})$ and each $i \in \{1, \ldots, p\}$ $\mathsf{t}(\mathsf{A}_i) \in \mathsf{D}_i$. A mapping in a relation is called a **tuple**. In the following, we write r instead of $\mathsf{r}(\mathsf{R})$ if it is clear that r is a relation on R. Let $\mathsf{r}(\mathsf{R})$ be a relation and $\mathsf{S} \subseteq \mathsf{R}$ then for $\mathsf{t} \in \mathsf{r}$ $\mathsf{t}(\mathsf{S})$ denotes t restricted to the attributes in S.

Remark 6.1:
Note that the term *domain* was introduced in Chapter 3 in Definition 3.6 and entitles a set of subsets of components. In the remainder it is obvious from the context whether we speak about the domain of an attribute or the domain in the sense of Definition 3.6.

Here we give an example that we use to illustrate the concept of relational schemata and relations. We use this example in the remainder to illustrate various operators on relations.

Example 6.1:
Let $\mathsf{R}_1 = \{A, B, C\}$ and $\mathsf{R}_2 = \{B, C, D\}$ be relational schemata such that the domain of each attribute in R_1 and R_2 equals the natural numbers. It is common to depict relations in the form of tables. Table 6.1 depicts two relations $\mathsf{r}_1(\mathsf{R}_1)$ and $\mathsf{r}_2(\mathsf{R}_2)$. Each row corresponds to a tuple in the according relation. The columns are labeled by the attribute names.

In the remainder we make use of the following operations on relations.

Definition 6.2:
Let R and S be relation schemata and $\mathsf{r}(\mathsf{R})$ and $\mathsf{s}(\mathsf{S})$ be relations.

A	B	C
1	2	2
3	1	2
3	1	5
2	5	3

(a) $\mathsf{r}_1(\mathsf{R}_1)$

B	C	D
2	8	5
1	3	1
1	2	1
1	5	3
5	3	2

(b) $\mathsf{r}_2(\mathsf{R}_2)$

Table 6.1: Graphical representation of the relations $\mathsf{r}_1(\mathsf{R}_1)$ and $\mathsf{r}_2(\mathsf{R}_2)$.

- The **join** of r with s is a relation $\mathsf{u}(\mathsf{U}) = \mathsf{r} \bowtie \mathsf{s}$ on $\mathsf{U} = \mathsf{R} \cup \mathsf{S}$ with $\mathsf{t} \in \mathsf{u}$ if and only if there are tuples $\mathsf{t}_\mathsf{r} \in \mathsf{r}$ and $\mathsf{t}_\mathsf{s} \in \mathsf{s}$ such that $\mathsf{t}_\mathsf{r} = \mathsf{t}(\mathsf{R})$ and $\mathsf{t}_\mathsf{s} = \mathsf{t}(\mathsf{S})$.
- Let $\mathsf{U} \subseteq \mathsf{R}$ $(\mathsf{U} \neq \emptyset)$. The **projection** of r on U is a relation $\mathsf{u}(\mathsf{U}) = \pi_\mathsf{U}(\mathsf{r})$ with $\mathsf{u} = \{\mathsf{t}(\mathsf{U}) | \mathsf{t} \in \mathsf{r}\}$.
- The **semijoin** of r with s is a relation $\mathsf{r} \ltimes \mathsf{s} = \pi_\mathsf{R}(\mathsf{r} \bowtie \mathsf{s})$.

Example 6.2:
Consider the relations $\mathsf{r}_1(\mathsf{R}_1)$ and $\mathsf{r}_2(\mathsf{R}_2)$ from Example 6.1. The relation $\mathsf{r}(\mathsf{U}) = \mathsf{r}_1 \bowtie \mathsf{r}_2$ for $\mathsf{U} = \mathsf{R}_1 \cup \mathsf{R}_2$ is depicted in Table 6.2.

A	B	C	D
3	1	2	1
3	1	5	3
2	5	3	2

Table 6.2: The relation $\mathsf{r}(\mathsf{U}) = \mathsf{r}_1 \bowtie \mathsf{r}_2$.

Example 6.3:
Consider the relations $\mathsf{r}_1(\mathsf{R}_1)$ and $\mathsf{r}_2(\mathsf{R}_2)$ from Example 6.1. The relation $\mathsf{r}(\mathsf{R}_1) = \mathsf{r}_1 \ltimes \mathsf{r}_2$ is depicted in Table 6.3.

A	B	C
3	1	2
3	1	5
2	5	3

Table 6.3: The relation $\mathsf{r}(\mathsf{R}_1) = \mathsf{r}_1 \ltimes \mathsf{r}_2$.

6.3 The Relational Edge-Match Operator

We now introduce a mapping from transition relations of transition systems that are based on a subset of components of an interaction system to relations on a relational scheme and show, how the Edge-Match operator can be modeled by using the semijoin operator. The mapping is straight forward because we interpret a transition relation as a relational algebraic relation, i.e., the mapping merely is a mapping between notations.

Definition 6.3:
Let Sys $= (\mathrm{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model $\mathrm{IM} = (K, \{A_i\}_{i \in K}, Int)$, $C \subseteq K$ $(C \neq \emptyset)$ a subset of components and $R = (Q_C, Int, \rightarrow_R, q_C^0)$ a transition system (see Definition 3.2 in Chapter 3).

Let $\mathsf{R}_C = C^f \cup \{\mathrm{Interaction}\} \cup C^t$ be the relation scheme with

- $C^f = \{i^f | i \in C\}$ with domains $\mathrm{dom}(i^f) = Q_i$ for $i \in C$ and
- $C^t = \{i^t | i \in C\}$ with domains $\mathrm{dom}(i^t) = Q_i$ for $i \in C$ and
- $\mathrm{dom}(\mathrm{Interaction}) = Int$.

Note that f abbreviates "from" and t abbreviates "to" as the attributes in C^f model the left hand side of a transition and the attributes in C^t the right hand side. Thus, a tuple t in a relation on R_C is a function

$$\mathsf{t} : C^f \cup \{\mathrm{Interaction}\} \cup C^t \rightarrow \bigcup_{i \in C} Q_i \cup \mathrm{Int}\,.$$

Let t be a tuple like this then t models a transition $q_C \xrightarrow{\alpha}_R q'_C$ with $q_C = (q_i)_{i \in C}$ and $q'_C = (q'_i)_{i \in C}$ if and only if $\mathsf{t}(\text{Interaction}) = \alpha$ and for $i \in C$ holds $\mathsf{t}(i^f) = q_i$ and $\mathsf{t}(i^t) = q'_i$.

Let $\mathsf{r}_R(\mathsf{R}_C)$ be the relation on R_C that consists of the tuples that model the transitions in $\rightarrow_R$. The relation scheme R_C is called the **relation scheme associated with** C and $\mathsf{r}_R(\mathsf{R}_C)$ the **relation associated with** R.

We say a tuple $\mathsf{t} \in \mathsf{r}_R(\mathsf{R}_C)$ is **reachable** in the relation $\mathsf{r}_R(\mathsf{R}_C)$ if the corresponding transition that is modeled by t is reachable in R.

In the following we show how the Edge-Match operator can be modeled by the semijoin operator. After giving an example we show that the semijoin operator actually refines relations that model abstract over-approximations in the same way as the Edge-Match operator refines abstract over-approximations. The following examples shows the result of the semijoin operator applied on relations associated with abstract over-approximations from the running example that was introduced in Example 1.1 in Chapter 1.

Example 6.4:
Let $\mathsf{r}_S(\mathsf{R}_C)$ be the relation associated with S (Example 3.2) and $\mathsf{r}_R(\mathsf{R}_D)$ the relation associated with R (Example 3.3). Relation r_S is depicted in Table 6.4 and r_R in Table 6.5. Note that both relations consists of reachable tuples. The columns are labeled by the respective attributes.

Table 6.6 depicts the relation $\mathsf{r}_S \ltimes \mathsf{r}_R$, i.e., the result of the semijoin operator on the relations r_S and r_R. From the 18 tuples in r_S are 8 tuples removed in $\mathsf{r}_S \ltimes \mathsf{r}_R$. Note that $\mathsf{r}_S \ltimes \mathsf{r}_R$ coincides with the relation associated with R_C from Example 3.4, i.e., the result of the Edge-Match operator applied on R and S.

The following theorem states that the result of the semijoin operator, on relations that represents transition systems with respect to a subset of components and are restricted to reachable tuples, corresponds to an application of the Edge-Match operator on the transition systems.

TER_1^f	GS^f	ADB^f	Interaction	TER_1^t	GS^t	ADB^t
$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}	$send\_req_1$	$q^1_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}	$send\_req_2$	$q^0_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}
$q^1_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}	ask_auth	$q^1_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}
$q^1_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}	$authorize$	$q^1_{\text{TER}_1}$	q^2_{GS}	q^0_{ADB}
$q^1_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}	$send\_data$	$q^1_{\text{TER}_1}$	q^3_{GS}	q^1_{ADB}
$q^1_{\text{TER}_1}$	q^2_{GS}	q^0_{ADB}	$send\_data$	$q^1_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}
$q^1_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}	get_reply_1	$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}	ask_auth	$q^0_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}
$q^0_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}	$authorize$	$q^0_{\text{TER}_1}$	q^2_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}	$send\_data$	$q^0_{\text{TER}_1}$	q^3_{GS}	q^1_{ADB}
$q^0_{\text{TER}_1}$	q^2_{GS}	q^0_{ADB}	$send\_data$	$q^0_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}	get_reply_2	$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}
$q^1_{\text{TER}_1}$	q^3_{GS}	q^1_{ADB}	$authorize$	$q^1_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}
$q^1_{\text{TER}_1}$	q^3_{GS}	q^1_{ADB}	get_reply_1	$q^0_{\text{TER}_1}$	q^0_{GS}	q^1_{ADB}
$q^0_{\text{TER}_1}$	q^0_{GS}	q^1_{ADB}	$authorize$	$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^0_{GS}	q^1_{ADB}	$send\_req_1$	$q^1_{\text{TER}_1}$	q^1_{GS}	q^1_{ADB}
$q^1_{\text{TER}_1}$	q^1_{GS}	q^1_{ADB}	$authorize$	$q^1_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^3_{GS}	q^1_{ADB}	$authorize$	$q^0_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}

Table 6.4: Relation $\mathsf{r}_S(\mathsf{R}_C)$ associated with S (Example 3.2).

GS^f	ADB^f	DB^f	Interaction	GS^t	ADB^t	DB^t
q^0_{GS}	q^0_{ADB}	q^0_{DB}	$send\_req_1$	q^1_{GS}	q^0_{ADB}	q^0_{DB}
q^0_{GS}	q^0_{ADB}	q^0_{DB}	$send\_req_2$	q^1_{GS}	q^0_{ADB}	q^0_{DB}
q^1_{GS}	q^0_{ADB}	q^0_{DB}	ask_auth	q^2_{GS}	q^1_{ADB}	q^0_{DB}
q^2_{GS}	q^1_{ADB}	q^0_{DB}	$authorize$	q^2_{GS}	q^0_{ADB}	q^1_{DB}
q^2_{GS}	q^0_{ADB}	q^1_{DB}	$send\_data$	q^3_{GS}	q^0_{ADB}	q^0_{DB}
q^3_{GS}	q^0_{ADB}	q^0_{DB}	get_reply_1	q^0_{GS}	q^0_{ADB}	q^0_{DB}
q^3_{GS}	q^0_{ADB}	q^0_{DB}	get_reply_2	q^0_{GS}	q^0_{ADB}	q^0_{DB}

Table 6.5: Relation $\mathsf{r}_R(\mathsf{R}_D)$ associated with R (Example 3.3).

TER_1^f	GS^f	ADB^f	Interaction	TER_1^t	GS^t	ADB^t
$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}	$send\_req_1$	$q^1_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}	$send\_req_2$	$q^0_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}
$q^1_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}	ask_auth	$q^1_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}
$q^1_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}	$authorize$	$q^1_{\text{TER}_1}$	q^2_{GS}	q^0_{ADB}
$q^1_{\text{TER}_1}$	q^2_{GS}	q^0_{ADB}	$send\_data$	$q^1_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}
$q^1_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}	get_reply_1	$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^1_{GS}	q^0_{ADB}	ask_auth	$q^0_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}
$q^0_{\text{TER}_1}$	q^2_{GS}	q^1_{ADB}	$authorize$	$q^0_{\text{TER}_1}$	q^2_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^2_{GS}	q^0_{ADB}	$send\_data$	$q^0_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}
$q^0_{\text{TER}_1}$	q^3_{GS}	q^0_{ADB}	get_reply_2	$q^0_{\text{TER}_1}$	q^0_{GS}	q^0_{ADB}

Table 6.6: Relation $\mathsf{r}_S \ltimes \mathsf{r}_R$.

Theorem 6.1:
Let Sys $= (\text{IM}, \{T_i\}_{i\in K})$ be an interaction system with interaction model $\text{IM} = (K, \{A_i\}_{i\in K}, Int)$ and $C, D \subseteq K$ $(C, D \neq \emptyset)$ subsets of components. Let $S = (Q_C, Int, \rightarrow_S, q^0_C)$ and $R = (Q_D, Int, \rightarrow_R, q^0_D)$ be transition systems with respect to C respectively D. Let $\mathsf{r}_S(\mathsf{R}_C)$ and $\mathsf{r}_R(\mathsf{R}_D)$ be the associated relations with S and R on the relation schemata R_C and R_D. Further, let r'_S and r'_R be r_S respectively r_R restricted to reachable tuples. Then

$$\mathsf{r}'_S \ltimes \mathsf{r}'_R = \mathsf{r}$$

such that r is the relation (on the relation scheme R_C) associated with the transition system $S' = EM(S, R)$.

Proof. The proof can be found in Appendix A on Page 203. □

Aside from showing in Theorem 6.1 an interesting connection between our refinement approach and the field of relational algebra we use this result in the following to show that the Edge-Match fixed-point of a family of abstract over-approximations is legitimate if the domain on which the family is based exhibits a certain structure.

6.4 A Preciseness Condition

In the following we describe how results on acyclic database schemata [Yan81] can be interpreted in our setting. If a join is executed on a set of tables then in general not all tuples in the relations actually "appear" in the result, i.e., the join yields the same result if certain tuples are removed from the relations. These, so-called "dangling" tuples are an unnecessary factor in data transmission if the relations are stored on multiple sites (see [Mai83]). A technique to reduce the number of dangling tuples in a pre-processing is the semijoin reduction, where the semijoin operator is applied pairwise on involved relations in order to remove dangling tuples. If the attributes of the tables that are involved in a semijoin reduction have a certain structure then one can make a preciseness assumption about the result of the semijoin reduction. In the following we exploit this preciseness assumption in order to make a preciseness assumption about Edge-Match fixed-points by using the result from the last section.

First we need to introduce additional definitions from the field of relational database theory.

Definition 6.4:
Let U be a set of attributes. A **relational database scheme** $\mathcal{R}$ over U is a family $\mathcal{R} = \{\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_p\}$ of relation schemata with $\mathsf{R}_i \subseteq \mathsf{U}$ and $\mathsf{R}_i \neq \emptyset$ for $i = 1, \ldots, p$ and $\cup_{i=1,\ldots,p} \mathsf{R}_i = \mathsf{U}$. A **relational database** $\mathfrak{d}$ on the relational database scheme $\mathcal{R}$ is a set of relations $\mathfrak{d} = \{\mathsf{r}_1(\mathsf{R}_1), \ldots, \mathsf{r}_p(\mathsf{R}_p)\}$.

As the join operator is associative, $\bowtie(\mathfrak{d})$ denotes the join over all relations in $\mathfrak{d}$, i.e.,

$$\bowtie(\mathfrak{d}) = \mathsf{r}_1 \bowtie \mathsf{r}_2 \bowtie \ldots \bowtie \mathsf{r}_p.$$

The **full reduction** of a relation $\mathsf{r}(\mathsf{R}) \in \mathfrak{d}$ relative to $\mathfrak{d}$ is $FR(\mathsf{r}, \mathfrak{d}) = \pi_{\mathsf{R}}(\bowtie(\mathfrak{d}))$. The relation $FR(\mathsf{r}, \mathfrak{d})$ is the part of r that is actually used in the join $\bowtie(\mathfrak{d})$, i.e.,

$$\bowtie(\mathfrak{d}) = \mathsf{r}_1 \bowtie \mathsf{r}_2 \bowtie \ldots \bowtie \mathsf{r}_p = FR(\mathsf{r}_1, \mathfrak{d}) \bowtie FR(\mathsf{r}_2, \mathfrak{d}) \bowtie \ldots \bowtie FR(\mathsf{r}_p, \mathfrak{d}).$$

A **semijoin program** SP for $\mathfrak{d}$ is a sequence of assignments of the form

$$\mathsf{r}_i \leftarrow \mathsf{r}_i \ltimes \mathsf{r}_j$$

for $i, j \in \{1, \ldots, p\}$. $SP(\mathsf{r}_i, \mathfrak{d})$ denotes the final value of relation r_i after the execution of SP on $\mathfrak{d}$. A semijoin program SP for $\mathfrak{d}$ is called **full-reducer** for $\mathcal{R}$ if (independent from the relations in $\mathfrak{d}$) for all $1 \leq i \leq p$

$$FR(\mathsf{r}_i, \mathfrak{d}) = SP(\mathsf{r}_i, \mathfrak{d}).$$

An important theorem in the field of relational database theory states that there exists a full-reducer for a database scheme $\mathcal{R}$ if and only if $\mathcal{R}$ is acyclic [BFMY83] (a proof can be found in [Mai83] as well). The next definition specifies in which case a database scheme is called acyclic.

Definition 6.5:
A **hypergraph** is a tuple $H = (V, E)$ with a set of nodes V and a set of **hyperedges** $E \subseteq 2^V \setminus \{\emptyset\}$. The **GYO-reduction** (named after Graham [Gra79], Yu and Ozsoyoglu [YO79]) of a hypergraph $H = (V, E)$ is the process of repeatedly removing nodes from H which appear in at most one hyperedge and removing all hyperedges that are included in other hyperedges. A hypergraph $H = (V, E)$ is called **acyclic** if and only if the hypergraph $H' = (V', E')$ that results from the GYO-reduction has no nodes and no hyperedges, i.e., $V' = \emptyset$ and $E' = \emptyset$. Note that the result is independent of the sequence of node and hyperedge removals.

Let $\mathcal{R}$ be a database scheme over the set of attributes U. The hypergraph $H = (\mathsf{U}, \mathcal{R})$ is called the associated hypergraph with $\mathcal{R}$. The database scheme $\mathcal{R}$ is called **acyclic** if and only if H is acyclic.

Let $\mathcal{R}$ be an acyclic database scheme. From the sequence of removed hyperedges during the GYO-reduction one can actually construct a full-reducer that only needs a number of semijoin operations that is linear in $|\mathcal{R}|$. It is easy to see that if a database $\mathfrak{d}$ on an acyclic database scheme is a fixed-point with respect to a pairwise application of the semijoin operation then a

full-reducer applied on $\mathfrak{d}$ does not change any relation in $\mathfrak{d}$, i.e., each relation in $\mathfrak{d}$ is a full reduction.

We can now formulate our proposition regarding the "preciseness" of a fixed-point of a family of abstract over-approximations with respect to an application of the Edge-Match operator. Let $\mathbb{C} \subseteq 2^K \setminus \{\emptyset\}$ be a domain and $\{R_C\}_{C \in \mathbb{C}}$ a family of abstract over-approximations of an interaction system Sys with components K. As addressed in Chapter 3, we cannot expect that there is no artifact in any abstract over-approximation in the Edge-Match fixed-point of $\{R_C\}_{C \in \mathbb{C}}$. The following corollary states that we can conclude that the Edge-Match fixed-point of $\{R_C\}_{C \in \mathbb{C}}$ is legitimate if the hypergraph $H_{\mathbb{C}} = (K, \mathbb{C})$ is acyclic.

Corollary 6.1:
Let Sys $= (\text{IM}, \{T_i\}_{i \in K})$ be an interaction system with interaction model IM $= (K, \{A_i\}_{i \in K}, Int)$. Let $\mathbb{C} \subseteq 2^K \setminus \{\emptyset\}$ be a domain such that the hypergraph $H_{\mathbb{C}} = (K, \mathbb{C})$ is acyclic. Let $\{R_C\}_{C \in \mathbb{C}}$ be a family of abstract over-approximations, then the Edge-Match fixed-point of $\{R_C\}_{C \in \mathbb{C}}$ is legitimate.

Proof. The proof can be found in Appendix A on Page 204. □

The following example illustrates a claim that is stated in the proof of Corollary 6.1. This is, let Sys be an interaction system with the set of components K and $\mathbb{C} \subseteq 2^K \setminus \{\emptyset\}$ a domain then a relational database scheme that consists of the relational schemata associated with the subsets in $\mathbb{C}$ is acyclic if the hypergraph $H_{\mathbb{C}} = (K, \mathbb{C})$ is acyclic.

Example 6.5:
Consider the interaction system Sys from Example 1.2. Let $\mathbb{C} = \{C_1, C_2, C_3\}$ be a domain with

C_1={TER$_1$, GS, ADB},
C_2={TER$_2$, GS, DB} and
C_3={GS, ADB, DB}.

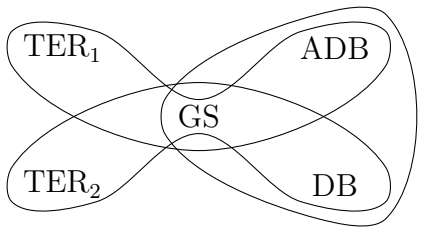


Figure 6.1: The hypergraph $H_{\mathbb{C}} = (K, \mathbb{C})$.

The hypergraph $H_{\mathbb{C}} = (K, \mathbb{C})$ is depicted in Figure 6.1. A hyperedge is depicted as a closed curve that includes all nodes in the hyperedge. It is easy to see that $H_{\mathbb{C}}$ is acyclic. The nodes TER_1 and TER_2 each only occur in one hyperedge and thus can be removed. After that two hyperedges can be removed as they are included in another hyperedge. Now the last three nodes and the resulting empty hyperedge can be removed. This is, $H_{\mathbb{C}}$ is acyclic and thus, according to Corollary 6.1, the Edge-Match fixed-point of any family of abstract over-approximations based on $\mathbb{C}$ is legitimate.

The relational schemata associated with C_1, C_2 and C_3 are

$$\mathsf{R}_{C_1}=\{\mathrm{TER}_1^f, \mathrm{GS}^f, \mathrm{ADB}^f, \mathrm{Interaction}, \mathrm{TER}_1^t, \mathrm{GS}^t, \mathrm{ADB}^t\},$$
$$\mathsf{R}_{C_2}=\{\mathrm{TER}_2^f, \mathrm{GS}^f, \mathrm{DB}^f, \mathrm{Interaction}, \mathrm{TER}_2^t, \mathrm{GS}^t, \mathrm{DB}^t\} \text{ and}$$
$$\mathsf{R}_{C_3}=\{\mathrm{GS}^f, \mathrm{ADB}^f, \mathrm{DB}^f, \mathrm{Interaction}, \mathrm{GS}^t, \mathrm{ADB}^t, \mathrm{DB}^t\}.$$

Let $\mathcal{R} = \{\mathsf{R}_{C_1}, \mathsf{R}_{C_2}, \mathsf{R}_{C_3}\}$ be a relational database scheme over the set of attributes $\mathsf{U} = \mathsf{R}_{C_1} \cup \mathsf{R}_{C_2} \cup \mathsf{R}_{C_3}$. Just as above it is easy to see that the hypergraph $H_{\mathcal{R}} = (\mathsf{U}, \mathcal{R})$ (depicted in Figure 6.2) is acyclic. This is, $\mathcal{R}$ is an acyclic database scheme.

6.5 Conclusion

We introduced a connection between the field of relational algebra and our refinement approach of abstract over-approximations by the Edge-Match operator. For this purpose we modeled abstract over-approximations as relations and the Edge-Match operator by the semijoin operator. Aside from

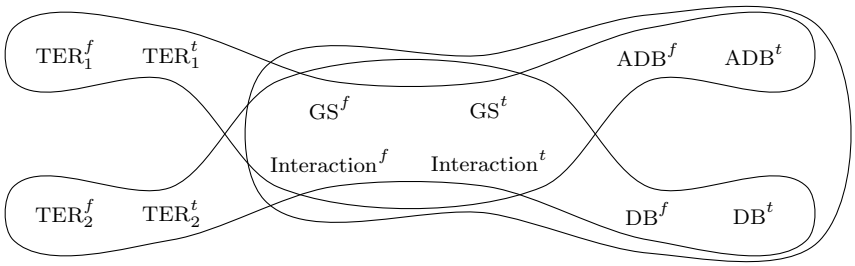


Figure 6.2: The hypergraph $H' = (\mathsf{U}, \mathcal{R})$.

pointing out this interesting connection we used results from the field of relational database theory to show that our preciseness property is guaranteed for a certain class of families of abstract over-approximations. This class is defined by a structural property on the domains on which the families of abstract over-approximations are based, i.e., the hypergraph that is induced by these domains is acyclic. This means particularly that this subclass is independent from the behavior of the components.

Chapter 7

Conclusion

We dealt in this work with various aspects of properties in cooperating systems. We used the formalism of interaction systems [GS03] to model cooperating systems.

In a first part (Chapter 2) we discussed complexity issues of the reachability problem in subclasses of interaction systems which are defined by architectural constraints on a graph structure that represents the communication among subsystems. We considered system classes with a tree-like, star-like and linear communication pattern. These subclasses include systems that are highly relevant in practice, e.g., client server systems exhibit communication patterns that form stars or trees. Deciding reachability of global states in interaction systems is PSPACE-complete [MCM08c]. We showed that deciding this problem remains PSPACE-complete in all of our subclasses. Furthermore, we argued that deciding progress in our subclasses and that deciding local reachability, i.e., the question whether or not a fixed local state occurs in a reachable global state, in tree-like systems is PSPACE-complete as well. Our result motivates further research on efficient techniques that are based on sufficient conditions and decide reachability based system properties as deadlock-freedom in cooperating systems with architectural constraints and justifies published results which introduce such tech-

niques for establishing deadlock-freedom [Hoa85, BR91, BCD02, MCM08a, Lam09, HJK10].

In a second part, which accounts for Chapter 3-6 we introduced an approach to efficiently represent over-approximations by abstract over-approximations of the global behavior of interaction systems, refine these abstract over-approximations based on the Edge-Match operator and present an approach based on abstract over-approximations for establishing deadlock-freedom in interaction systems. We presented results of our approaches for various parameterized examples that are calculated by a tool developed by us. In addition, we illustrated an interesting link between our approach to refine abstract over-approximations and the semi-join operator from the field of relational database theory and used this link in order to show a result regarding the preciseness of the fixed-point of a family of abstract over-approximations with respect to an application of the Edge-Match operator.

Our refinement approach extends the Cross-Checking approach introduced in [MMC09b] that deals with compact representations of the reachable global state space. Our approach enhances the Cross-Checking approach by dealing with over-approximations of the reachable behavior, i.e., including transitions, instead by dealing with over-approximations of the reachable state space. Our results show that a refinement based on our Edge-Match operator can result in significantly less reachable states in abstract over-approximations than a refinement by the Cross-Checking operator (applied to our context). We proposed a procedure that calculates the fixed-point of a family of abstract over-approximations with respect to an application of the Edge-Match operator. Let Sys = (IM, $\{T_i\}_{i \in K}$) be an interaction system with interaction model IM = (K, $\{A_i\}_{i \in \mathrm{K}}$, Int) and $\mathbb{C}$ a domain of IM. We argued that the costs of the application of the Edge-Match operator on a pair of abstract over-approximations are bounded by

$$em = 2\left(m^{c_{\max}} + m^{2 \cdot c_{\max}} \cdot |Int|\right) + \left(m^{2 \cdot c_{\max}} \cdot |Int|\right)^2,$$

where m is the size of the largest local state space of the components in

K and $c_{\max}$ is the size of the larges set in $\mathbb{C}$. The costs of our procedure, applied on Sys, based on the domain $\mathbb{C}$, are bounded by

$$|\mathbb{C}| \cdot m^{2 \cdot c_{\max}} \cdot |Int| \cdot |\mathbb{C}|^2 \cdot em.$$

This is, our procedure runs in polynomial time if $c_{\max}$ is a constant, i.e., the number of components in each set in $\mathbb{C}$ is bounded by a constant and $|\mathbb{C}|$ is of polynomial size. Similar as in [MMC09b] we proposed to use a domain that consists of all subsets of components with a constant size d where the interaction graph restricted to a subset is connected – a domain like this guarantees that our procedure runs in polynomial time.

We introduced an approach that is based on a sufficient condition to establish deadlock-freedom in interaction systems in polynomial time. Our approach works on arbitrary systems, i.e., our approach does not depend on characteristics of subclasses as, e.g., architectural constraints. We argued that our approach is incomparable with the waiting chain approach introduced in [Min10], i.e., if our approach fails to ensure deadlock-freedom then the waiting chain approach might succeeds and vice versa. In Chapter 5 our results show that our approach succeeds to, e.g., establish deadlock-freedom for Tanenbaum's solution of the Dining Philosophers Problem, where the waiting chain approach fails (see [Min10]). Our abstract over-approximations are compact representations of over-approximations of the reachable behavior of an interaction system, i.e., abstract over-approximations provide further potential to be the basis of approaches to establish other safety properties in interaction systems.

Summarizing the above we introduced an approach to establish deadlock-freedom, in polynomial time, based on an analysis of abstract over-approximations of the global behavior of an interaction system, that can be refined by the Edge-Match operator. Our approach can be applied to arbitrary interaction systems and does not require any architectural restrictions or other constraints whatsoever. Our approach can be easily applied to cooperating systems modeled by other formalisms. This can be done either by adapting

our approach or by using a mapping among formalisms – see, e.g., [Min10] for a mapping between interaction systems and 1-safe Petri nets. If we fail to establish deadlock-freedom then the output of our approach includes information on where the system in consideration can be modified such that our approach succeeds.

Our approach provides potential for further research and development. The following points itemize some suggestions for future research.

- We did not consider to use our approach in combination with well-known state space reduction techniques.

- Here we only treated the safety property of deadlock-freedom. Our abstract over-approximations are compact representations of over-approximations of the reachable global behavior of an interaction system. Thus, it appears promising to research further conditions that can be applied on abstract over-approximations to the purpose of establishing general safety in cooperating systems.

- Our procedure for calculating the Edge-Match fixed-point of a family of abstract over-approximations (Algorithm 2 in Chapter 3) is based on applications of the Edge-Match operator on all reasonable pairs of abstract over-approximations. A lot of these applications are independent from each other. Thus, our procedure is highly parallelizable because a great deal of applications of the Edge-Match operator can be done in parallel.

- Right now, in order to calculate the Edge-Match fixed-point of a family of abstract over-approximations, we apply the Edge-Match operator on all reasonable pairs of abstract over-approximations until there is no further refinement. This approach might be improved by investigating a sophisticated order of applications in order to minimize the number of applications of the Edge-Match operator. A starting point is, for example, the introduced link between our approach and the field of

relational algebra in Chapter 6. From the hypergraph structure of an acyclic database scheme one can derive an order of semijoin operations that is a full-reducer and consists of a number of semijoin operations that is linear in the number of relational schemata in the database.

Appendix A

Proofs

A.1 Proofs from Chapter 2

Proof of Theorem 2.1

The idea of the proof of Theorem 2.1 is as follows. In Section 2.3 we introduced the straightforward, recursive Algorithm 1 called *eval* to determine whether or not a QBF P (given over the restricted grammar) is in $TQBF$. Note that the algorithm is deterministic. We show that the evaluation of the algorithm corresponds to a path in the global behavior of Sys_H and illustrate that this path ends in q^t if and only if H is true. For this purpose we map certain events in the algorithm *eval* to interactions in Sys_H and show that there is only one path (up to some interleaving) in the behavior of Sys_H that corresponds to the evaluation of *eval* with respect to this mapping.

Before we prove Theorem 2.1, we need some preliminaries.

Recursive Algorithm

In Algorithm 1 (Section 2.3) we assume that in Line 8 and 2 the conjunction respectively the disjunction is called in sequence from left to right. In addition, we assume, that $eval(P'')$ is not called in Line 8 if $eval(P')$ is evaluated $false$ and $eval(P'_{x=false})$ is not called in Line 2 if $eval(P'_{x=true})$ is evaluated $true$. These assumptions imply a deterministic execution of $eval(H)$ for a QBF H.

The execution of $eval(H)$ for a QBF H can be described by a sequence over the following words.

- "$call\_eval(P)$": subformula P is called by $eval$
- "$eval(P) = true$": subformula P was evaluated $true$ by $eval$
- "$eval(P) = false$": subformula P was evaluated $false$ by $eval$

For a QBF H let Seq_H be this sequence and $Seq_H(i)$ the ith word in Seq_H for $i = 1, \ldots, length(Seq_H)$, where $length(Seq_H)$ is the number of words in Seq_H. It is clear, that $H \in TQBF$ if and only if the last entry of Seq_H is "$eval(H) = true$" and "$eval(H) = false$" otherwise.

Example A.1:
Consider the QBF $H = \neg\exists x_1.(x_1 \wedge \neg x_1)$ with its subformulas abbreviated as in Figure 2.5, then Seq_H is given by

- $call\_eval(H)$
- $call\_eval(P_1)$
- $call\_eval(P_2)$ (true is assigned to x_1)
- $call\_eval(P_3)$
- $eval(P_3) = true$
- $call\_eval(P_4)$
- $call\_eval(P_5)$
- $eval(P_5) = true$

- $eval(P_4) = false$
- $eval(P_2) = false$
- $call\_eval(P_2)$ (false is assigned to x_1)
- $call\_eval(P_3)$
- $eval(P_3) = false$
- $eval(P_2) = false$
- $eval(P_1) = false$
- $eval(H) = true$

Mapping the words of Seq_H to Int

Let $H \in QBF$ and Sys_H be the associated tree-like interaction system. We treat the associated interaction graph G_H as a rooted tree with component H' as the root. In these terms, if we speak of a successor, a predecessor or a subtree spanned by a component, we refer to components with respect to G_H. Let $\text{Int}' \subseteq \text{Int}$ be the subset of interactions given by:

$$\{eval\_P \to P_k, P\_ask\_P_k\_true, P\_ask\_P_k\_false | P \in \text{K}_1 \cup \{H'\} \text{ with succ. } P_k\}$$

and S the set of words that can occur in Seq_H, given by

$$\{\text{“}call\_eval(P)\text{”}, \text{“}eval(P) = true\text{”}, \text{“}eval(P) = false\text{”} | P \text{ is a subformula of } H\}.$$

We define a mapping from S to Int' by the function $f : S \to \text{Int}'$ with

- $f(\text{“}call\_eval(P)\text{”}) = eval\_P' \to P$,
- $f(\text{“}eval(P) = true\text{”}) = P'\_ask\_P\_true$ and
- $f(\text{“}eval(P) = false\text{”}) = P'\_ask\_P\_false$.

where P' is the predecessor of P and $P' = H'$ if $P = H$.

Lemma A.1:
Let $\tilde{\sigma}$ be a sequence of interactions that corresponds to the interactions on a path in the global behavior of Sys_H, such that $\tilde{\sigma}$ is infinite or the corresponding path ends in a state where no transition is possible. Let σ be

the sequence obtained by removing the interactions in $\mathrm{Int} \setminus \mathrm{Int}'$ and let $\sigma(i)$ be the ith interaction in σ for $i = 1, \ldots, length(\sigma)$, where $length(\sigma)$ is the length of σ. Then $length(\sigma) = length(Seq_H)$ and

$$\bigwedge_{i=1,\ldots,length(\sigma)} f(Seq_H(i)) = \sigma(i).$$

Before we prove Lemma A.1 by induction, we need some observations which follow from invariants of algorithm *eval*. In the following we refer repeatedly to the structure of the transition systems given in Figure 2.6, 2.7 and 2.8 and the interactions given on page 47. We assume the induction hypothesis to be true, i.e., for some $i < length(\sigma)$ holds that for all $1 \leq k \leq i$ $f(Seq_H(k)) = \sigma(k)$.

Observation A.1:
Consider $\sigma(i)$ to be performed and let $Seq_H(i+1) =$ "$eval(P) = true$" where P' is the predecessor of P, then P' waits to perform $P'\_ask\_P\_true$.

The same applies for $Seq_H(i+1) =$ "$eval(P) = false$" where P' waits to perform $P'\_ask\_P\_false$.

Proof. There is $1 \leq j \leq i$ with $Seq_H(j) =$ "$call\_eval(P)$", i.e., if subformula P is evaluated to true then it is assured that P was called previously. Let j be maximal for this property. For $j+1 \leq k \leq i$ $Seq_H(k) \notin \{$"$call\_eval(P)$", "$eval(P) = true$", "$eval(P) = false$"$\}$, i.e., P is not involved in between. It follows that $\sigma(j) = f($"$call\_eval(P)$"$) = eval\_P' \rightarrow P$, this is, P' reached a state in which it waits for $P'\_ask\_P\_true$ or $P'\_ask\_P\_false$. Since these interactions were not performed for $j+1 \leq k \leq i$, it is assured, that after $\sigma(i)$ has been performed, P' still waits to perform $P'\_ask\_P\_true$ or $P'\_ask\_P\_false$. □

Observation A.2:
Consider $\sigma(i)$ to be performed and let $Seq_H(i+1) =$ "$call\_eval(P)$". Let component P' be the predecessor of component P, then P eventually reaches a state in which it waits to perform $eval\_P' \rightarrow P = f(Seq_H(i+1))$.

Proof. There are six cases for P:

i) P waits to perform $eval\_P' \to P$, i.e., P is in a state labeled t or f, then $eval\_P' \to P$ is enabled.

ii) P waits to perform $set\_\tilde{P}\_true\_P \to \tilde{P}$ for a successor $\tilde{P}$ of P. It is easy to see that this is only possible if $f(\text{“}eval(H) = true\text{”}) = H'\_ask\_H\_true$ performed which is not the case.

iii) P waits to perform $eval\_P \to \tilde{P}$. If this situation occurs, either $eval\_P' \to P$, $P\_ask\_\tilde{P}\_false$ (if $P = \exists.x_l\tilde{P}$) or $P\_ask\_\bar{P}'\_false$ (if $P = \bar{P} \wedge \tilde{P}$) was performed. Let this be the case for $f(Seq_H(j))$ with $j < i$ (j maximal). Due to the structure of Seq_H, $f(Seq_H(j+1)) = eval\_P \to \tilde{P}$ would be the next interaction to be performed, i.e., P cannot stay in a state waiting to perform $eval\_P \to \tilde{P}$.

iv) P waits to perform $P\_ask\_\tilde{P}\_true$. Analogously to case 3, this cannot happen.

v) $P = \exists.x_l\tilde{P}$ waits to perform $set\_x_l'\_true/false$ or $ask\_true/false_{x_l'}$ for $1 \leq l \leq n$. This interaction would always be performed, and subsequently P waits to perform $eval\_P \to \tilde{P}$, which is not possible due to case 3.

vi) P waits to perform $set\_x_l\_true\_P \to \tilde{P}$ or $set\_x_l\_false\_P \to \tilde{P}$ for $1 \leq l \leq n$. If $\tilde{P}$ models a variable then $set\_x_l\_true\_P \to \tilde{P}$ respectively $set\_x_l\_false\_P \to \tilde{P}$ is enabled by $\tilde{P}$ and can perform. After this, $f(Seq_H(i+1)) = eval\_P' \to P$ becomes enabled by P. If $\tilde{P}$ does not model a variable then analogously (to case 1-5) either $\tilde{P}$ enables $set\_x_l\_true\_P \to \tilde{P}$ respectively $set\_x_l\_false\_P \to \tilde{P}$ or $\tilde{P}$ waits to perform $set\_x_r\_true\_\tilde{P} \to \bar{P}$ respectively $set\_x_r\_false\_\tilde{P} \to \bar{P}$ ($r \in \{1, \ldots, n\}$) for a successor $\bar{P}$ of $\tilde{P}$. By induction follows, that this interaction performs eventually. Therefore, $f(Seq_H(i+1)) = eval\_P' \to P$ eventually becomes enabled as well.

□

Observation A.3:
Let $Seq_H(i+1) =$ "$eval(P) = true$" such that $P = x_l^r$ for $1 \leq l \leq n$ and $r \in \{1, \ldots, k_l\}$, then it is assured that P waits to perform $P'\_ask\_P\_true$ after $\sigma(i)$ is performed. The same applies for $Seq_H(i+1) =$ "$eval(P) = false$" with $\sigma(i+1) = P'\_ask\_P\_false$.

Proof. Let $Seq_H(i+1) =$ "$eval(P) = true$" (respectively $Seq_H(i+1) =$ "$eval(P) = false$"), then $Seq_H(i) =$ "$call\_eval(P)$" and there is $Q' = \exists x_l.Q$ and $j < i$ such that $Seq_H(j) =$ "$call\_eval(Q)$", i.e., if algorithm *eval* calls a variable recursively then it is assured that beforehand a subformula was called that quantifies this variable. Let j be maximal for this property. There are two cases for $j-1$:

a) $Seq_H(j-1) =$ "$call\_eval(Q')$", i.e., x_l is set to true in the subsequent call of $eval(Q)$ (see algorithm *eval*). After $\sigma(j-1) = f(Seq_H(j-1))$ was performed, either

   a.1) $set\_x_l'\_true$ or

   a.2) $ask\_true_{x_l'}$ becomes enabled.

b) $Seq_H(j-1) =$ "$eval(Q) = false$", i.e., Q was evaluated to false and is called by *eval* again with x_l set to false. After $\sigma(j-1) = f(Seq_H(j-1))$ was performed, either

   b.1) $set\_x_l'\_false$ or

   b.2) $ask\_false_{x_l'}$ becomes enabled. This is not possible, because then there is no way $\sigma(j) = call\_Q' \rightarrow Q = f(Seq_H(j))$.

Consider Case a.1) (resp. b.1)). Let, after $\sigma(j-1) = f(Seq_H(j-1))$ was performed, $set\_x_l'\_true$ (respectively $set\_x_l'\_false$) be enabled and perform. This means that Q' waits to perform $set\_x_l\_true\_Q' \rightarrow Q$ (respectively $set\_x_l\_false\_Q' \rightarrow Q$). Analogously to A.2, this interaction eventually becomes enabled. If $set\_x_l\_true\_Q' \rightarrow Q$ (respectively $set\_x_l\_false\_Q' \rightarrow Q$) is performed it is clear that $eval\_Q' \rightarrow Q = \sigma(j) = f(Seq_H(j))$ be-

comes enabled. Analogously, for each component $\tilde{Q}$ and its predecessor $\tilde{Q}'$, $set\_x_l\_true\_\tilde{Q}' \rightarrow \tilde{Q}$ (respectively $set\_x_l\_false\_\tilde{Q}' \rightarrow \tilde{Q}$) has to be performed before $eval\_\tilde{Q}' \rightarrow \tilde{Q}$ becomes enabled. This is until $\tilde{Q}$ models a variable. If $\tilde{Q}$ models an occurrence of x_l, then true (resp. false) is assigned to $\tilde{Q}$, else, there is no effect on the current state of $\tilde{Q}$. Therefore it is assured that P waits to perform $P'\_ask\_P\_true$ (respectively $P'\_ask\_P\_false$) after $\sigma(i)$ is performed.

Consider Case a.2), i.e., $ask\_true_{x_l'}$ is enabled after $\sigma(j-1)$ performed. Then the component x_l' is in the state labeled t. This means, the last interaction involving x_l' cannot be $set\_x_l'\_false$ or $ask\_false_{x_l'}$. There are three cases

1.) x_l' was never involved since $\sigma(j-1)$ is performed. Due to the fact that all components modeling occurrences of variables start in their state labeled t, it is easy to see that it is not possible that any of these components could reach the state labeled f. Therefore these components are still in the state labeled t when P' waits to perform $P'\_ask\_P\_true$.

2.) The last interaction involving x_l' was $set\_x_l'\_true$. With Case a.1) follows that all components that model occurrences of x_l were set in their respective state labeled t. As there was no interaction involving x_l' since $set\_x_l'\_true$ performed, it is assured that these components are still in the state labeled t when P' waits to perform $P'\_ask\_P\_true$.

3.) The last interaction involving x_l' was $ask\_true_{x_l'}$. This case is easily reducible to the last two cases. Therefore it is assured that all components modeling occurrences of x_l are still in the state labeled t when P' waits to perform $P'\_ask\_P\_true$.

□

Proof of Lemma A.1

In the initial state of Sys_H all components but H' are in their state labeled t. The only enabled interaction is $eval\_H' \rightarrow H$ with $f(\text{“}call\_eval(H)\text{”}) = eval\_H' \rightarrow H$. Thus, $\sigma(1) = f(Seq_H(1))$. Lemma A.1 is proven by induction, i.e., we have to show that, if $f(Seq_H(i)) = \sigma(i)$ is performed, under the assumption $f(Seq_H(j)) = \sigma(j)$ for $1 \leq j \leq i$ then the interaction in Int' that is performed next is $f(Seq_H(i{+}1))$. In fact we show that $f(Seq_H(i{+}1))$ eventually becomes enabled, such that in between only interactions in $\text{Int} \setminus \text{Int}'$ are performed.

We now consider the three possible cases for $Seq_H(i)$.

Induction - Case 1

Consider $Seq_H(i) = \text{“}eval(P) = true\text{”}$, i.e., $\sigma(i) = f(\text{“}eval(P) = true\text{”}) = P'\_ask\_P\_true$ where P' is the subformula P is included in and $P' = H'$ if $P = H$. If existent, let P'' be the predecessor of P' ($P'' = H'$ if $P' = H$). It is clear, that P is in its state labeled t. There are five cases:

1.a) If $P' = P \wedge \tilde{P}$, then $Seq_H(i+1) = \text{“}call\_eval(\tilde{P})\text{”}$. This means, that P' waits to perform $eval\_P' \rightarrow \tilde{P}$. From Observation A.2 follows the same for $\tilde{P}$ as well. It follows that the only newly enabled interaction in Sys_H is $eval\_P' \rightarrow \tilde{P} = f(Seq_H(i+1))$.

1.b) If $P' = \tilde{P} \wedge P$, then $Seq_H(i+1) = \text{“}eval(P') = true\text{”}$. The component P' waits to perform $P''\_ask\_P'\_true = f(Seq_H(i+1))$ and from Observation A.1 follows that this is the only newly enabled interaction in Sys_H.

1.c) If $P' = \neg P$, then $Seq_H(i+1) = \text{“}eval(P') = false\text{”}$. The component P' waits to perform $P''\_ask\_P'\_false = f(Seq_H(i+1))$ and from Observation A.1 follows that this is the only newly enabled interaction in Sys_H.

1.d) If $P' = \exists x_i.P$, then $Seq_H(i+1) =$ "$eval(P') = true$". The component P' waits to perform $P''\_ask\_P'\_true = f(Seq_H(i+1))$ and from Observation A.1 follows that this is the only newly enabled interaction in Sys_H.

1.e) If $P' = H'$, then $i = length(Seq_H)$, i.e., there is no next word on Seq_H and no new interaction $\in \text{Int}'$ is enabled in Sys_H.

Induction - Case 2

Consider $Seq_H(i) =$ "$eval(P) = false$", i.e., $\sigma(i) = f($"$eval(P) = false$"$) = P'\_ask\_P\_false$ where P' is the predecessor of P and $P' = H'$ if $P = H$. If existent, let P'' be the predecessor of P' ($P'' = H'$ if $P' = H$). It is clear, that P is in its state labeled t. There are five cases:

2.a) If $P' = P \wedge \tilde{P}$, then $Seq_H(i+1) =$ "$eval(P') = false$". The component P waits to perform $P''\_ask\_P'\_false = f(Seq_H(i+1))$ which is, according to Observation A.1, enabled by P''.

2.b) If $P' = \tilde{P} \wedge P$, analogously to Case 2.a).

2.c) If $P' = \neg P$, analogously to Case 2.a).

2.d) If $P' = \exists x_i.P$, then there must be $j < i$ with $Seq_H(j) =$ "$call\_eval(P)$", i.e., if P was evaluated to false then P was called by eval previously. Let j be the largest value with this property. By assumption follows that $j < i$ is the biggest index with $\sigma(j) = f($"$call\_eval(P)$"$) = eval\_P' \rightarrow P$. In line 2 of the eval algorithm P can be called by eval with x_i set to true and afterwards with x_i set to false. Accordingly, there are two cases for $Seq_H(j-1)$. Either P' was called, i.e., P is called with x_i set to true or P was evaluated to false and was called a second time with x_i set to false.

2.d.a) If $Seq_H(j-1) =$ "$call\_eval(P')$" then we have $Seq_H(i+1) =$ "$call\_eval(P)$". By the induction assumption we can conclude

that $\sigma(j-1) = eval\_P'' \rightarrow P' = f(\text{“}call\_eval(P')\text{”})$, i.e., either $set\_x_i'\_true$ or $ask\_true_{x_i'}$ was enabled after $\sigma(j-1)$ performed. This assures that the component x_i' is in its state t after $\sigma(j)$ performed. Hence there was no interaction involving component P' since $\sigma(j)$, x_i' is still in its state labeled t when $\sigma(i)$ is performed. Therefore, after $\sigma(i)$ performs, the only newly enabled interaction is $set\_x_i'\_false$, after that $set\_x_i\_false\_P' \rightarrow P$ and after that P' waits to perform $eval\_P' \rightarrow P = f(\text{“}call\_eval(P)\text{”})$ which is, by Observation A.2, assured to become enabled eventually.

2.d.b) If $Seq_H(j-1) = \text{“}eval(P) = false\text{”}$ then we have $Seq_H(i+1) = \text{“}eval(P') = false\text{”}$. By assumption follows that $\sigma(j-1) = P'\_ask\_P\_false = f(\text{“}eval(P) = false\text{”})$. By Case 1.d.a follows that x_i' is in its state labeled f when $\sigma(i)$ is performed, i.e., after $\sigma(i)$, the only newly enabled interaction is $ask\_false_{x_i'}$. When $ask\_false_{x_i'}$ is performed, it follows from Observation A.1 that the only newly enabled interaction is $P''\_ask\_P'\_false$ which equals $f(\text{“}eval(P') = false\text{”})$.

2.e) If $P' = H'$, analogously to case Case 2.a).

Induction - Case 3

Consider the case that $Seq_H(i) = \text{“}call\_eval(P)\text{”}$. This is, in this case $\sigma(i) = f(\text{“}call\_eval(P)\text{”}) = eval\_P' \rightarrow P$ where P' is the predecessor of P and $P' = H'$ if $P = H$. There are four cases

3.a) If $P = \neg\tilde{P}$, then $Seq_H(i+1) = \text{“}call\_eval(\tilde{P})\text{”}$. The component P waits to perform $eval\_P \rightarrow \tilde{P} = f(Seq_H(i+1))$ which is, enabled by $\tilde{P}$ accordingly to Observation A.2 and therefore the only newly enabled interaction.

3.b) If $P = \tilde{P}_1 \wedge \tilde{P}_2$, then $Seq_H(i+1) = \text{“}call\_eval(\tilde{P}_1)\text{”}$. The component

P waits to perform $eval\_P \rightarrow \tilde{P}_1 = f(Seq_H(i+1))$. From Observation A.2 follows that this is the only new enabled interaction.

3.c) If $P = \exists x_i.\tilde{P}$, then $Seq_H(i+1) =$ "$call\_eval(\tilde{P})$". In Sys_H the only new enabled interaction is either $set\_x_i'\_true$ or $ask\_true_{x_i'}$. If $set\_x_i'\_true$ is executed then the only newly enabled interaction is $set\_x_i\_true\_P \rightarrow \tilde{P}$. Anyway, if $set\_x_i\_true\_P \rightarrow \tilde{P}$ or $ask\_true_{x_i'}$ is executed, P waits to perform $eval\_P \rightarrow \tilde{P} = f($"$call\_eval(\tilde{P})$"$)$ which is enabled by $\tilde{P}$ due to Observation A.2.

3.d) If $P = x_l^r$, for $1 \leq l \leq n$ and $r \in \{1, \ldots, k_i\}$, then either $Seq_H(i+1) =$ "$eval(P) = true$" or $Seq_H(i+1) =$ "$eval(P) = false$". With Observation A.3 follows that P waits to perform either $f(Seq_H(i+1)) = P'\_ask\_P\_true$ or $f(Seq_H(i+1)) = P'\_ask\_P\_false$. Due to the fact that P' waits to perform this interaction as well, $f(Seq_H(i+1))$ is the only newly enabled interaction $\in \text{Int}'$.

Proof of Theorem 2.1

Proof. By Lemma A.1 we have shown that, if $H \notin TQBF$, i.e., Algorithm 1 applied on H returns false, then each path in the global behavior in our model Sys_H eventually reaches a state where only the interaction $H'\_ask\_H\_false$ is enabled. If $H'\_ask\_H\_false$ performed, then there is no way q^t can be reached.

Analogously, if $H \in TQBF$, i.e., Algorithm 1 applied on H returns true, then eventually the interaction $H'\_ask\_H\_true$ is performed. The only new enabled interaction is $set\_H\_true\_H' \rightarrow H$. If $set\_H\_true\_H' \rightarrow H$ is performed then $T_{H'}$ reaches its state labeled t. Let $P' \in \text{K}_1 \cup \{H'\}$ be a component and $P \in \text{K}_1$ its successor (i.e., P' does not model a variable) such that $set\_P\_true\_P' \rightarrow P$ is enabled. There are four cases for the structure of P and two for P' if $set\_P\_true\_P' \rightarrow P$ is performed.

- P models a variable, then it is assured that P reaches its state labeled

t and no new interaction is enabled.

- $P = \exists x_i.\tilde{P}$, then either $set\_x_i'\_true$ or $ask\_true_{x_i'}$ becomes enabled. Anyway, it is assured, that x_i' reaches its state labeled t. This is, $set\_\tilde{P}\_true\_P \rightarrow \tilde{P}$ becomes enabled.
- $P = \tilde{P} \wedge \bar{P}$ or $P = \neg\tilde{P}$, then $set\_\tilde{P}\_true\_P \rightarrow \tilde{P}$ becomes enabled.
- $P' = P \wedge \bar{P}$, then $set\_\bar{P}\_true\_P' \rightarrow \bar{P}$ becomes enabled.
- In any other case, P' reaches its state labeled t.

This is, eventually all components reach their state labeled t. It follows that $H \in TQBF \Leftrightarrow (\mathrm{Sys}_H, q^t) \in TRIST$. □

Proof of Corollary 2.1

Proof. The proof of this corollary follows from the proof of Theorem 2.1. Let H be a QBF, Sys_H the associated interaction system that is constructed as in Section 2.3 and q^t the global state in the global behavior T of Sys_H in which all components are in their state labeled t. Theorem 2.1 states that H is true if and only if q^t is reachable in T. Actually, in the proof of Theorem 2.1, we used that a global state is reachable where the local behavior of component H' is in the local state $t_{H'}$ if and only if H is true. We showed that, from such a global state, there is always the possibility to reach the designated global state q^t by propagating, starting from H', down through the interaction graph (that forms a tree) that each component shall reach its local state labeled t. This is, we showed in the proof of Theorem 2.1 that component H' reaches the state $t_{H'}$ if and only if H is true. □

Proof of Theorem 2.2

Proof. We prove this theorem by giving an isomorphism, with respect to transitions in Sys_M and transitions of configurations in M, between global

states of Sys_M and configurations of M. The statement of the theorem then follows by induction as the isomorphism maps the initial configuration of M to the initial state of Sys_M.

Let R be the set of configurations of M. We map $(p; \gamma_0, \ldots, \underline{\gamma_i}, \ldots, \gamma_{n+1}) \in R$ to a global state $q = (q_0, \ldots, q_{n+1})$ such that $q_i = (p, \gamma_i)$ and $q_j = (s, \gamma_j)$ for $j \neq i$. Let Q' be the set of global states that correspond to the configurations in R. It is clear that this mapping is a bijection between R and Q'.

Let $(p; \gamma_0, \ldots, \underline{\gamma_i}, \ldots, \gamma_{n+1}) \in R$ and $q = (q_0, \ldots, q_{n+1}) \in Q'$ be the associated state in Sys_M. Let $\delta(p, \gamma_i) = (p', \gamma_i', \mathbb{T})$, i.e., the next configuration in M is $(p'; \gamma_0, \ldots, \gamma_i', \underline{\gamma_{i+1}}, \ldots, \gamma_{n+1}) \in R$ if $\mathbb{T} = 1$ (the case $\mathbb{T} = -1$ is treated analogously). The only enabled port in component i is $(p, \gamma_i)_i^1$, then the only enabled interaction in q is $\{(p, \gamma_i)_i^1, (p, \gamma_i)_{i+\mathbb{T}}^2\}$. Thus, component i reaches the state (s, γ_i') and component $i + \mathbb{T}$ the state $(p', \gamma_{i+\mathbb{T}})$. The resulting global state q' corresponds to the respective configuration in M. The fact that the inverse of the mapping is also a homomorphism can be shown analogously. □

Proof of Theorem 2.3

Proof. Let q be a state in the global behavior T of Sys and q' be the state in the global behavior T' of Sys′ where $q_i = q_i'$ for $i \in K$ and $q_{cc}' = q_{cc}^0$, i.e., component cc is in its initial state. Let $\mathrm{Int} = \{\alpha^1, \alpha^2, \ldots, \alpha^k\}$ and $\alpha^j = \{a_{j_1}, \ldots, a_{j_{|\alpha^j|}}\}$ such that each port in α^j is enabled in q, i.e., all local states q_l', $l = j_1, \ldots, j_{|\alpha^j|}$ in q' enable the ports a_l^{ok} and a_l and do not enable $a_l^{\neg ok}$. The state q' enables the interaction $\{\alpha_{cc}^j\}$. If this interaction is performed then the only possible sequence of interactions results in a state $\tilde{q}'$ with $\tilde{q}_i = \tilde{q}_i'$ for $i \in K$ and $\tilde{q}_{cc}' = q_{cc}^0$. Let there be a port in α^j that is not enabled in q, e.g., q_l with $l \in \{j_1, \ldots, j_{|\alpha^j|}\}$ does not enable a_l then q_l' does enable $a_l^{\neg ok}$ and not a_l^{ok}. If $\{\alpha_{cc}^j\}$ performed in q' then the only possible sequence of interactions in Sys′ leads back to state q'. For the global initial

states q^0 of Sys and $q^{0\prime}$ of Sys$'$ holds that $q_i^0 = q_i^{0\prime}$ for $i \in K$ and $q_{cc}^{0\,\prime}$ is the initial state of the local behavior of component *cc*. The "if" part follows by induction over paths in the global behavior of Sys. The "and only if" part follows analogously. □

Proof of Theorem 2.4

Proof. We already argued that the progress problem in tree-like interaction systems is in PSPACE. Let H be a QBF and Sys'_H the associated interaction system which is constructed as described above. As argued in Section 2.3, Sys'_H is tree-like and of polynomial size. It is easy to see that the interaction *evaluated_true* is the only interaction that is enabled if the state q^t is reached. In this case, *evaluated_true* performs infinitely often in every run of the modified system, i.e., the component *pro* participates infinitely often. Therefore the component *pro* may progress if H is true. If H is false then eventually $end\_false_{H'}$ becomes enabled in component H' and the interaction *evaluated_true* never becomes enabled, i.e., *pro* may not progress. Thus, the component *pro* may progress if and only if H is true, i.e., the progress problem in tree-like interaction systems is PSPACE-hard. □

A.2 Proofs from Chapter 3

Proof of Lemma 3.1

Proof. Let R be an abstract over-approximation of T. Assume that $E(R)$ is not an over-approximation of the global behavior T of Sys, i.e., there is a transition $q \xrightarrow{\alpha}_T q'$ reachable in T such that $q \xrightarrow{\alpha}_{E(R)} q'$ is not a reachable transition in $E(R)$. This is, either $q \xrightarrow{\alpha}_{E(R)} q'$ is included in $E(R)$ but not reachable in $E(R)$ or $q \xrightarrow{\alpha}_{E(R)} q'$ is not even included in the transition relation of $E(R)$. We distinguish these two cases.

1. If $q \xrightarrow{\alpha}_{E(R)} q'$ is not a transition in $E(R)$ then, according to the preliminaries of Lemma 3.1, there is no transition $q_C \xrightarrow{\alpha}_R q'_C$ in R with $q{\downarrow_C} = q_C$ and $q'{\downarrow_C} = q'_C$. This is a contradiction as $q \xrightarrow{\alpha}_T q'$ is reachable in T and thus $q{\downarrow_C} \xrightarrow{\alpha}_R q'{\downarrow_C}$ is reachable in the abstract over-approximation R.

2. Let $q \xrightarrow{\alpha}_{E(R)} q'$ be a transition in $E(R)$. This is, $q \xrightarrow{\alpha}_{E(R)} q'$ is not reachable in $E(R)$. $q \xrightarrow{\alpha}_T q'$ is reachable in T, i.e., there is a sequence of transitions in T (as in Definition 1.2) where the first transition originates in the global initial state q^0 and the last transition is $q \xrightarrow{\alpha}_T q'$. As $q \xrightarrow{\alpha}_{E(R)} q'$ is not reachable in $E(R)$ it follows that there is a transition $\bar{q} \xrightarrow{\bar{\alpha}}_T \bar{q}'$ in the sequence such that $\bar{q} \xrightarrow{\bar{\alpha}}_{E(R)} \bar{q}'$ is not reachable in $E(R)$. Let $\bar{q} \xrightarrow{\bar{\alpha}}_T \bar{q}'$ be the first transition in the sequence with this property, i.e., $\bar{q} \xrightarrow{\bar{\alpha}}_{E(R)} \bar{q}'$ is not even included in the transition relation of $E(R)$. By the first case follows that there is no transition $q_C \xrightarrow{\bar{\alpha}}_R q'_C$ in R with $\bar{q}{\downarrow_C} = q_C$ and $\bar{q}'{\downarrow_C} = q'_C$, i.e., this is a contradiction because R is an abstract over-approximation of T.

Let $E(R)$ be an over-approximation of T then we want to show that R is an over-approximation of T'' (see Definition 3.1). Let $q_C \xrightarrow{\alpha}_{T''} q'_C$ be a reachable transition in T''. This is, there is a reachable transition $q \xrightarrow{\alpha}_T q'$ in T with $q{\downarrow_C} = q_C$ and $q'{\downarrow_C} = q'_C$. As $E(R)$ is an over-approximation of T it follows that $q \xrightarrow{\alpha}_{E(R)} q'$ is reachable in $E(R)$ as well. According to the preliminaries of Lemma 3.1, there is a transition $\bar{q}_C \xrightarrow{\alpha}_R \bar{q}'_C$ in R with $q{\downarrow_C} = \bar{q}_C$ and $q'{\downarrow_C} = \bar{q}'_C$. It is easy to see that $\bar{q}_C = q_C$ and $\bar{q}'_C = q'_C$. This is, if $E(R)$ is an over-approximation of T and $q_C \xrightarrow{\alpha}_{T''} q'_C$ is a reachable transition in T'' then $q_C \xrightarrow{\alpha}_R q'_C$ is a transition included in the transition relation of R. It remains to be shown that $q_C \xrightarrow{\alpha}_R q'_C$ is reachable in R.

Let $q_C \xrightarrow{\alpha}_{T''} q'_C$ be a reachable transition in T'' and assume that $q_C \xrightarrow{\alpha}_R q'_C$ is not reachable in R. There is a sequence of transitions in T'' (as in Definition 1.2) where the first transition originates in the initial state q^0_C and the last transition equals $q_C \xrightarrow{\alpha}_{T''} q'_C$. As $q_C \xrightarrow{\alpha}_R q'_C$ is not reachable

in R it follows that there is a transition $\bar{q}_C \xrightarrow{\alpha}_{T''} \bar{q}'_C$ in the sequence such that $\bar{q}_C \xrightarrow{\alpha}_R \bar{q}'_C$ is not reachable in R. Let $\bar{q}_C \xrightarrow{\alpha}_{T''} \bar{q}'_C$ be the first transition in the sequence with this property. The existence of this transition is a contradiction because, as we showed above, $\bar{q}_C \xrightarrow{\alpha}_R \bar{q}'_C$ is a transition in R and is thus reachable. □

Proof of Lemma 3.2

Proof. Let $q \xrightarrow{\alpha}_T q'$ be a reachable transition in T. Then there is a sequence of transitions

$$q^0 \xrightarrow{\alpha^1}_T q^1, q^1 \xrightarrow{\alpha^2}_T q^2, \ldots, q^{k-1} \xrightarrow{\alpha^k}_T q^k$$

in T with $q = q^{k-1}$, $\alpha = \alpha^k$ and $q' = q^k$. Let $i \in \{1, \ldots, k\}$ then by the definition of S_C follows that the transition $q^{i-1}{\downarrow_C} \xrightarrow{\alpha^i}_{S_C} q^i{\downarrow_C}$ is in S_C. It follows that all transitions in the sequence projected on the components in C form a sequence in S_C that starts in q^0_C and ends in $q{\downarrow_C} \xrightarrow{\alpha}_{S_C} q'{\downarrow_C}$. This is, $q{\downarrow_C} \xrightarrow{\alpha}_{S_C} q'{\downarrow_C}$ is a reachable transition in S_C. □

Proof of Theorem 3.1

Proof. Let $S' = (Q_C, Int, \rightarrow_{S'}, q^0_C) = EM(S, R)$. We assume that S' is not an abstract over-approximation of T, i.e., there is a transition $\tilde{q} \xrightarrow{\tilde{\alpha}}_T \tilde{q}'$ reachable in T such that $\tilde{q}{\downarrow_C} \xrightarrow{\tilde{\alpha}}_{S'} \tilde{q}'{\downarrow_C}$ is not a reachable transition in S'. As $\tilde{q} \xrightarrow{\tilde{\alpha}}_T \tilde{q}'$ is reachable in T, there is a path starting in q^0 to $\tilde{q}$ in T. Let $q \xrightarrow{\alpha}_T q'$ be the last transition on this path with $q{\downarrow_C}$ reachable in S', i.e., $q{\downarrow_C} \xrightarrow{\alpha}_{S'} q'{\downarrow_C}$ is not a transition in S'. As S is an abstract over-approximation of T, $q{\downarrow_C} \xrightarrow{\alpha}_S q'{\downarrow_C}$ is reachable in S. This means that $q{\downarrow_C} \xrightarrow{\alpha}_{S'} q'{\downarrow_C}$ is not a reachable transition in S' because there is no transition $q_D \xrightarrow{\alpha}_R q'_D$ reachable in R with $q_D{\downarrow_C} = q_C{\downarrow_D}$ and $q'_D{\downarrow_C} = q'_C{\downarrow_D}$. This is a contradiction, as R is an abstract over-approximation and thus $q{\downarrow_D} \xrightarrow{\alpha}_R q'{\downarrow_D}$ has to be reachable in R.

It follows that S' is an abstract over-approximation of T. □

Proof of Lemma 3.3

Proof. Let $i \in K$ be a component and q_i a state in the local behavior T_i of i. Assume that we have a technique available that calculates an exact abstract over-approximation $S = (Q_C, Int, \rightarrow_S, q_C^0)$ for the input Sys and $C \subseteq K$ in polynomial time in the size of the specification of Sys. Let $C = \{i\}$ then the size of the transition relation $\rightarrow_S$ is bounded by $|Q_i|^2 \cdot |Int|$, i.e., the size of S is polynomial in the size of the specification of Sys. This means that we can check in polynomial time whether the state q_i is reachable in S. As S is exact, we can conclude that there is a state q reachable in the global behavior T of Sys where component i is in state q_i if and only if q_i is reachable in S. This is a direct contradiction to Corollary 2.1 because deciding this problem is PSPACE-complete. □

Proof of Lemma 3.4

Proof. We assume that $\{R'_C\}_{C \in \mathbb{C}} \neq \{R''_C\}_{C \in \mathbb{C}}$. Note that this means that there is $C \in \mathbb{C}$ with $R'_C \neq R''_C$, i.e., there is a transition $q_C \xrightarrow{\alpha}_{R'_C} q'_C$ in R'_C such that $q_C \xrightarrow{\alpha}_{R''_C} q'_C$ is not a transition in R''_C or vice versa. Without loss of generality we assume the first.

R''_C is a refined version of R_C for $C \in \mathbb{C}$, i.e., R_C is an over-approximation of R''_C. Let $\{\tilde{R}_C\}_{C \in \mathbb{C}}$ be the family where

- $\{\tilde{R}_C\}_{C \in \mathbb{C}}$ resulted from applications of the Edge-Match operator on $\{R_C\}_{C \in \mathbb{C}}$ which correspond to a prefix of seq_1,
- $\tilde{R}_C$ is an over-approximation of R''_C for each $C \in \mathbb{C}$ and
- the application of the Edge-Match operator that corresponds to the next entry in seq_1 would violate the last item, e.g., if (C, D) is the

next entry in seq_1 then $RM(\tilde{R}_C, \tilde{R}_D)$ results in a transition system $\tilde{R}'_C$ such that there is a transition $q_C \xrightarrow{\alpha}_{R''_C} q'_C$ in R''_C but $q_C \xrightarrow{\alpha}_{\tilde{R}'_C} q'_C$ is not a transition in $\tilde{R}'_C$.

This means, that there is no reachable transition $q_D \xrightarrow{\alpha}_{\tilde{R}_D} q'_D$ in $\tilde{R}_D$ with $q_D{\downarrow_C} = q_C{\downarrow_D}$ and $q'_D{\downarrow_C} = q'_C{\downarrow_D}$. There cannot be a reachable transition of this form in R''_D as well, because $\tilde{R}_D$ is an over-approximation of R''_D. This means that $EM(R''_C, R''_D)$ results in the removal of a transition in R''_C. This is a contradiction to the assumption that $\{R''_C\}_{C\in\mathbb{C}}$ is a fixed-point with respect to the Edge-Match operator. □

Proof of Lemma 3.5

Proof. Let seq be a sequence of the Edge-Match operator on $\{S_C\}_{C\in\mathbb{C}}$ which result in the fixed-point $\{R_C\}_{C\in\mathbb{C}}$, e.g., a sequences of tuples in $\mathbb{C}\times\mathbb{C}$. In seq might be tuples of the form (C, D) and (D, C) for $C \in \mathbb{C}'$. Let seq' equal the sequence seq where each occurrence of (C, D) is replaced by the tuples $(C, D_1), (C, D_2), \ldots, (C, D_k)$ and each occurrence of (D, C) is replaced by $(D_1, C), (D_2, C), \ldots, (D_k, C)$. We prove here by induction on the length of seq that seq' applied on $\{S_C\}_{C\in\tilde{\mathbb{C}}}$ results in $\{R'_C\}_{C\in\tilde{\mathbb{C}}}$.

For the initial families $\{S_C\}_{C\in\mathbb{C}}$ and $\{S_C\}_{C\in\tilde{\mathbb{C}}}$ holds trivially that $q_D \xrightarrow{\alpha}_{S_D} q'_D$ is a reachable transition in S_D if and only if $q_{D_i} \xrightarrow{\alpha}_{S_{D_i}} q'_{D_i}$ with $q_D{\downarrow_{D_i}} = q_{D_i}$ and $q'_D{\downarrow_{D_i}} = q'_{D_i}$ is a reachable transition in S'_{D_i} for each $1 \leq i \leq k$. This follows directly from the Definition of S_D in Lemma 3.2. Thus, property 1. and 2. hold for the initial families of abstract over-approximations.

Let $\{\bar{R}_C\}_{C\in\mathbb{C}}$ be the family of abstract over-approximations before the jth application of the Edge-Match operator in seq on $\{S_C\}_{C\in\mathbb{C}}$. Analogously, let $\{\bar{R}'_C\}_{C\in\tilde{\mathbb{C}}}$ be the family of abstract over-approximations before the respective application of the Edge-Match operator in seq' on $\{S_C\}_{C\in\tilde{\mathbb{C}}}$. This is, if the Edge-Match operator with respect to a tuple (C, D) respectively (D, C) in seq was applied in the construction process of $\{\bar{R}_C\}_{C\in\mathbb{C}}$ then,

in the construction process of $\{\bar{R}'_C\}_{C\in\tilde{\mathbb{C}}}$, the Edge-Match operator with respect to the corresponding sequence $(C, D_1), (C, D_2), \ldots, (C, D_k)$ respectively $(D_1, C), (D_2, C), \ldots, (D_k, C)$ in seq' was applied. We assume that property 1. and 2. hold for $\{\bar{R}_C\}_{C\in\mathbb{C}}$ and $\{\bar{R}'_C\}_{C\in\mathbb{C}}$.

We distinguish three cases for the next application described in seq.

If the jth application is of the form (C, C') with $C, C' \neq D$ then the next respective application of the Edge-Match operator in seq' is (C, C') as well. If we apply these on the respective abstract over-approximations then it is clear that $EM(\bar{R}_C, \bar{R}_{C'}) = EM(\bar{R}'_C, \bar{R}'_{C'})$. All other abstract over-approximations in both families remain the same, i.e., property 1. and 2. still hold.

If the jth application is of the form (C, D) then we consider the applications of the Edge-Match operator on $\{\bar{R}'_C\}_{C\in\tilde{\mathbb{C}}}$ that correspond to the respective subsequence $(C, D_1), (C, D_2), \ldots, (C, D_k)$ in seq'. Property 2. holds after the applications of the Edge-Match operator on both families because the over-approximations $\bar{R}_D$ respectively $\bar{R}'_{D_i}$ for $1 \leq i \leq k$ remain untouched. Let $\tilde{R}_C$ the respective refined version of $\bar{R}_C$ and $\tilde{R}'_C$ the respective refined version of $\bar{R}'_C$. Assume that the transition $q_C \xrightarrow{\alpha}_{\bar{R}_C} q'_C$ got removed in the refinement, but $q_C \xrightarrow{\alpha}_{\tilde{R}'_C} q'_C$ remains in $\tilde{R}'_C$, i.e., there is no transition $q_D \xrightarrow{\alpha}_{\bar{R}_D} q'_D$ reachable in $\bar{R}_D$ with $q_C{\downarrow_D} = q_D{\downarrow_C}$ and $q'_C{\downarrow_D} = q'_D{\downarrow_C}$. It follows from the second property that there is $1 \leq i \leq k$ such that the transition $q_{D_i} \xrightarrow{\alpha}_{\bar{R}'_{D_i}} q'_{D_i}$ with $q_D{\downarrow_{D_i}} = q_{D_i}$ and $q'_D{\downarrow_{D_i}} = q'_{D_i}$ is not reachable in $\bar{R}'_{D_i}$. This is a contradiction because then the transition $q_C \xrightarrow{\alpha}_{\tilde{R}'_C} q'_C$ is not included in $EM(\bar{R}'_C, \bar{R}'_{D_i})$. The other direction follows analogously.

If the jth application is of the form (D, C) then we consider the application of the Edge-Match operator on $\{\bar{R}'_C\}_{C\in\tilde{\mathbb{C}}}$ that corresponds to the respective subsequence $(D_1, C), (D_2, C), \ldots, (D_k, C)$ in seq'. Property 1. holds after the applications of the Edge-Match operator on both families because the over-approximations $\bar{R}_C$ respectively $\bar{R}'_C$ remain untouched. Let $\tilde{R}_D$ the respective refined version of $\bar{R}_D$ and $\tilde{R}'_{D_i}$ the respective refined version of

$\bar{R}'_{D_i}$ for $1 \leq i \leq k$. Assume that $q_D \xrightarrow{\alpha}_{\tilde{R}_D} q'_D$ is reachable in $\tilde{R}_D$ but there is $1 \leq i \leq k$ such that $q_{D_i} \xrightarrow{\alpha}_{\bar{R}'_{D_i}} q'_{D_i}$ with $q_D{\downarrow_{D_i}} = q_{D_i}$ and $q'_D{\downarrow_{D_i}} = q'_{D_i}$ got removed from $\bar{R}'_{D_i}$ during the refinement process, i.e., the second property is violated after the refinement. It follows that there is no transition $q_C \xrightarrow{\alpha}_{\bar{R}'_C} q'_C$ with $q_C{\downarrow_{D_i}} = q_{D_i}{\downarrow_C}$ and $q'_C{\downarrow_{D_i}} = q_{D_i}{\downarrow_{C'}}$ reachable in $\bar{R}'_C$. This is a contradiction because property 1. holds, i.e., $\bar{R}_C = \bar{R}'_C$ and thus $q_D \xrightarrow{\alpha}_{\bar{R}_D} q'_D$ should have been removed by $EM(\bar{R}_D, \bar{R}_C)$. The other direction follows analogously. □

Proof of Lemma 3.6

Proof. Let R'_{D_1} be $EM(S_{D_1}, R'_{D_2})$ restricted to reachable transitions, i.e., the result of the application of the Edge-Match operator on S_{D_1} with R'_{D_2} restricted to reachable transitions. If now

a) $R'_{D_1} = EM(R'_{D_1}, R'_C)$ for all $C \in \mathbb{C}'$ and

b) $R'_C = EM(R'_C, R'_{D_1})$ for all $C \in \mathbb{C}'$

then it follows that the family of abstract over-approximations $\{R'_C\}_{C \in \mathbb{C}}$ is the Edge-Match fixed-point of the family $\{S_C\}_{C \in \mathbb{C}}$. The first part from the assumption then follows from Lemma 3.4 because this fixed-point and the fixed-point $\{R_C\}_{C \in \mathbb{C}}$ are identical. The second part follows because $R'_{D_1} = R_{D_1}$ and $R'_{D_2} = EM(R'_{D_2}, R_{D_1})$ as well as $R_{D_1} = EM(R_{D_1}, R'_{D_2})$.

Assume that a) does not hold, i.e., there is $C \in \mathbb{C}'$ such that $R'_{D_1} \neq EM(R'_{D_1}, R'_C)$. This means that there is a reachable transition $q_{D_1} \xrightarrow{\alpha}_{R'_{D_1}} q'_{D_1}$ in R'_{D_1} such that there is no reachable transition $q_C \xrightarrow{\alpha}_{R'_C} q'_C$ in R'_C with $q_C{\downarrow_{D_1}} = q_{D_1}{\downarrow_C}$ and $q'_C{\downarrow_{D_1}} = q'_{D_1}{\downarrow_C}$. Note that from the definition of R'_{D_1} follows that $C \neq D_2$ because there is a transition $q_{D_2} \xrightarrow{\alpha}_{R'_{D_2}} q'_{D_2}$ reachable in R'_{D_2} with $q_{D_2}{\downarrow_{D_1}} = q_{D_1}$ and $q'_{D_2}{\downarrow_{D_1}} = q'_{D_1}$. As $D_1 \subseteq D_2$ it follows that $q_{D_2} \xrightarrow{\alpha}_{R'_{D_2}} q'_{D_2}$ is removed in $EM(R'_{D_2}, R'_C)$. This is a contradiction to the assumption that the family $\{R'_C\}_{C \in \mathbb{C}'}$ is the Edge-Match fixed-point of $\{S_C\}_{C \in \mathbb{C}'}$.

We assume now that b) does not hold, i.e., there is $C \in \mathbb{C}'$ such that $R'_C \neq EM(R'_C, R'_{D_1})$. This is the case if there is a transition $q_C \xrightarrow{\alpha}_{R'_C} q'_C$ reachable in R'_C such that there is no transition $q_{D_1} \xrightarrow{\alpha}_{R'_{D_1}} q'_{D_1}$ with $q_C{\downarrow_{D_1}} = q_{D_1}{\downarrow_C}$ and $q'_C{\downarrow_{D_1}} = q'_{D_1}{\downarrow_C}$ reachable in R'_{D_1}. Assume that $C \neq D_2$. Because of the assumption that the family $\{R'_C\}_{C \in \mathbb{C}'}$ is the Edge-Match fixed-point of $\{S_C\}_{C \in \mathbb{C}'}$ it follows that a respective transition $q_{D_2} \xrightarrow{\alpha}_{R'_{D_2}} q'_{D_2}$ with $q_{D_2}{\downarrow_C} = q_C{\downarrow_{D_2}}$ and $q'_{D_2}{\downarrow_C} = q'_C{\downarrow_{D_2}}$ must be reachable in R'_{D_2}. It follows that the transition $q_{D_2} \xrightarrow{\alpha}_{R'_{D_2}} q'_{D_2}$ is removed in $EM(R'_{D_2}, R'_{D_1})$. Thus, without loss of generality, we can assume that $C = D_2$ and that there is a transition that is reachable in R'_{D_2} which projection on D_1 is not reachable in R'_{D_1}. Assume, without loss of generality, that $q_{D_2}{\downarrow_{D_1}}$ is a reachable state in R'_{D_1}. By the definition of S_{D_2} in Lemma 3.2 and the fact that $q_{D_2}{\downarrow_{D_1}}$ is reachable in R'_{D_1} follows that the transition $q_{D_2}{\downarrow_{D_1}} \xrightarrow{\alpha}_{S_{D_1}} q'_{D_2}{\downarrow_{D_1}}$ is reachable in S_{D_1}, i.e., this transition got removed by the operation $EM(S_{D_1}, R'_{D_2})$. This is a contradiction because $q_{D_2} \xrightarrow{\alpha}_{R'_{D_2}} q'_{D_2}$ is a reachable transition in R'_{D_2}. □

A.3 Proofs from Chapter 6

Proof of Theorem 6.1

Proof. Let r'_S and r'_R be r_S respectively r_R restricted to reachable tuples.

Let $\mathsf{t} \in \mathsf{r}$, then t corresponds to a transition $q_C \xrightarrow{\alpha}_{S'} q'_C$ in $S' = EM(S, R)$. This means, that $q_C \xrightarrow{\alpha}_S q'_C$ is a reachable transition in S, i.e., $\mathsf{t} \in \mathsf{r}'_S$. The transition is in S' after the application of the Edge-Match operator, i.e., there is a reachable transition $q_D \xrightarrow{\alpha}_R q'_D$ in R such that $q_C{\downarrow_D} = q_D{\downarrow_C}$ and $q'_C{\downarrow_D} = q'_D{\downarrow_C}$. Let $\mathsf{t}_R \in \mathsf{r}'_R$ be the tuple that corresponds to $q_D \xrightarrow{\alpha}_R q'_D$. This means that $\mathsf{t} \in \mathsf{r}'_S \ltimes \mathsf{r}'_R$ as $\mathsf{t} \in \mathsf{r}'_S$ and $\mathsf{t}_R \in \mathsf{r}'_R$.

Let $\mathsf{t} \in \mathsf{r}'_S \ltimes \mathsf{r}'_R$ then t corresponds to a transition $q_C \xrightarrow{\alpha}_S q'_C$ that is reachable in S. Furthermore, there is a tuple $\mathsf{t}_R \in \mathsf{r}'_R$ that agrees with t_S on shared attributes. Let t_R correspond to the reachable transition $q_D \xrightarrow{\alpha}_R q'_D$ in R.

This means that $q_C \xrightarrow{\alpha}_S q'_C$ is a transition in $EM(S,R)$ as $q_C{\downarrow_D} = q_D{\downarrow_C}$ and $q'_C{\downarrow_D} = q'_D{\downarrow_C}$, i.e., $\mathsf{t} \in \mathsf{r}$. □

Proof of Corollary 6.1

Proof. Let $H_{\mathbb{C}} = (K, \mathbb{C})$ be acyclic. Let $\mathbb{C} = \{C_1, C_2, \ldots, C_k\}$ and $\mathcal{R} = \{\mathsf{R}_{C_1}, \mathsf{R}_{C_2}, \ldots, \mathsf{R}_{C_k}\}$ be the relational database scheme that consists of the relational schemata associated with the sets of components in $\mathbb{C}$. It is easy to see that $\mathcal{R}$ is acyclic if the hypergraph $H_{\mathbb{C}} = (K, \mathbb{C})$ is acyclic (see Example 6.5 for an illustration).

Let $\{R'_C\}_{C\in\mathbb{C}}$ be the Edge-Match fixed-point of $\{R_C\}_{C\in\mathbb{C}}$ and let

$$\mathfrak{d} = \{\mathsf{r}_{C_1}(\mathsf{R}_{C_1}), \mathsf{r}_{C_2}(\mathsf{R}_{C_2}), \ldots, \mathsf{r}_{C_k}(\mathsf{R}_{C_k})\}$$

be the database on $\mathcal{R}$ that consists of the relations associated with the transition systems in $\{R'_C\}_{C\in\mathbb{C}}$.

As $\{R'_C\}_{C\in\mathbb{C}}$ is a fixed-point with respect to the application of the Edge-Match operator each abstract over-approximation is restricted to reachable transitions and no application of the Edge-Match operator on a pair of abstract over-approximations results in a refinement. It is easy to see that it follows from Theorem 6.1 that similarly each tuple in each relation in $\mathfrak{d}$ is reachable and no application of a semijoin on a pair of relations results in a removal of tuples. As $\mathcal{R}$ is acyclic it follows that each relation in $\mathfrak{d}$ is a full reduction.

Let $C \in \mathbb{C}$ and $q_C \xrightarrow{\alpha}_{R'_C} q'_C$ a reachable transition in R'_C then $q_C \xrightarrow{\alpha}_{R'_C} q'_C$ corresponds to a reachable tuple $\mathsf{t}_C \in \mathsf{r}_C$ ($\mathsf{r}_C \in \mathfrak{d}$). As r_C is a full reduction it follows that $\mathsf{r}_C = \pi_{\mathsf{R}_C}(\bowtie(\mathfrak{d}))$, i.e., there is a tuple $\mathsf{t} \in \bowtie(\mathfrak{d})$ such that $\mathsf{t}_C = \mathsf{t}(\mathsf{R}_C)$ and for each $D \in \mathbb{C}$ $(D \neq C)$ $\mathsf{t}(\mathsf{R}_D) \in \mathsf{r}_D$. The tuple t corresponds to a transition $q \xrightarrow{\alpha}_T q'$ in the global behavior T of Sys and each $\mathsf{t}(\mathsf{R}_D)$ $(D \neq C)$ corresponds to a reachable transition $q_D \xrightarrow{\alpha}_{R'_D} q_D$ in R'_D with

- $q{\downarrow_C} = q_C$ and $q'{\downarrow_C} = q'_C$ and

- for all $D \in \mathbb{C}$ $(D \neq C)$ holds $q{\downarrow_D} = q_D$ and $q'{\downarrow_D} = q'_D$.

It follows that $q_C \xrightarrow{\alpha}_{R'_C} q'_C$ is legitimated and thus that $\{R'_C\}_{C \in \mathbb{C}}$ is legitimated. $\square$

Appendix B

Source Code

B.1 A Description Language for Interaction Systems

The code in Listing B.1 is an example for an interaction system specified in a description language that we use as an input for our tool that implements our approach to refine abstract over-approximations and establish deadlock-freedom. The code specifies a model of Tanenbaum's solution of the Dining Philosophers problem (see Section 5.3). The language is relatively minimalistic and features only the control structures *if* (restricted to integer comparison) and *for* (restricted to single step incrementation). In Line 3 the integer variable n is initialized with the value 20 which specifies the number of philosophers in the model. A *COMPONENT*-block (e.g., Line 7 to 16 models a philosopher) specifies a component by an initial state (Line 9) and a set of transitions – for example, in Line 10 a transition from a state named *init* to a state named *hp* labeled by *get_prior* is defined. A *CONNECTOR*-block defines an interaction[1]. For example, Line 44 to 48

[1] In [GS03], where interaction systems were introduced, interactions are called connectors.

models that philosopher i takes the fork on his left side

```
SYSTEM "philosophers";

VAR n = 20;

FOR(VAR i=0;n-1)
{
  COMPONENT "phil_"+i
  {
    INIT "init";
    TRANS "init"{"get_prior"}"hp";
    TRANS "hp"{"take_left"}"hl";
    TRANS "hp"{"take_right"}"hr";
    TRANS "hl"{"take_right"}"hlr";
    TRANS "hr"{"take_left"}"hlr";
    TRANS "hlr"{"put"}"init";
  }

  COMPONENT "fork_"+i
  {
    INIT "f";
    TRANS "f"{"take"}"t";
    TRANS "t"{"put"}"f";
  }

  COMPONENT "sem_"+i
  {
    INIT "free";
    TRANS "free"{"is_free"}"free";
    TRANS "free"{"take"}"taken";
    TRANS "taken"{"put"}"free";
  }
}

FOR(VAR i=0;n-1)
{
  CONNECTOR "priority_"+i
  {
    "phil_"+i:"get_prior";
    "sem_"+i:"take";
    "sem_"+(i-1)%n:"is_free";
    "sem_"+(i+1)%n:"is_free";
  }

  CONNECTOR "take_left_"+i
  {
    "phil_"+i:"take_left";
```

```
47      "fork_"+(i-1)%n:"take";
48    }
49
50    CONNECTOR "take_right_"+i
51    {
52      "phil_"+i:"take_right";
53      "fork_"+i:"take";
54    }
55
56    CONNECTOR "put_"+i
57    {
58      "phil_"+i:"put";
59      "fork_"+(i-1)%n:"put";
60      "fork_"+i:"put";
61      "sem_"+i:"put";
62    }
63  }
```

Listing B.1: "Model of Tanenbaum's Dining Philosophers."

B.2 Java Source Code

Our tool that we used to calculate the results presented in Chapter 5 is implemented in Java. In this section we describe the two most important methods in our implementation, this is, the method that restricts the behavior of an abstract over-approximation to reachable transitions and the method that implements the application of the Edge-Match operator on a pair of abstract over-approximations. Our implementation of Algorithm 2 in Chapter 3 is based on these two methods. In our tool, sets of transitions and sets of states are modeled by BDDs.

Listing B.2 depicts the method *reach* that restricts an abstract over-approximation, that is modeled by a BDD, to reachable transitions. We perform a symbolic reachability analysis on the behavior of an abstract over-approximation and restrict the transition relation to transitions that start in a reachable state. Symbolic reachability analyses were introduced by Coudert et al. [CBM90]. The analysis is based on successively extending a set of

reachable states, starting with the set that consists of the initial state, by its image until no new states are added. In Line 4 we initialize the BDD *a1* which represents the set of reachable states. In Line 7 to 17 the reachability analysis is performed. The restriction of the set of transitions is accomplished in Line 19.

```
1  public static void reach(Subsystem s,InteractionSystem is)
2  {
3    BDD a0 = is.getBDDFactory().zero();
4    BDD a1 = s.getInit().id();
5    BDD tmp;
6
7    while(!a0.equals(a1))
8    {
9      a0 = a1.id();
10     tmp = a1.and(s.getBDD());
11     tmp = tmp.exist(s.getVarsFrom().union(sys.getCodedActSet()));
12     BDDPairing p = B.makePair();
13     p.set(s.getVarsTo().getDomains(),s.getVarsFrom().getDomains());
14     tmp.replaceWith(p);
15     p.freepair();
16     a1.orWith(tmp);
17   }
18
19   s.getBDD().andWith(a1);
20   a0.free();
21 }
```

Listing B.2: "The method *reach*."

Listing B.3 depicts the method *EM* that implements the application of the Edge-Match operator on a pair of abstract over-approximations. Note, if S and R are abstract over-approximations, that the method implements the application of the Edge-Match operator on S and R and on R and S, i.e., S and R are refined. If an abstract over-approximation is refined, i.e., at least one transition is removed, then the respective transition system gets restricted to reachable transitions. The method returns *false* if and only if one of the abstract over-approximations is refined, i.e., if and only if at least one transition is removed in at least one of the two abstract over-approximations. We use this return value in order to check a condition

for the termination of our implementation of Algorithm 2 in Chapter 3. Our refinement procedure terminates if the Edge-Match operator applied on all (reasonable) pairs of abstract over-approximations does not cause any refinement.

The method works as follows. The input consists of two abstract over-approximations `s1` and `s2`. In Line 4 the BDD representation of the transition relation of `s2` is copied. In Line 6 to 10 the copy is projected to components that are shared between `s1` and `s2`. In Line 21 the transition relation of `s1` is restricted to transitions that have a corresponding transition in `s2` which agree on shared components. In Line 22 to 26 we test whether or not there are transitions removed from `s1`. If so, there might be transitions in `s1` that become unreachable. Thus, in this case, we restrict the refined transition relation of `s1` to reachable transitions. `s2` is refined in the same manner with respect to `s1`.

```
1  public static boolean EM(Subsystem s1,Subsystem s2)
2  {
3    BDD s1BDD = s1.getBDD().id();
4    BDD s2BDD = s2.getBDD().id();
5
6    Set<Component> minus1 = s2.minus(s1);
7    for(Component k:minus1)
8    {
9     s2BDD=s2BDD.exist(k.getFromDomain().set().union(k.getToDomain().set()));
10   }
11
12   Set<Component> minus2 = s1.minus(s2);
13   for(Component k:minus2)
14   {
15    s1BDD=s1BDD.exist(k.getFromDomain().set().union(k.getToDomain().set()));
16   }
17
18   boolean b = true;
19
20   BDD s1tmp = s1.getBDD().id();
21   s1.getBDD().andWith(s2BDD);
22   if(!s1.getBDD().equals(s1tmp))
23   {
24     b = false;
25     ISToolBox.reach(s1);
26   }
```

```
27    s1tmp.free();
28
29    BDD s2tmp = s2.getBDD().id();
30    s2.getBDD().andWith(s1BDD);
31    if(!s2.getBDD().equals(s2tmp))
32    {
33      b = false;
34      ISToolBox.reach(s2);
35    }
36    s2tmp.free();
37
38    return b;
39  }
```

Listing B.3: "The method *EM*."

Bibliography

[AC05] Paul C. Attie and Hana Chockler. Efficiently Verifiable Conditions for Deadlock-Freedom of Large Concurrent Programs. In *Proceedings of the 6th international conference on Verification, Model Checking, and Abstract Interpretation*, VMCAI'05, pages 465–481, Berlin, Heidelberg, 2005. Springer-Verlag.

[ACG86] Sudhir Ahuja, Nicholas Carriero, and David Gelernter. Linda and Friends. *Computer*, 19(8):26–34, August 1986.

[AG97] Robert Allen and David Garlan. A Formal Basis for Architectural Connection. *ACM Trans. Softw. Eng. Methodol.*, 6(3):213–249, July 1997.

[AKY99] Rajeev Alur, Sampath Kannan, and Mihalis Yannakakis. Communicating Hierarchical State Machines. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, ICAL '99, pages 169–178, London, UK, UK, 1999. Springer-Verlag.

[Arb04] Farhad Arbab. Reo: a Channel-Based Coordination Model for Component Composition. *Mathematical. Structures in Comp. Sci.*, 14(3):329–366, June 2004.

[Arn94] André Arnold. *Finite Transition Systems: Semantics of Communicating Systems.* Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1994.

[BBG11] Borzoo Bonakdarpour, Marius Bozga, and Gregor Goessler. A Theory of Fault Recovery for Component-Based Models. In *Proceedings of the 2011 IEEE 30th International Symposium on Reliable Distributed Systems*, SRDS '11, pages 265–270, Washington, DC, USA, 2011. IEEE Computer Society.

[BBNS09] Saddek Bensalem, Marius Bozga, Thanh-Hung Nguyen, and Joseph Sifakis. D-Finder: A Tool for Compositional Deadlock Detection and Verification. In *Proceedings of the 21st International Conference on Computer Aided Verification*, CAV '09, pages 614–619, Berlin, Heidelberg, 2009. Springer-Verlag.

[BBS06] Ananda Basu, Marius Bozga, and Joseph Sifakis. Modeling Heterogeneous Real-time Components in BIP. In *Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods*, SEFM '06, pages 3–12, Washington, DC, USA, 2006. IEEE Computer Society.

[BBSN08] Saddek Bensalem, Marius Bozga, Joseph Sifakis, and Thanh-Hung Nguyen. Compositional Verification for Component-Based Systems and Application. In *Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis*, ATVA '08, pages 64–79, Berlin, Heidelberg, 2008. Springer-Verlag.

[BCCZ99] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic Model Checking without BDDs. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, TACAS

'99, pages 193–207, London, UK, UK, 1999. Springer-Verlag.

[BCD02] Marco Bernardo, Paolo Ciancarini, and Lorenzo Donatiello. Architecting Families of Software Systems With Process Algebras. *ACM Trans. Softw. Eng. Methodol.*, 11(4):386–426, October 2002.

[BCL91] J. R. Burch, Edmund M. Clarke, and David E. Long. Symbolic Model Checking with Partitioned Transition Relations. In *1991 University of California, Santa Cruz Conference on Advanced Research in VLSI*, pages 49–58. North-Holland, 1991.

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. *Inf. Comput.*, 98(2):142–170, June 1992.

[BFMY83] Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the Desirability of Acyclic Database Schemes. *J. ACM*, 30(3):479–513, July 1983.

[BGL+11] Saddek Bensalem, Andreas Griesmayer, Axel Legay, Thanh-Hung Nguyen, Joseph Sifakis, and Rongjie Yan. D-finder 2: Towards Efficient Correctness of Incremental Design. In *Proceedings of the Third international conference on NASA Formal methods*, NFM'11, pages 453–458, Berlin, Heidelberg, 2011. Springer-Verlag.

[BHH+06] Hubert Baumeister, Florian Hacklinger, Rolf Hennicker, Alexander Knapp, and Martin Wirsing. A Component Model for Architectural Programming. *Electronic Notes in Theoretical Computer Science*, 160:75–96, August 2006.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model*

Checking. The MIT Press, 2008.

[BR91] Stephen D. Brookes and A. W. Roscoe. Deadlock Analysis in Networks of Communicating Processes. *Distributed Computing*, 4:209–230, 1991.

[Bry86] Randal E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986.

[BVW94] Orna Bernholtz, Moshe Y. Vardi, and Pierre Wolper. An Automata-Theoretic Approach to Branching-Time Model Checking (Extended Abstract). In *CAV*, pages 142–155, 1994.

[CBM90] O. Coudert, C. Berthet, and J. C. Madre. Verification of Synchronous Sequential Machines Based on Symbolic Execution. In *Proceedings of the international workshop on Automatic verification methods for finite state systems*, pages 365–373, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[CC77] P. Cousot and R. Cousot. Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.

[CCK+05] Ran Canetti, Ling Cheung, Dilsun Kirli Kaynar, Moses Liskov, Nancy A. Lynch, Olivier Pereira, and Roberto Segala. Using Probabilistic I/O Automata to Analyze an Oblivious Transfer Protocol. *IACR Cryptology ePrint Archive*, 2005:452, 2005.

[CCQ94] Gianpiero Cabodi, P. Camurati, and Stefano Quer. Symbolic

Exploration of Large Circuits with Enhanced Forward/Backward Traversals. In *Proceedings of the conference on European design automation*, EURO-DAC '94, pages 22–27, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs*, pages 52–71, 1981.

[CEP95] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity Results for 1-Safe Nets. *Theor. Comput. Sci.*, 147(1-2):117–136, August 1995.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8:244–263, 1986.

[CGL94] Edmund M. Clarke, Orna Grumberg, and David E. Long. Model Checking and Abstraction. *ACM Trans. Program. Lang. Syst.*, 16(5):1512–1542, 1994.

[CHM+93] Hyunwoo Cho, Gary D. Hachtel, Enrico Macii, Bernard Plessier, and Fabio Somenzi. Algorithms for Approximate FSM Traversal. In *Proceedings of the 30th international Design Automation Conference*, DAC '93, pages 25–30, New York, NY, USA, 1993. ACM.

[CJEF96] Edmund M. Clarke, Somesh Jha, Reinhard Enders, and Thomas Filkorn. Exploiting Symmetry in Temporal Logic Model Checking. *Formal Methods in System Design*, 9(1/2):77–104, 1996.

[CS01] R. Cleaveland and O. Sokolsky. Equivalence and Preorder Checking for Finite-State Systems. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, chapter 6, pages 391–424. Elvesier Science B.V., North-Holland, Amsterdam, 2001.

[dAH01] Luca de Alfaro and Thomas A. Henzinger. Interface Automata. *SIGSOFT Softw. Eng. Notes*, 26(5):109–120, September 2001.

[Dij02] Edsger W. Dijkstra. Hierarchical Ordering of Sequential Processes. In Per Brinch Hansen, editor, *The Origin of Concurrent Programming*, pages 198–227. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[Ell97] Clarence Ellis. Team Automata for Groupware Systems. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work: the integration challenge*, GROUP '97, pages 415–424, New York, NY, USA, 1997. ACM.

[EN94] Javier Esparza and Mogens Nielsen. Decidability Issues for Petri Nets - A Survey. *Bulletin of the European Association for Theoretical Computer Science*, 52:245–262, 1994.

[ES96] E. Allen Emerson and A. Prasad Sistla. Symmetry and Model Checking. *Form. Methods Syst. Des.*, 9(1-2):105–131, August 1996.

[Esp98] Javier Esparza. Decidability and Complexity of Petri Net Problems - An Introduction. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets*, pages 374–428, London, UK, UK, 1998. Springer-Verlag.

[Flo67] R. W. Floyd. Assigning Meaning to Programs. In *Proceedings of the Symposium on Applied Maths*, volume 19, pages 19–32. AMS, 1967.

[GD99] Shankar G. Govindaraju and David L. Dill. Approximate Symbolic Model Checking Using Overlapping Projections. In *First International Workshop on Symbolic Model Checking (SMC99) at Federated Logic Conference (FLOC)*, July 1999. Trento, Italy.

[GDHH98] Shankar G. Govindaraju, David L. Dill, Alan J. Hu, and Mark A. Horowitz. Approximate Reachability with BDDs Using Overlapping Projections. In *Proceedings of the 35th annual Design Automation Conference*, DAC '98, pages 451–456, New York, NY, USA, 1998. ACM.

[GGMC+06] Gregor Gößler, Susanne Graf, Mila E. Majster-Cederbaum, Moritz Martens, and Joseph Sifakis. Ensuring Properties of Interaction Systems. In *Program Analysis and Compilation*, pages 201–224, 2006.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[GPS96] Patrice Godefroid, Doron Peled, and Mark G. Staskauskas. Using Partial-Order Methods in the Formal Validation of Industrial Concurrent Programs. In *ISSTA*, pages 261–269, 1996.

[Gra79] H. Graham. On the Universal Relation. Technical report, University of Toronto. Computer Systems Research Group, 1979.

[GS97] Susanne Graf and Hassen Saïdi. Construction of Abstract State Graphs with PVS. In *Proceedings of the 9th International Conference on Computer Aided Verification*, CAV '97, pages 72–83, London, UK, UK, 1997. Springer-Verlag.

[GS03] Gregor Gössler and Joseph Sifakis. Composition for Component-Based Modeling. In *Proceedings of FMCO'02*, volume 2852 of *LNCS*, pages 443–466. Springer, 2003.

[GSM07] Vijay K. Garg, Chakarat Skawratananond, and Neeraj Mittal. Timestamping Messages and Events in a Distributed System Using Synchronous Communication. *Distributed Computing*, 19(5-6):387–402, 2007.

[GW92] Patrice Godefroid and Pierre Wolper. Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties. In *CAV '91: Proceedings of the 3rd International Workshop on Computer Aided Verification*, pages 332–342, London, UK, 1992. Springer-Verlag.

[HD95] Monika Heiner and Peter Deussen. P.: Petri Net Based Qualitative Analysis - a Case Study. Technical report, BTU Cottbus, Dep. of CS, Techn. Report I-08/1995, 1995.

[Heg91] Stephen J. Hegner. Pairwise-Definable Subdirect Decompositions of General Database Schemata. In *Proceedings of the 3rd symposium on Mathematical fundamentals of database and knowledge base systems*, pages 243–257, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[Hil96] Jane Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, New York, NY, USA, 1996.

[HJK10] Rolf Hennicker, Stephan Janisch, and Alexander Knapp. On the Observable Behaviour of Composite Components. *Electron. Notes Theor. Comput. Sci.*, 260:125–153, January 2010.

[HJR93] Rodney R. Howell, Petr Jančar, and Louis E. Rosier. Completeness Results for Single-Path Petri Nets. *Inf. Comput.*, 106(2):253–265, October 1993.

[HKW12] Yi-Ling Hwong, Vincent J. J. Kusters, and Tim A. C. Willemse. Analysing the Control Software of the Compact Muon Solenoid Experiment at the Large Hadron Collider. In *Proceedings of the 4th IPM international conference on Fundamentals of Software Engineering*, FSEN'11, pages 174–189, Berlin, Heidelberg, 2012. Springer-Verlag.

[Hoa69] C. A. R. Hoare. An Axiomatic Basis for Computer Programming. *Commun. ACM*, 12(10):576–580, October 1969.

[Hoa85] Charles A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International series in computer science. Prentice-Hall, Englewood Cliffs, NJ [u.a.], 1985.

[Hol97] Gerard J. Holzmann. The Model Checker SPIN. *IEEE Trans. Softw. Eng.*, 23(5):279–295, May 1997.

[ID96] C. Norris Ip and David L. Dill. Better Verification through Symmetry. *Form. Methods Syst. Des.*, 9(1-2):41–75, August 1996.

[IU01] Paola Inverardi and Sebastián Uchitel. Proving Deadlock Freedom in Component-Based Programming. In *Proceedings of the 4th International Conference on Fundamental Approaches to Software Engineering*, FASE '01, pages 60–75, London, UK,

UK, 2001. Springer-Verlag.

[Jen96] Kurt Jensen. Condensed State Spaces for Symmetrical Coloured Petri Nets. *Form. Methods Syst. Des.*, 9(1-2):7–40, August 1996.

[JLL77] Neil D. Jones, Lawrence H. Landweber, and Y. Edmund Lien. Complexity of Some Problems in Petri Nets. *Theor. Comput. Sci.*, 4(3):277–299, 1977.

[KBS88] Marek Karpinski, Hans Kleine Büning, and Peter H. Schmitt. On the Computational Complexity of Quantified Horn Clauses. In *Proceedings of the 1st Workshop on Computer Science Logic*, CSL '87, pages 129–137, London, UK, UK, 1988. Springer-Verlag.

[KEH+09] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. seL4: Formal Verification of an OS Kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 207–220, New York, NY, USA, 2009. ACM.

[KK96] Young Chan Kim and Tag Gon Kim. Petri Nets Modeling and Analysis Using Extended Bag-Theoretic Relational Algebra. *Trans. Sys. Man Cyber. Part B*, 26(4):599–605, August 1996.

[KLSV06] Dilsun K. Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager. *The Theory of Timed I/O Automata (Synthesis Lectures in Computer Science)*. Morgan & Claypool Publishers, 2006.

[Koz82] Dexter Kozen. Results on the Propositional μ-Calculus. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 348–359, London, UK, UK, 1982. Springer-Verlag.

[KS83] Paris C. Kanellakis and Scott A. Smolka. CCS Expressions, Finite State Processes, and three Problems of Equivalence. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, PODC '83, pages 228–240, New York, NY, USA, 1983. ACM.

[Kup95] Orna Kupferman. *Model Checking for Branching-Time Temporal Logics*. Phd thesis, Israel Institute of Technology, Haifa, Israel, 1995.

[Kur94] Robert P. Kurshan. *Computer-Aided Verification of Coordinating Processes: the Automata-Theoretic Approach*. Princeton University Press, Princeton, NJ, USA, 1994.

[KVW00] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. An Automata-Theoretic Approach to Branching-Time Model Checking. *J. ACM*, 47(2):312–360, 2000.

[Lam77] L. Lamport. Proving the Correctness of Multiprocess Programs. *IEEE Trans. Softw. Eng.*, 3(2):125–143, March 1977.

[Lam09] Christian Lambertz. Exploiting Architectural Constraints and Branching Bisimulation Equivalences in Component-Based Systems. In *Proceedings of the Doctoral Symposium on Formal Methods (FM2009)*, number 0915 in University of Eindhoven Technical Report, pages 1–7. University of Eindhoven, 2009.

[Lip76] R. Lipton. The Reachability Problem Requires Exponential

Space. Technical Report 62, Yale University, 1976.

[LL88] T. T. Lee and M. Y. Lai. A Relational Algebraic Approach to Protocol Verification. *Software Engineering IEEE Transactions on*, 14(2):184–193, 1988.

[LL95] Claus Lewerentz and Thomas Lindner, editors. *Formal Development of Reactive Systems - Case Study Production Cell*, volume 891 of *Lecture Notes in Computer Science*. Springer, 1995.

[LMC11] Christian Lambertz and Mila Majster-Cederbaum. Analyzing Component-Based Systems on the Basis of Architectural Constraints. In Farhad Arbab and Marjan Sirjani, editors, *Proceedings of the 4th International Conference on Fundamentals of Software Engineering (FSEN 2011)*, 2011.

[Lon93] David E. Long. *Model Checking, Abstraction, and Compositional Verification*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, Pittsburgh, PA, USA, 1993. UMI Order No. GAX94-02579.

[LPJ+96] Woohyuk Lee, Abelardo Pardo, Jae-Young Jang, Gary Hachtel, and Fabio Somenzi. Tearing Based Automatic Abstraction for CTL Model Checking. In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, ICCAD '96, pages 76–81, Washington, DC, USA, 1996. IEEE Computer Society.

[LS85] Leslie Lamport and Fred B. Schneider. Formal Foundation for Specification and Verification. In *Distributed Systems*, volume 190 of *Lecture Notes in Computer Science*, chapter 5, pages 203–285. Springer Berlin / Heidelberg, 1985.

[LT87] Nancy A. Lynch and Mark R. Tuttle. Hierarchical Correctness Proofs for Distributed Algorithms. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, PODC '87, pages 137–151, New York, NY, USA, 1987. ACM.

[Mai83] David Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.

[Mar09] Moritz Martens. *Establishing Properties of Interaction Systems.* PhD thesis, University of Mannheim, November 2009.

[MCM08a] Mila Majster-Cederbaum and Moritz Martens. Compositional Analysis of Deadlock-Freedom for Tree-Like Component Architectures. In *Proceedings of the 8th ACM international conference on Embedded software*, EMSOFT '08, pages 199–206, New York, NY, USA, 2008. ACM.

[MCM08b] Mila Majster-Cederbaum and Christoph Minnameier. Deriving Complexity Results for Interaction Systems from 1-Safe Petri Nets. In *Proceedings of the 34th conference on Current trends in theory and practice of computer science*, SOFSEM'08, pages 352–363, Berlin, Heidelberg, 2008. Springer-Verlag.

[MCM08c] Mila Majster-Cederbaum and Christoph Minnameier. Everything Is PSPACE-Complete in Interaction Systems. In *Proceedings of the 5th international colloquium on Theoretical Aspects of Computing*, pages 216–227, Berlin, Heidelberg, 2008. Springer-Verlag.

[MCMM07] Mila Majster-Cederbaum, Moritz Martens, and Christoph Minnameier. A Polynomial-Time Checkable Sufficient Condition for Deadlock-Freedom of Component-Based Systems.

In *Proceedings of SOFSEM'07*, volume 4362, pages 888–899. Springer, 2007.

[MCMM08] Mila Majster-Cederbaum, Moritz Martens, and Christoph Minnameier. Liveness in Interaction Systems. *Electron. Notes Theor. Comput. Sci.*, 215:57–74, June 2008.

[MCS10] Mila Majster-Cederbaum and Nils Semmelrock. Reachability in Tree-Like Component Systems is PSPACE-Complete. *Electron. Notes Theor. Comput. Sci.*, 263:197–210, June 2010.

[MCS13a] Mila E. Majster-Cederbaum and Nils Semmelrock. A Basis for Compositionally Ensuring Safety Properties and its Connection to Relational Algebraic Operators (in revision). *Science of Computer Programming*, 2013.

[MCS13b] Mila E. Majster-Cederbaum and Nils Semmelrock. Reachability in Cooperating Systems with Architectural Constraints is PSPACE-Complete. In *GRAPHITE Electron. Notes Theor. Comput. Sci. (to appear)*, 2013.

[MCSW07] Mila E. Majster-Cederbaum, Nils Semmelrock, and Verena Wolf. Interaction Models for Biochemical Reactions. In *BIOCOMP*, pages 480–486, 2007.

[Mea55] George H. Mealy. A Method for Synthesizing Sequential Circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

[Mil82] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[Min10] Christoph Minnameier. *Interaction in Concurrent Systems.* PhD thesis, University of Mannheim, April 2010.

[MJH+98] In-Ho Moon, Jae-Young Jang, Gary D. Hachtel, Fabio Somenzi, Jun Yuan, and Carl Pixley. Approximate Reachability don't Cares for CTL Model Checking. In *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, ICCAD '98, pages 351–358, New York, NY, USA, 1998. ACM.

[MMC09a] Moritz Martens and Mila Majster-Cederbaum. Using Architectural Constraints for Deadlock-Freedom of Component Systems with Multiway Cooperation. In *TASE*, pages 225–232, 2009.

[MMC09b] Christoph Minnameier and Mila Majster-Cederbaum. Cross-Checking – Enhanced Over-Approximation of the Reachable Global State Space of Component-Based Systems. In *Proceedings of RP'09*, volume 5797 of *LNCS*, pages 189–202. Springer, 2009.

[MN06] Guido Moerkotte and Thomas Neumann. Analysis of two Existing and one new Dynamic Programming Algorithm for the Generation of Optimal Bushy Join Trees without Cross Products. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 930–941. VLDB Endowment, 2006.

[Moo56] Edward F. Moore. Gedanken Experiments on Sequential Machines. In *Automata Studies*, pages 129–153. Princeton U., 1956.

[MPW92a] Robin Milner, Joachim Parrow, and David Walker. A Calculus

of Mobile Processes, I. *Inf. Comput.*, 100(1):1–40, September 1992.

[MPW92b] Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, II. *Inf. Comput.*, 100(1):41–77, September 1992.

[Osa12] Ammar Osaiweran. *Formal Development of Control Software in the Medical Systems Domain*. Phd thesis, University of Technology Eindhoven, Eindhoven, Netherlands, 2012.

[Pel93] Doron Peled. All from One, One from All: on Model Checking Using Representatives. In *Proceedings of the 5th International Conference on Computer Aided Verification, Greece*, number 697 in Lecture Notes in Computer Science, pages 409–423, Berlin-Heidelberg-New York, 1993. Springer.

[Pet67] C.A. Petri. *Grundsätzliches zur Beschreibung Diskreter Prozesse*. Uddrag af ISNM Vol.6, 1967 - 3. Colloquium über Automatentheorie. Rheinisch-Westfälisches Institut für, Instrumentelle Matematik, 1967.

[PH98] Abelardo Pardo and Gary D. Hachtel. Incremental CTL Model Checking Using BDD Subsetting. In *Proceedings of the 35th annual Design Automation Conference*, DAC '98, pages 457–462, New York, NY, USA, 1998. ACM.

[PL08] M. Praveen and Kamal Lodaya. Analyzing Reachability for Some Petri Nets With Fast Growing Markings. *Electron. Notes Theor. Comput. Sci.*, 223:215–237, December 2008.

[Pnu77] Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer*

Science, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[Pnu79] Amir Pnueli. The Temporal Semantics of Concurrent Programs. In *Proceedings of the International Sympoisum on Semantics of Concurrent Computation*, pages 1–20, London, UK, UK, 1979. Springer-Verlag.

[Pri57] Arthur N. Prior. *Time and Modality.* Oxford University Press, 1957.

[PT87] Robert Paige and Robert E. Tarjan. Three Partition Refinement Algorithms. *SIAM J. Comput.*, 16(6):973–989, December 1987.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and Verification of Concurrent Systems in CESAR. In *Symposium on Programming*, pages 337–351, 1982.

[RJB99] James Rumbaugh, Ivar Jacobson, and Grady Booch, editors. *The Unified Modeling Language Reference Manual.* Addison-Wesley Longman Ltd., Essex, UK, UK, 1999.

[Sav70] Walter J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, April 1970.

[Sch02] Ph. Schnoebelen. The Complexity of Temporal Logic Model Checking. In *Advances in Modal Logic*, pages 393–436, 2002.

[Sif05] Joseph Sifakis. A Framework for Component-based Construction Extended Abstract. In *Proceedings of the Third IEEE International Conference on Software Engineering and Formal*

Methods, SEFM '05, pages 293–300, Washington, DC, USA, 2005. IEEE Computer Society.

[Tan07] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2007.

[TBEKR03] Maurice H. Ter Beek, Clarence A. Ellis, Jetty Kleijn, and Grzegorz Rozenberg. Synchronizations in Team Automata for Groupware Systems. *Comput. Supported Coop. Work*, 12(1):21–69, February 2003.

[VU98] Moshe Y. Vardi and Rice University. Linear vs. Branching Time: A Complexity-Theoretic Perspective. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*, LICS '98, pages 394–, Washington, DC, USA, 1998. IEEE Computer Society.

[VW86] Moshe Y. Vardi and Pierre Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *Proceedings of the 1st Annual Symposium on Logic in Computer Science (LICS'86)*, pages 332–344. IEEE Comp. Soc. Press, June 1986.

[vW08] Muck van Weerdenburg. Process Algebra with Local Communication. *Electron. Notes Theor. Comput. Sci.*, 215:191–208, June 2008.

[WRS+08] Andreas Weidemann, Stefan Richter, Matthias Stein, Sven Sahle, Ralph Gauges, Razif Gabdoulline, Irina Surovtsova, Nils Semmelrock, Bruno Besson, Isabel Rojas, Rebecca Wade, and Ursula Kummer. Sycamore – a systems biology computational analysis and modeling research environment. *Bioinformatics*, 24(12):1463–1464, June 2008.

[Yan81] Mihalis Yannakakis. Algorithms for Acyclic Database Schemes. In *Proceedings of the seventh international conference on Very Large Data Bases - Volume 7*, VLDB '81, pages 82–94. VLDB Endowment, 1981.

[YO79] C.T. Yu and M.Z. Ozsoyoglu. An Algorithm for Tree-Query Membership of a Distributed Query. In *Computer Software and Applications Conference, 1979. Proceedings. COMPSAC 79. The IEEE Computer Society's Third International*, pages 306 – 312, 1979.

Index