Establishing Properties of Interaction Systems

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

vorgelegt von
Moritz Martens
aus Köln

Mannheim, November 2009

Dekan:	Professor Dr. Felix Freiling, Universität Mannheim
Referentin:	Professor Dr. Mila Majster-Cederbaum, Universität Mannheim
Korreferent:	Professor Dr. Felix Freiling, Universität Mannheim

Tag der mündlichen Prüfung: 18. Dezember 2009

Abstract

We exhibit sufficient conditions for generic properties of component based systems. The model we use to describe component based systems is the formalism of interaction systems. Because the state space explosion problem is encountered in interaction systems (i.e., an exploration of the state space gets unfeasible for a large number of components), we follow the guideline that these conditions have to be checkable efficiently (i.e., in time polynomial in the number of components). Further, the conditions are designed in such a way that the information gathered is reusable if a condition is not satisfied.

Concretely, we consider deadlock-freedom and progress in interaction systems. We state a sufficient condition for deadlock-freedom that is based on an architectural constraint: We define what it means for an interaction system to be tree-like, and we derive a sufficient condition for deadlock-freedom of such systems. Considering progress, we first present a characterization of this property. Then we state a sufficient condition for progress which is based on a directed graph. We combine this condition with the characterization to point out one possibility to proceed if the graph-criterion does not yield progress. Both sufficient conditions can be checked efficiently because they only require the investigation of certain subsystems. Finally, we consider the effect that failure of some parts of the system has on deadlock-freedom and progress. We define robustness of deadlock-freedom respectively progress under failure, and we explain how the sufficient conditions above have to be adapted in order to be also applicable in this new situation.

Zusammenfassung

Wir präsentieren hinreichende Bedingungen für allgemeine Eigenschaften von komponentenbasierten Systemen. Als Formalismus zur Modellierung komponentenbasierter Systeme nutzen wir Interaktionssysteme (*interaction systems*). Da für Interaktionssysteme das Problem der Zustandsraumexplosion auftritt (d.h. für eine große Anzahl von Komponenten ist eine Analyse des gesamten Zustandsraums nicht durchführbar), sollen die Bedingungen effizient überprüfbar sein (d.h. mit polynomiellem Aufwand in der Anzahl der Komponenten). Des Weiteren sind die Bedingungen so geartet, dass die gesammelte Information wiederbenutzbar ist, falls eine Bedingung nicht erfüllt sein sollte.

Im Einzelnen betrachten wir Deadlockfreiheit und Fortschritt in Interaktionssystemen. Wir formulieren eine hinreichende Bedingung für Deadlockfreiheit, welche auf einer Einschränkung an die Kommunikationsarchitektur basiert: Wir definieren baumartige Interaktionssysteme und leiten dann eine hinreichende Bedingung für die Deadlockfreiheit solcher Systeme her. Im Hinblick auf Fortschritt geben wir zunächst eine Charakterisierung dieser Eigenschaft an. Dann formulieren wir basierend auf einem gerichteten Graphen eine hinreichende Bedingung für Fortschritt. Wir kombinieren diese Bedingung mit der Charakterisierung, um einen Ausweg aufzuzeigen, falls das Graphkriterium nicht erfüllt ist. Beide hinreichenden Bedingungen können effizient überprüft werden, weil nur die Analyse bestimmter Teilsysteme erforderlich ist. Schließlich untersuchen wir noch die Auswirkungen, welche der Ausfall bestimmter Teile eines Systems auf Deadlockfreiheit und Fortschritt hat. Wir definieren zunächst Robustheit von Deadlockfreiheit beziehungsweise Fortschritt unter Ausfall. Dann erklären wir, wie die obigen hinreichenden Bedingungen angepasst werden müssen, damit sie auf diese neue Situation angewendet werden können.

To Hennes VII

Ruhe in Frieden, Unglücksbock!

Contents

Acknowledgments

First, I am grateful to Professor Mila Majster-Cederbaum for introducing me to this new area of research and for being a great supervisor. She was always approachable, helping me with guidance, advice, and encouragement.

I also thank the second referee Professor Felix Freiling. He took the time to discuss the thesis with me in great detail, giving me valuable feedback.

Finally, I thank all my colleagues on the corridor — I had a (X)Blast with you. In particular, my fellow co-workers Christoph, Nils, and Christian made my time very enjoyable. Special thanks go to Nils and Christian for proof-reading countless pages and spending a lot of time listening to my results.

Chapter 1

Introduction

1.1 Component Based Systems

With the large variety of communicating parallel systems occurring in all kinds of areas — which of course include but are by far not restricted to computer science — concurrency (theory) has developed into a central branch of scientific research. Dating back its emergence to works by Petri [117] and by Dijkstra [59], there are now scores of different formalisms providing the means to specify and investigate parallel systems[1].

Being one particular field of research in concurrency, component based systems or simply component systems have emerged as a more recent area of increasing significance. Component based design and engineering have proved an important paradigm to master design complexity and reusability (of parts) of parallel systems. There is a collection of component based technologies (cf. JavaBeans [4, 105], the Microsoft Component Object Model [2], or CORBA [3], for example[2]) all using component related concepts and notions in one way or the other without rigorously defining what a component respectively a component based system really are, not to mention agreeing on common definitions and terminology. The mere existence of these technologies and the resulting ambiguities about the notion of a component necessitate a firm theoretical foundation which allows to model and investigate systems in such technologies. In order to get such a foundation, it is first of all important to agree upon an understanding of what a component

[1] Cleaveland and Smolka [53] give a more comprehensive survey of the various theories.

[2] Lau and Wang [94] give a more comprehensive survey of the technologies in use.

is. As noted above, in the various technologies there have been different conceptions of what a (software) component should be, but it is widely agreed that components are individual entities providing certain services to their environment. Further, since the need to enhance the reusability of parts of distributed systems is one motivation for component based design, components should be as independent as possible from their context of use. Ideally, a component may be deployed in any meaningful environment that can use the component's functionality. Contrary to other approaches, a component is in particular not allowed to invoke operations or access data of other components directly. It should be made sure that the communication between the individual components of a system is *not* intrinsically linked to their respective behavior. Instead, a component should hide as much of its internals as possible from the environment. It should only provide an interface consisting of ports which can be used by the environment to access the services the component offers. These ports can be used to glue several components together by means of an *external* gluing mechanism which is separately adapted to each concrete setting a set of components is used in.

Combining such common aspects in the requirements imposed on a component, Messerschmitt and Szyperski [106] list the following characterizing properties of a component:

- It should be reusable in multiple projects.
- It should be independent of the context it may be used in.
- It should be composable with other components.
- It should be encapsulated, i.e., the environment should be able to access the services a component provides through the component's interface but it should not be able to access or even change the internals of the component through this interface.
- It should be a unit of independent deployment and versioning. In particular, replacement or upgrade of single components in a given system should be possible independently of the other components.

These characteristics are not mutually exclusive, and they can be interpreted to various extents. Regarding composition of components, for example, one

can choose a point of view towards hierarchical composition (composing several components yields a new component, the identities of the original components are lost) vs. parallel composition (composing several components yields a component system which maintains the identity of the original components). On the other hand, the rigor of encapsulation that is reasonable in practice depends on the scenario the components are deployed in, what properties should be guaranteed for the composition of these components, and so forth. Despite the various possibilities to realize these items we want to agree that they should be required at the least.

Having chosen this kind of global characterization of a component, it is now possible to develop formalisms which can be used to design and model distributed systems constructed from components as well as to specify and verify their properties. The fact that the defining characteristics of a component may be interpreted differently has lead to a wide range of formalisms for component based systems, each one adequately tailored to fulfill a certain purpose. There have been attempts to compare these models by applying them to a common benchmark example in order to see how they cope with standard problems encountered in component based design and to ultimately find or develop a unified component model. Most recently there has been the CoCoME[3] contest [82], for instance. Without going into the details of the conclusions drawn, we substantiate the necessity — despite the variety of existing models — for further research towards a well-founded formalism of component based systems and in particular towards feasible ways to establish properties by shortly quoting the final verdict of the CoCoME jury [40]: *"[...], the gap between practical applicability and demonstrated modeling capability remains significant, leaving much room for further work."* This thesis is concerned with establishing properties of component based systems.

Gössler and Sifakis [70–72] introduced one formalism for component systems — namely *interaction systems*. The formalism has been devised with the objective to closely follow the characteristics listed for a component above. Most notably, the formalism strictly separates the description of the glue-code, i.e., of the communication, from the description of the behavior of

[3]Common Component Modeling Example

the components. These issues are defined in two independent layers that are orthogonal to each other in the sense that each layer can be modified easily without touching the other. Because the formalism is fairly general we feel that it is well suited to model and investigate component systems on a level of abstraction which is initially concerned with questions about component interoperability rather than verification of concrete specifications. One issue occurring at this level of abstraction is the investigation of *generic* properties such as deadlock-freedom or progress of a component. Based on interaction systems we investigate how such generic properties can be established *without* analyzing the global system. It should be noted that the model can be refined in order to also account for the investigation of concrete functional properties once the properties above have been dealt with.

1.2 Establishing Properties — Goal of this Thesis

Before going into the details of how we go about establishing deadlock-freedom and progress in interaction systems we review some methodologies that have been conceived for establishing properties. We take a more general perspective by discussing these not in the context of component systems but in the context of concurrent systems and concurrent programs.

1.2.1 Methodologies

In order to be able to check general properties for concurrent systems, it is first necessary to be able to specify properties and to state when they are satisfied by a system. The main approaches for specification of properties use temporal logics such as LTL [120], CTL [49], or the μ-calculus [91], respectively behavioral relations like bisimulation or observational equivalence as discussed by Milner [107]. In the latter approach the same formalism has to be used for the description of the system and of the specification. Equivalence of the two assesses that the model of the system is indeed a correct implementation of the specification. In the temporal logic approach specific properties are formulated in the corresponding logic which can then be checked against the description of the system in the chosen formalism. Both

approaches are amenable for algorithmic verification in the finite state case. Model checking [49, 51] allows for an automatic check whether a given system satisfies a property expressed in a temporal logic. Algorithms have also been conceived for the automatic computation of various equivalences. We refer to the algorithms of Kanellakis and Smolka [87] and of Paige and Tarjan [114], for example, which can be used to compute the bisimulation quotient of a transition system. Cleaveland and Sokolsky [54] give a more detailed survey of results concerning the establishment of various equivalences. The important point to note is the fact that for both model checking and equivalence checking the size of the system to be treated appears as a factor in the complexity of the algorithms. Since any formalism for concurrent systems has to cope with the state space explosion problem it becomes clear that the feasibility of these approaches is limited. For interaction systems this "infeasibility" has been formalized by showing that deciding deadlock-freedom and progress, for example, is PSPACE-complete [99, 101, 128].

There are various techniques that can be applied to diminish the impact of the state space explosion problem. Such techniques may work very well for the reduction of the relevant state space of a concrete system and therefore may be of benefit if they are combined with model checking, for instance. However, the complexity result stated above shows that there will always be systems that cannot be handled if we try to establish a property in interaction systems directly. The helpfulness of these techniques may depend on certain structural characteristics of the system. For example, if the system exhibits a high degree of asynchrony, i.e., there is a lot of interleaving, then partial order reduction (cf. Godefroid [67] or Peled [116]) is one way to try to substantially reduce the state space. The idea of partial order reduction is to identify so-called independent actions which are actions whose order of execution does not have any effect on the resulting state. For such actions it suffices to only investigate one thread of execution as opposed to all threads resulting from the possible orderings of the actions. Other measures that may help to reduce the state space that has to be investigated exploit symmetries of a system [50, 63], compositionality [52] (i.e., one tries to combine properties that have been established for subsystems by means of model checking, for example, in order to argue about a global property),

or one might even use behavioral equivalences as above in order to obtain a compactified system which abstracts from the details that are not relevant for the respective question. More generally, abstraction techniques aim at reducing the part of the a system or program that has to be investigated with regard to a certain problem, and they may be combined with various verification methods: The goal is to derive from a concrete semantics an abstract semantics which is more simple and therefore easier to handle on the one hand, but which should convey enough information such that the problem in question can still be answered. Abstract interpretation as introduced by Cousot and Cousot [55] is one approach which formalizes such program abstraction and analysis methods by means of Galois connections between partially ordered sets. These sets represent the semantics (or interpretation) of a program and its abstraction(s). A notion of consistency of an abstract interpretation is given with respect to a concrete one. Using arguments about fixed points in complete lattices, information about program behavior is obtained. This approach can be seen as a unification of various other abstraction and analysis methods. We refer to Cleaveland and Smolka [53] for a more thorough review of the methodologies above and to Baier and Katoen [23] for an elaboration of the details.

The methodologies above deliver a definitive answer to the question whether a given (abstract) model of a system has a property/implements a specification or not. However, it is advisable to sound a note of caution with regard to what this means for more concrete models respectively the system itself. Such an answer holds for the system *model* that has been investigated, and if abstraction techniques have been used to obtain this model (which is almost always the case because a description of a system which takes into account all details cannot be handled) it has to be clarified if information about the property in question carries over to a more concrete semantics. In the case of abstract interpretation as mentioned above, for example, a positive answer to the question whether a property is satisfied by an abstraction of a system carries over to more concrete semantics. If it turns out that the abstraction does not have the property, it is in general not possible to derive information regarding the question whether more concrete models have the property or not. In such cases where the chosen model proves to be too

abstract, i.e., no information can be derived for more concrete semantics or the actual system, the model must be *refined*[4].

Thus, it is not always possible to derive information about the presence of a property in a concrete system by means of investigating abstractions of the system — even if it is possible to give a definitive answer for such an abstraction. Because of this observation it is possible to conceive different methodologies which desist from the claim that at the level of a given abstraction a definitive answer has to be given. These methodologies yield correct answers in some cases (from which information about the concrete system can be derived). On the other hand, for such an approach it may also be possible that the answer is "don't know", i.e., no information is provided as for whether the abstraction or even the system itself has the property in question. Settling for such an approach, the disadvantage of not being able to provide any answer in some cases can be compensated by the fact that such approaches may be checkable very efficiently. In view of the complexity results stated above and the state space explosion problem this is not an unattractive option at all. Since model checking or equivalence checking as above will become infeasible and therefore will *definitely* not provide any answer for increasingly large systems, more efficient albeit incomplete procedures present an alternative which should be taken into account. At worst, we do not know anything after the application of the procedure, and we can still apply model checking, for example. There are various methodologies towards establishing properties which fall into this category. They of course differ in the way how they go about establishing properties. However, they also fundamentally differ in *what* kind of answer they can possibly convey:

1. (Software) testing [111], on the one hand, is an approach which tries to find errors in a program or system respectively in parts of a program or system at the stage where they are (almost) ready to use. By testing it for a number of inputs or more formally test cases [122], which consist of an input and an expected output, the goal is to find possible errors in the implementation. This approach *cannot* prove that a system is correct with respect to a specification or property (unless it uses an

[4]Refinement can be understood to be the counterpart of abstraction.

exhaustive set of test cases which is not feasible again). Even if no considered test case violates the specification or property there might be other test cases for which this is the case. Only if a test yields a not intended result a definitive (negative) answer can be given. Ideally, in this case the test case should also provide information about the nature of the violation and where possible corrections could be necessary.

The various testing methods are usually categorized as black box testing or white box testing[5] where the difference lies in the knowledge the tester has about internal details of the implementation [122]. There are scores of different testing procedures and approaches, where amongst other issues particular importance is granted to the generation of an adequate set of test cases. In the context of this thesis, we only further mention model-based testing [39] because it uses formal models such as automata, for example, for an automatic generation of test cases. Since testing is deployed on an actual implementation of a system it should be noted that the ambiguities arising from the use of an abstraction of a concrete system mentioned above do not occur. A fault detected during the process of testing is an actual fault in the system.

2. If sufficient conditions for a property are to be exhibited a definitive answer is only possible if the system satisfies the condition. Otherwise, one might be tempted to interpret this as an indicator that the system does not have the property. However, in this case no information about the presence or absence of the property can be derived.

   There is no general or even best way how to develop sufficient conditions for a property. The details may significantly differ depending on the underlying formalism, the approach, and even on the property itself. For example, it is possible to consider a situation where a property is violated and to derive a *necessary* condition for such a violation. Negating this condition, then yields a sufficient condition for the property. Approaches following this idea have been presented by Attie and Emerson [21] and by Attie and Chockler [20] with respect to deadlock-freedom, for example. On the other hand, it is also promising

[5] There are gray box approaches combining aspects of the two, though.

to include structural information into the considerations about sufficient conditions. This can be the information that the communication architecture has a certain structure (e.g., it is given by a tree) [33, 79] or that the maximal branching degree of any state of any process is bounded by a fixed number [21], for example. Fairly often, the conditions are based on information derived from analyses of subsystems, i.e., abstraction is realized by hiding processes [84, 100, 102] (also compare the approaches above). Note, though, that other abstraction methods can be used, as well. Gössler and Sifakis [70], for example, try to establish deadlock-freedom in interaction systems based on a cycle analysis in a certain graph which constitutes a simplified representation of the possible dependencies between the components.

There are further methodologies (e.g., formal systems consisting of axioms and inference rules as discussed by Apt et al. [12] and Francez [64] in the context of CSP, for example) to argue about correctness and properties of systems. However, at this point we conclude the discussion and explain which methodology we choose to follow in this thesis.

1.2.2 Goal of this Thesis

This thesis is supposed to advance the understanding as well as the practical usefulness of interaction systems. As stated above we investigate deadlock-freedom and progress. These properties are clearly desirable and important in systems design. Moreover, deadlock-freedom is of interest because checking regular safety properties can be reduced to checking deadlock-freedom [68]. On the other hand, the complexity results above show that dealing with these properties has to be handled with care. Just like in other formalisms for concurrent systems, the above-mentioned state space explosion problem is encountered in interaction systems, and therefore a direct check of the properties by analyzing the global state space is not feasible. Indeed, the complexity results show that any approach trying to present an *efficient* decision-procedure for one of the properties which works for *all* systems is doomed to failure. In this thesis we want to tackle the complexity issues by following the last methodology presented above, i.e., we want to

exhibit sufficient conditions for the properties. In the following we establish a few paradigms that we want to follow in the search for such conditions:

- Since we settle for *sufficient* conditions for a property this "incompleteness" should be made up for by the fact that the conditions can be checked *efficiently*[6].
- We want to accomplish the conditions in a compositional manner. For example, we only want to consider parts of the system by hiding components and combine the information gathered from the subsystems to obtain a statement about a property of the global system. This way we make sure that the conditions can indeed be checked efficiently.
- We want to formulate these conditions in such a way that the information necessary for checking them has not been gathered in vain if the condition is *not* satisfied, even though the conditions are only sufficient: In this case we want to be able to draw conclusions about further steps that can be taken to detect the property in question.

The reason we follow the methodology of exhibiting sufficient conditions is twofold: First, model checking has been thoroughly investigated and is well understood. If no other technique is at hand it can be consulted to in interaction systems just like in most other formalisms for concurrent systems. We do not feel it would be too much of a contribution to investigate model checking in the context of interaction systems. Second, the strict separation of behavior and communication in interaction systems has proved to be very suitable for stating sufficient conditions. Once certain characteristics of a property have been identified they can be tackled on each layer of description of interaction systems independently without having to touch the other.

1.3 Other Approaches Towards Component Systems

We conclude by giving a short review of some of the models that are used for the investigation of component based systems. For some of these

[6]Here "efficiently" means "in time polynomial in the number and size of the components and the size of the glue-code".

models the main purpose consists of providing a means to support systems design and to allow for a convenient way to describe and discuss component systems. Others also emphasize the investigation of various properties. We shortly discuss these in the context of the methodologies considered above.

Reflecting the condition that the internals of a component should not be accessible directly from the outside, most formalisms have in common that components are equipped with some kind of interface consisting of ports which can be used to access the services a component offers. The interfaces can be used to combine various components yielding complex systems. In order to make allowance for a high degree of encapsulation there is a range of approaches which consider a component as a black box that does not make any specification about the internal behavior of the component [32, 41, 48, 109]. A component only provides the I/O behavior which is *externally* observable for each port. The approaches differ in how the ports' behaviors are specified (using pre- and postconditions or data-streams, for instance) and what kind of statements are possible about the constructs obtained from composing components. Such approaches allow for automatic reasoning about component compatibility or about properties specified on the data-streams. However, for the properties, that we are interested in, this high degree of encapsulation is not practicable. For example, it does not suffice to compare the execution traces of communicating components to exclude the possibility of deadlocks [107, Chapter 9.4]. Thus, for the purpose of analyzing such properties the model used to describe components must also provide information about the internal behavior of each component.

There is a variety of proposals on how to incorporate this information into the formal description of the components. There have been approaches (cf. Bastide and Barboni [28] and Aoumeur and Saake [11], for example) to describe the local behavior of components using Petri nets. Petri nets are convenient to describe concurrent processes because they convey an intuitive graphical idea[7] of the system's structure and behavior without neglecting the requirements for a well-grounded formalism. Furthermore, it is possible to formulate and investigate a range of interesting properties, and Petri nets

[7] An intuitive access to a formalism is a convenient feature not least with regard to using it as a basis for systems design.

allow for formal verification by means of model checking. However, in our view Petri nets are suitable to only a limited extent as a formal model for component based systems owing to the fact that there is no adequate *general* composition operator for Petri nets. This is a drawback because one of the defining properties of a component named in Section 1.1 is composability. Thus, any formalism modeling the components by Petri nets must provide explicit rules on how to construct the Petri net describing the global system from the nets of the single components respectively subsystems. Such constructions can be rather complicated: For the definition of the net describing a composed system Bastide and Barboni [28], for example, have to fall back to an extension of Petri nets causing various properties to become undecidable. In addition, explicitly constructing a Petri net for the global system has the disadvantage that the identities of the single components are lost, i.e., there is no canonical way to relate the places and transitions of the global net to the net of the component they came from.

Since component based systems are concurrent systems after all it makes sense to use the existing and well-understood formalisms that are based on process algebra and automata respectively labeled transition systems to describe the (branching) behavior of a component. This way of describing the behavior of the components has its strength in the fact that process algebras have been thoroughly investigated. They provide a means to describe the branching behavior of concurrent processes, and they allow for an easy and intuitive way for parallel composition of processes. There is a range of behavioral equivalences which are — as mentioned in the previous section — an important instrument with regard to showing that a system conforms to a specification. Furthermore, transition systems are amenable to model checking. The latter two features are indeed extensively used for the verification of component systems in models based on process algebra or automata.

There are various component models which are based on process algebras such as CCS [107], CSP [83], or the π-calculus [108], or on combinations respectively subclasses of these. The models falling into this category [e.g., 8, 33, 38, 84, 85, 97, 112] differ in the calculus that is used and in the details of *which* parts of the components' behavior are modeled. Interestingly,

though, quite a few of these models try to establish deadlock-freedom of a composed system (respectively of the construct obtained by gluing several ports together in the case of Allen and Garlan [8]) by following the methodology of presenting sufficient conditions. These conditions are based on various ideas. On the one hand, Brookes and Roscoe [38] derive deadlock-freedom results from constraints on the communication architecture of the system. Similarly, Bernardo et al. [33] also put restrictions on the architecture (the authors consider star topologies and rings of components) and additionally check behavioral equivalences for systems resulting from certain subsets of components. A (partial) behavioral equivalence between pairs of components is also defined and used by Inverardi and Uchitel [84]. Roughly speaking, this equivalence makes sure that the components' behaviors are compatible until they do not require cooperation with each other any more. Finally, Allen and Garlan [8] derive deadlock-freedom from conditions that can be checked on the glue-code used to combine ports of several components. These approaches aim at conditions whose check is not as complex as an analysis of the global system. Despite the fact that process algebras have been thoroughly investigated and that there exists a great variety of formal tools which can be used to verify process properties in such algebras, they also have a drawback when it comes to modeling components: In general, process algebras do not cleanly detach synchronization between the processes from their behavior. Considering CSP, for example, the possible candidates for synchronization over an action a are predetermined by the name of the action. Synchronization is always accompanied by renaming of actions of the respective processes (which constitutes an interference with the behavior of the processes).

There are numerous models which use slightly more general notions of automata or labeled transition systems to describe the components[8]. To begin with, there are I/O-automata [96], interface automata [56], and team automata [62, 134]. Based on these formalisms there are various other approaches towards providing components with an internal behavior (cf. Baumeister et al. [30] and Hennicker et al. [79] based on interface

[8]Not all of these models have been explicitly designed to model component systems. Nonetheless, they are suitable to be used in this context, as well.

automata or the CoIn approach [37, 46] based on team automata, for example). The SOFA component model and its extension SOFA2.0 [45, 119] use behavior protocols [90, 118] to describe the behavior of the components. Note that the above-mentioned model FRACTAL [41] has been extended by behavior protocols to also allow for specification of component behavior (cf. Bulej et al. [44]). Furthermore, there is the Vereofy setting [6] which extends the Reo coordination language [13, 14] by constraint automata [15] in order to describe the components' behavior (and in fact also the behavior of the glue-code). Bergner et al. [32] suggest state transition diagrams as another possibility to describe the I/O-behavior of a component. We conclude by mentioning that there are various further ways to use automata to provide components with an internal behavior, for example, statecharts [78] from which the UML state machine diagrams [5] have been derived.

Not all of the above-mentioned models put the main focus on the investigation of properties (in particular, those properties we are interested in). It is clear, though, that model checking is a valid option. Indeed, some of the models above have been formulated and applied in the context of certain model checking tools. This is the case for the CoIn approach [137] towards the CoCoME project using the DiVinE model checker [25] and for the Vereofy setting which provides a model checker[9] of its own, for example. On the other hand, Lynch and Tuttle [96] use compositional proof rules in the context of I/O-automata in order to allow for arguments about composed systems without having to investigate the global state space. Furthermore, there also are results on testing of I/O-automata [136]. Finally, some authors, e.g., Hennicker et al. [79], provide sufficient conditions for properties using architectural constraints. This approach has also been taken in the context of establishing deadlock-freedom in interaction systems [70, 100, 102]. Note that Attie and Chockler [20] use similar ideas as the latter two works in a setting for general concurrent systems.

It remains to be said that the above review of component models is far from complete. We mainly focused on such models where similar questions

[9]This model checker features Binary Decision Diagrams (BDDs) [36, 42, 43] as the underlying data structure of the description of the automata. BDDs allow for an efficient representation and manipulation of the automata.

to the ones we will treat here arise. Needless to say, there are numerous additional interesting properties which have been formulated and investigated in other component models. Each of these models exhibits various strengths with regard to the concrete problem to be treated, and it is difficult to relate these to the problems we treat in the context of interaction systems. Exemplarily, we say a few words about some other approaches towards component based systems and the properties that have been investigated. There are various models such as the KobrA component design method [18, 19] which set greater store by the actual modeling and design of a system than by analysis and verification of system properties. Such models are supposed to provide a common basis for the discussion of questions arising in the development process of a system. On the other hand, there are approaches, e.g., the Palladio Component Model [123], providing a means to investigate *non-functional* properties (performance, reliability, resource management, etc.) which exceed the requirements put upon a component system by a specification. In particular, with regard to application of components it can be essential to also be able to evaluate the quality of a system with respect to such issues. The various approaches mentioned so far have one thing in common: They are all based on a *discrete* setting, i.e., each component has countably many (typically finitely many) states and actions. This suffices to treat a great variety of situations. In contrast, Masaccio [80] is a model for component systems that allows for continuity by letting the states of the components range over the real numbers, while the components' behavior is described by differential equations. Such an approach offers a whole new field of possibilities which cannot be handled in any discrete setting.

1.4 Thesis Outline

The thesis consists of 6 chapters. Chapters 3, 4, and 5 are followed by an appendix containing the proofs of the results presented in the chapter. There is only one exception to the convention of moving the proofs to the end of each chapter. We state one proof right away. It is a constructive proof whose construction is necessary to understand the arguments to follow.

- Chapter 2 contains the definitions of the basic notions with respect to

interaction systems. We define deadlock-freedom and progress. The chapter also presents an example modeling the dining philosophers as an interaction system.

- Chapter 3 uses architectural constraints in order to derive deadlock-freedom results for a subclass of interaction systems. We first deal with reachability in interaction systems and we explain how the reachability of given combinations of states may be excluded by only analyzing subsystems of size two. We define the interaction graph of an interaction system which we require to be a tree. This constraint is used to identify pairs of states that may cause global deadlocks. We combine these results with the considerations about reachability in order to exclude the possibility that combinations of such pairs are in fact reachable. We simplify the results for systems obeying to further restrictions. We discuss three complex examples.

- In Chapter 4 we consider progress of a set of ports. We first present a characterization of all sets of ports that make progress in a given system. Thereafter, we turn to the question of how progress can be detected in practice. We state a sufficient condition for progress which is based on a graph. The preciseness of the criterion can be controlled according to a parameter d which is incorporated into the construction of the graph. This construction can be achieved efficiently where the degree of efficiency is governed by d. By discussing situations where the criterion is not satisfied we motivate how the information gathered from the graph can be combined with other criteria. Finally, we use the results to show that every philosopher makes progress in the system modeling the dining philosophers introduced in Chapter 2.

- In Chapter 5 we consider situations where certain components respectively ports may fail during the execution of the system. We investigate how properties of a system are influenced by this failure. We define robustness of deadlock-freedom with respect to failure of a set of ports respectively progress without a set of ports. Next, we show that the approaches chosen towards deadlock-freedom and progress in

the previous two chapters can be adapted to also account for this new situation. We obtain results which adjust the criteria presented in these chapters to systems where ports may fail.

- Chapter 6 concludes the thesis. We review the results and classify them with regard to other methodologies and the question in what ways the challenge has been met. We point out directions for future work.

Chapter 2

Interaction Systems

In this chapter we state the definitions concerning the formalism of *interaction systems*. All results in this thesis are based on these definitions. We first define interaction systems by giving the two layers that an interaction system is composed of. We continue by defining deadlock-freedom and progress in interaction systems. This chapter is a collection of the notions introduced by Gössler, Sifakis, and others [71–74, 102] although it should be noted that we do not always stick to the nomenclature chosen there. Furthermore, in some regards we consider slight generalizations of the notions introduced in these papers. Nonetheless, interaction systems originate from these sources (in particular from the works by Gössler and Sifakis [71, 72]). Readers who are familiar with interaction systems may want to skim through this chapter in order to get accustomed to the notations we use in this thesis and use it as reference for clarification of terms occurring in the following chapters.

Before going into the details note that the BIP- [29] and the PROMETHEUS-tool [69] implement data structures which are based on interaction systems. Furthermore, the model is used as a common formal platform in the EU project SPEEDS [22, 35].

2.1 Components, Interactions, and Interaction Systems

There are two layers of description for an interaction system. We will first define an *interaction model* which constitutes the *static* layer of description of an interaction system. An interaction model consist of a *component system*

and an *interaction set*. The component system comprises the names of the components and their port sets. The interaction set defines the "glue-code" used to stick the components together. It is given by so-called interactions which are sets of ports such that each component participates with at most one port[1]. Intuitively, an interaction α describes a possible cooperation of the participating components over the ports contained in α.

Definition 2.1.1. *A **component system** is a pair $CS = (K, \{\mathcal{A}_i\}_{i \in K})$. Here $K = \{1, 2, \ldots, n\}$ is a finite set of **components**. We usually use variables $i, j, k, \ldots$ to denote the components. $\mathcal{A}_i$ denotes the **port set** of component $i \in K$. We assume that the port sets are nonempty, finite, and pairwise disjoint. We usually denote the elements of $\mathcal{A}_i$ by $a_i, b_i, c_i, \ldots$ and call them **ports** or **actions**. The union $\mathcal{A} = \bigcup_{i \in K} \mathcal{A}_i$ of all port sets is called the **port set** of CS.*

*A nonempty subset $\alpha \subseteq \mathcal{A}$ is called an **interaction** of CS if $|\alpha \cap \mathcal{A}_i| \leq 1$ for all $i \in K$. We usually use variables $\alpha, \beta, \gamma, \ldots$ to denote interactions. For $i \in K$ and interaction α we define $i(\alpha) := \mathcal{A}_i \cap \alpha$, and we say that i **participates** in α if $i(\alpha) \neq \emptyset$. An **interaction set** (for CS) is a set Int of interactions of CS that covers all ports, i.e., $\bigcup_{\alpha \in Int} \alpha = \mathcal{A}$. The function $comp : Int \to 2^K$ where $comp(\alpha) := \{i \in K | i(\alpha) \neq \emptyset\}$ yields the components that participate in α. We say that two components i and j, $i \neq j$, **interact** if there is an interaction $\alpha \in Int$ with $i, j \in comp(\alpha)$. For interacting components i and j we define the set of ports of i that are **used for communication with** j as $comm_i(j) := \{a_i \in \mathcal{A}_i | \exists \alpha \in Int$ with $a_i \in \alpha$ and $j(\alpha) \neq \emptyset\}$. Further, let $i \in K$ be a component and let $Int' \subseteq Int$ be a subset of interactions. By $Int'(i) := \{\alpha \in Int' | i(\alpha) \neq \emptyset\}$ we denote the set of interactions in Int' that i participates in.*

*The pair $IM := (CS, Int)$ is called an **interaction model**.*

At this level of description we still completely abstract from the local behavior of the components, but even at this level the same set of components may yield various component systems because the components can be

[1]This requirement makes sure that no component ever has to perform a self-synchronization which is a natural assumption if one considers ports to be atomic actions which can be executed by a component one at a time.

glued together using different sets of interactions. Thus, interaction systems adhere to one of the characteristics, namely context independence, required for components in Section 1.1. By separating the glue from the specification of the behavior of a component, which will be given by the second layer of description, we ensure that each component is as independent from the context it is supposed to be used in as possible. For many formalisms behavior and communication are *not* cleanly detached from each other. This is in particular the case if for a given action those actions of other components that are possible candidates for communication are predetermined by the name of the action: If communication consists of renaming *before* synchronization then this separation is not enforced (cf. CCS [107], for example).

Figure 2.1 below indicates how an interaction model can be depicted graphically. We represent components by large white squares whereas the ports are given by small black squares on the edges of the corresponding component. An interaction is depicted as a line connecting the ports it contains. In the following chapters we do not build on such graphical representations which is why we will not go into the details of depicting the interaction models of upcoming examples. It should be borne in mind, though, that the notion of interaction model allows for an intuitive access which is, as stated in Chapter 1, a convenient feature in systems design.

Remark 2.1.1. *There are various other definitions of an interaction model which put further requirements on the interaction sets. For example, there exist notions of a* connector *c and a* connector set *C [72–74]. A connector is an interaction in our sense. A connector set is an interaction set that satisfies the additional condition that all connectors in C are maximal with respect to set inclusion. Using this notion there is a further special case: Arbitrary subsets of connectors may be declared as* complete interactions[2]*. The interaction set for this kind of system is then given by $C \cup Comp$ where $Comp$ is the set of complete interactions.*

It is clear that these definitions are special cases of the notion of an interaction set Int as defined above. We will always work with this notion

[2]Further restrictions may be imposed on the set of complete interactions. For example, Gössler and Sifakis [72] require every superset of a complete interaction to also be complete.

of an interaction set, keeping in mind that the results hold for the special cases, as well.

We present an example illustrating the notions. We model a version of the dining philosophers [60] as an interaction system. In such concrete examples we deviate from the naming scheme in Definition 2.1.1 if intuitive names for the objects help for a better understanding.

Example 2.1.1. The problem of the dining philosophers as posed by Dijkstra is set in an academia inhabited by five philosophers. The life cycle of a philosopher primarily consists of his considerations. Every once in a while he gets hungry whereupon he will sit down at the circular dining table. Five plates and five forks are arranged around the table such that in between any two plates there is one fork. In order to be able to eat a philosopher has to pick up both forks to the sides of his plate.

We model a slightly more general version of the dining philosophers consisting of $m \geq 2$ philosophers and m forks that have to be shared. We represent each philosopher by a component p_i and each fork by a component f_i where $0 \leq i \leq m-1$. We get $K'_{phil_m} := \{p_i, f_i | 0 \leq i \leq m-1\}$. The port set $\mathcal{A}_{p_i}$ of philosopher p_i contains the ports $activate_i$, $enter_i$, get_i^r, get_i^l, eat_i, put_i^r, put_i^l, and $leave_i$. So far, we have not provided any explanation for the ports $activate_i$, $enter_i$, and $leave_i$. For now, we only require that the philosophers have to be activated simultaneously, i.e., there is an interaction $act := \{activate_0, \ldots, activate_{m-1}\}$. The port set of fork f_i is given by $\mathcal{A}_{f_i} := \{get_i, put_i\}$. There are two interactions allowing the i-th philosopher to pick up and put down the fork to his right: $acquire_i^r := \{get_i^r, get_i\}$ respectively $release_i^r := \{put_i^r, put_i\}$. Analogously, there are interactions involving the i-th philosopher and the fork to his left: $acquire_i^l := \{get_i^l, get_{i+1 \bmod m}\}$ respectively $release_i^l := \{put_i^l, put_{i+1 \bmod m}\}$. Finally, we introduce the interactions $\{eat_i\}$ allowing p_i to eat and $\{enter_i\}$ respectively $\{leave_i\}$. We define $Int'_{phil_m} := \{act, acquire_i^r, release_i^r, acquire_i^l, release_i^l, \{eat_i\}, \{enter_i\}, \{leave_i\} \mid 1 \leq i \leq m\}$ and $IM'_{phil_m} := (CS'_{phil_m}, Int'_{phil_m})$.

IM'_{phil_m} is depicted in Figure 2.1. The line connecting the ports $activate_i$ represents the interaction act, for example. No lines end in $enter_i$, eat_i, and $leave_i$ because these ports are contained in singleton interactions.

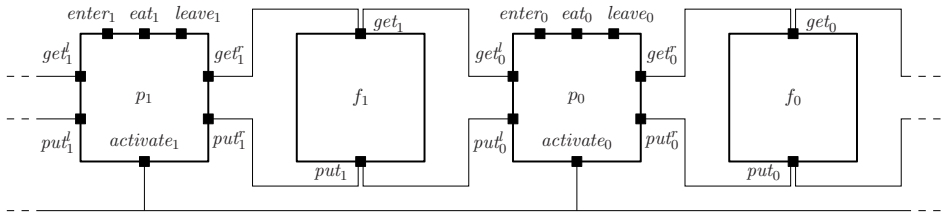

Figure 2.1: Graphical representation of IM'_{phil_m}

The example illustrates the great flexibility which is provided by the notion of an interaction model. Int'_{phil_m} contains interactions consisting of a single action, binary interactions, as well as an interaction involving m components. The degree of synchronization is not restricted. This is not the case for formalisms only allowing binary interaction between the processes such as CCS [107], for example. Other formalisms like CSP [83] or I/O--automata [96] allow multiway communication between an arbitrary number of components but such a communication has to be realized by synchronization over ports of the *same* name. In particular, this means that one port of a component can only be used for exactly one cooperation, namely for cooperation with *all* other components whose alphabets contain that respective port. By contrast, the ports of f_i are used for synchronization with various components (the two philosophers next to the fork). In some models these problems can be circumvented but this is not achieved straightforwardly and partly requires comparatively involved constructions.

The second layer of description of an interaction system is the *dynamic* layer. It is given by the local behavior of the components. For each component we specify the local behavior by a labeled transition system where the labels are taken from the port set of the component. The global behavior of the system is obtained by combining these local systems according to the interaction set: In a global state an interaction α can be performed if all components involved in α offer the required port in the corresponding local states. Performing α results in the global state change obtained by performing the corresponding local transitions for the components involved. Thus, the availability of actions in the global system is restricted in two ways. On the one hand, it is restricted by the order of execution of the actions

specified by the transition system of the component. It is also restricted by the requirement that every action has to be synchronized according to one of the interactions it is contained in.

Definition 2.1.2. *Let IM be an interaction model. Let* $\{T_i\}_{i\in K}$ *be a family of labeled transition systems.* $T_i := \left(Q_i, \mathcal{A}_i, \rightarrow_i, q_i^0\right)$ *is a tuple, where* Q_i *is a finite set of* ***local states****,* $\rightarrow_i \subseteq Q_i \times \mathcal{A}_i \times Q_i$ *is the* ***local transition relation***[3]*, and* $q_i^0 \in Q_i$ *is the* ***local initial state****. For* $q_i \in Q_i$ *we define* $\mathcal{en}(q_i) := \{a_i \in \mathcal{A}_i | \exists q_i' \in Q_i \text{ with } q_i \xrightarrow{a_i}_i q_i'\}$ *the set of ports that are* ***enabled*** *in* q_i*. Further let* $Int' \subseteq Int$ *be a subset of interactions.* $Int'(q_i) := \{\alpha \in Int' | \alpha \cap \mathcal{en}(q_i) \neq \emptyset\}$ *denotes the set of interactions in* Int' *that contain a port in* $\mathcal{en}(q_i)$*. A local state* q_i *is called* ***complete*** *if there exists an interaction* $\alpha = \{a_i\} \in Int(q_i)$*. We write* $need(q_i) := \bigcup_{\alpha \in Int(q_i)} comp(\alpha) \backslash \{i\}$ *for the set of components that* i ***potentially needs for cooperation in*** q_i*.*

$Sys := \left(IM, \{T_i\}_{i\in K}\right)$ *is an* ***interaction system*** *with* ***induced global transition system*** $\tilde{T}_{Sys} := \left(Q_{Sys}, Int, \rightarrow, q^0\right)$*. Here:*

1. $Q_{Sys} := \prod_{i\in K} Q_i$*, we call* $q = (q_1, \ldots, q_n)$ *a* ***(global) state*** *of Sys.*

2. $\rightarrow \subseteq Q_{Sys} \times Int \times Q_{Sys}$ *is the* ***(global) transition relation*** *where for all* $q, q' \in Q_{Sys}$ *and all* $\alpha \in Int$ *we have* $q \xrightarrow{\alpha} q'$ *if and only if for all* $i \in comp(\alpha)$ *with* $i(\alpha) = \{a_i\}$ *we have* $q_i \xrightarrow{a_i}_i q_i'$ *and* $q_i' = q_i$ *for all other components.*

3. $q^0 = \left(q_1^0, \ldots, q_n^0\right)$ *is the* ***(global) initial state****.*

For a state q *and interaction* α *we say that* α *is* ***enabled*** *in* q *if* $i(\alpha) \subseteq \mathcal{en}(q_i)$ *for all* $i \in comp(\alpha)$*. A state* q *is* ***reachable*** *in Sys if there is a sequence* $\sigma = q^0 \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_l} q$ *in* $\tilde{T}_{Sys}$ *that starts in* q^0 *and ends in* q*. We denote the set of reachable states in Sys by* $reach(Sys) := \{q | q \text{ is reachable in } Sys\}$*. We define* $T_{Sys} := (reach(Sys), Int, \rightarrow, q^0)$ *to be the* ***induced global behavior*** *of Sys. Here* $\rightarrow \subseteq reach(Sys) \times Int \times reach(Sys)$ *is the labeled transition relation of* $\tilde{T}_{Sys}$ *restricted to the reachable states.*

[3]We write $q_i \xrightarrow{a_i}_i q_i'$ instead of $(q_i, a_i, q_i') \in \rightarrow_i$.

*A **path of length** l in Sys is a sequence $\sigma = q \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{l-1}} q^l$ in T_{Sys}*[4]*. A **run** in Sys is an infinite sequence $\sigma = q \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} q^2 \xrightarrow{\alpha_2} \ldots$ in T_{Sys}. A **cycle** in Sys is a path σ of length ≥ 1 in T_{Sys} such that the first and the last state of σ coincide. Let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports and let σ be a path or a run. We say that $\mathcal{A}_0$ **participates in** σ if there is an interaction α_s on σ such that $\mathcal{A}_0 \cap \alpha_s \neq \emptyset$. We say that a component i **participates in** σ if $\mathcal{A}_i$ participates in σ.*

We will write Q instead of Q_{Sys} if there is no danger of resulting confusion. Further, we will also refer to T_i as the local behavior of i. Note that paths, runs, and cycles are defined for the reachable part of *Sys*, i.e., each path, run, or cycle always starts in a reachable state. It becomes clear now why we made the assumption that every interaction α contains at most one port of every component. Because every local transition is labeled with exactly one port it would not be clear what effect the execution of α containing two ports $a_i \neq b_i$ of component i would have for i.

Remark 2.1.2. *Without loss of generality we assume that $\mathit{en}(q_i) \neq \emptyset$ for all i and all q_i, i.e., there are no local sink states or dead ends but every local state has at least one outgoing transition. This property can always be enforced by introducing idle actions allowing a component to loop in a state that originally did not enable any action. We get $Int(q_i) \neq \emptyset$ for all q_i and $need(q_i) \neq \emptyset$ for all q_i that are not complete. We also assume that for each i and each $a_i \in \mathcal{A}_i$ at least one transition in T_i is labeled with a_i and that every q_i is reachable from q_i^0 in T_i.*

The way that labeled transition systems are used for the description of the behavior of the components is similar to the approach taken for interface-automata [56] and I/O-automata [96]. It is possible to model a system in these formalisms by an interaction system. Furthermore, CCS- and CSP-processes [83, 107] can also be modeled as an interaction system. Conversely, it is also possible to model an interaction system in these models. It should be noted that this translation is more intricate because the flexibility provided by the notion of an interaction set has to be modeled in a more

[4] If $l = 0$ we set $q^l := q$.

restrictive setting, i.e., only binary communication is allowed respectively synchronization is only possible over actions of the same name.

Example 2.1.1 continued: We assume that all philosophers respectively all forks exhibit the same following behavior up to renaming of states and actions. When a philosopher gets hungry he first has to be activated. We assume that the table is situated in a distinct dining room. Therefore the philosopher has to enter the dining room before being able to take the fork to his right and then take the fork to his left. Having acquired both forks he may eat. Afterwards he puts down the forks and leaves the dining room to go back to his considerations. The transition system modeling this behavior is given in Figure 2.2 a) for p_i. The set of local states is $Q_{p_i} := \{q_{p_i}^0, \ldots, q_{p_i}^7\}$. For better readability, we denote $q_{p_i}^x$ by its superindex x in the figure. The behavior of the forks is very simple: Each fork can be picked up and put down afterwards. This behavior is depicted in Figure 2.2 b) for f_i. We have $Q_{f_i} := \{q_{f_i}^0, q_{f_i}^1\}$. We use the same convention for denoting the states of f_i in the figure. We get $Sys'_{phil_m} := (IM'_{phil_m}, \{T_i\}_{i \in K'_{phil_m}})$.

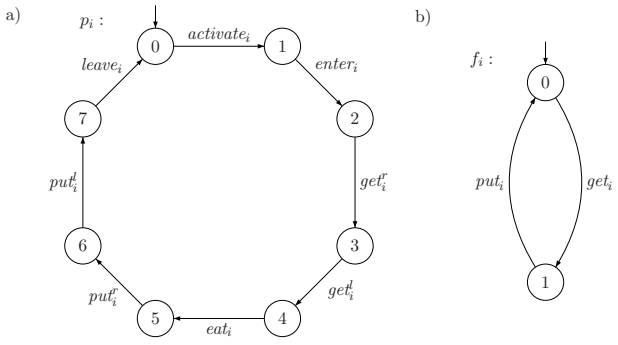

Figure 2.2: The local behavior of the components in K'_{phil_m}

It is easy to equip a component with a different behavior. For example, by simply adjusting T_{p_i} we could allow the i-th philosopher to pick up the left fork first or we could even allow him to nondeterministically choose which fork to pick up first. We emphasize once more that this can be done without touching the glue-code defining the interactions between the components.

In this thesis *Sys* will always denote an interaction system where *IM*

and the T_i are given as above. When an interaction system is specified in an example, we usually state the set K of components, the port sets $\mathcal{A}_i$, the interaction set *Int*, and the local behaviors T_i. We shall then simply speak of the induced interaction system *Sys*, where $Sys := \big(IM, \{T_i\}_{i\in K}\big)$, $IM := (CS, Int)$, and $CS := (K, \{\mathcal{A}_i\}_{i\in K})$.

For a given interaction system *Sys* we need a notion of subsystem with respect to a nonempty subset $K' \subseteq K$ of components.

Definition 2.1.3. *Let Sys be an interaction system and let $K' \subseteq K$ be a nonempty subset of components. The* ***projection of*** *Sys* ***to*** *K' is the interaction system given by $Sys\downarrow_{K'} := \big(IM\downarrow_{K'}, \{T_i\}_{i\in K'}\big)$. Here*

- $IM\downarrow_{K'} := (CS\downarrow_{K'}, Int\downarrow_{K'})$,
- $CS\downarrow_{K'} := (K', \{\mathcal{A}_i\}_{i\in K'})$,
- *and* $Int\downarrow_{K'} := \{\alpha\downarrow_{K'} \mid \alpha \in Int \text{ and } comp(\alpha)\cap K' \neq \emptyset\}$

where $\alpha\downarrow_{K'} := \{a_j \mid a_j \in \alpha \wedge j \in K'\}$ is the ***projection of*** *α* ***to*** *K'. More generally, let $Int' \subseteq Int$ be a subset of interactions. We write $Int'\downarrow_{K'} := \{\alpha\downarrow_{K'} \mid \alpha \in Int' \text{ and } comp(\alpha)\cap K' \neq \emptyset\}$. The* ***projection of a state*** *q* ***to*** *K' is $q\downarrow_{K'} := (q_i)_{i\in K'}$.*

Let $\sigma = q \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{l-1}} q^l$ be a path in Sys. Let $0 \leq m_1 < m_2 < \ldots < m_r \leq l$ denote those indices with $K' \cap comp(\alpha_{m_s}) \neq \emptyset$. The ***projection of*** *σ* ***to*** *K' is given by:*

$$\sigma\downarrow_{K'} := q\downarrow_{K'} \xrightarrow{\alpha_{m_1}\downarrow_{K'}} q^{m_1+1}\downarrow_{K'} \xrightarrow{\alpha_{m_2}\downarrow_{K'}} q^{m_2+1}\downarrow_{K'} \xrightarrow{\alpha_{m_3}\downarrow_{K'}} \ldots \xrightarrow{\alpha_{m_r}\downarrow_{K'}} q^{m_r+1}\downarrow_{K'}$$

Let $\sigma = q \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} q^2 \xrightarrow{\alpha_2} \ldots$ be a run in Sys. If there are finitely many m with $K' \cap comp(\alpha_m) \neq \emptyset$ then the ***projection of*** *σ* ***to*** *K' is the path*

$$\sigma\downarrow_{K'} := q\downarrow_{K'} \xrightarrow{\alpha_{m_1}\downarrow_{K'}} q^{m_1+1}\downarrow_{K'} \xrightarrow{\alpha_{m_2}\downarrow_{K'}} q^{m_2+1}\downarrow_{K'} \xrightarrow{\alpha_{m_3}\downarrow_{K'}} \ldots \xrightarrow{\alpha_{m_r}\downarrow_{K'}} q^{m_r+1}\downarrow_{K'}$$

in $Sys\downarrow_{K'}$ where $0 \leq m_1 < m_2 < \ldots < m_r < \infty$ are defined as above. Otherwise the ***projection of*** *σ* ***to*** *K' is the run*

$$q\downarrow_{K'} \xrightarrow{\alpha_{m_1}\downarrow_{K'}} q^{m_1+1}\downarrow_{K'} \xrightarrow{\alpha_{m_2}\downarrow_{K'}} q^{m_2+1}\downarrow_{K'} \xrightarrow{\alpha_{m_3}\downarrow_{K'}} \ldots$$

in $Sys\downarrow_{K'}$ where $0 \leq m_1 < m_2 < \ldots$ are defined as above.

Remark 2.1.3. *We also call Sys* $\downarrow_{K'}$ *the* subsystem of *Sys* with respect to K'. *Note that we do not necessarily obtain a run* $\sigma \downarrow_{K'}$ *when we consider the projection to* K' *of a run* σ *of the global system because* K' *may only participate in a finite number of the interactions occurring on* σ.

2.2 Properties of Interaction Systems

Next, we define the properties of interaction systems that we will investigate. We deal with deadlock-freedom first. There are various reasons why deadlock-freedom is a key property with regard to distributed systems:

1. Deadlock-freedom itself is a desirable property in systems-design. The case that a system may reach a state where all components mutually block each other due to a conceptual mismatch in the specification of the system should clearly be avoided. In such a state the system will not be able to fulfill its functionality any more and depending on where the system is deployed the consequences may vary from nuisance over financial loss to disaster.

2. Deadlock-freedom is a basic requirement in order to be able to define other generic properties that make statements about all runs (e.g., progress), because this kind of property would be present automatically if there is no run. For a deadlock-free system every path can be elongated to form a run because in every reachable state at least one interaction is enabled.

3. Checking an arbitrary regular safety property can be reduced to checking deadlock-freedom [68]. Thus, a range of properties can be handled once one has a thorough understanding of deadlock-freedom.

We define the notions of deadlock in a state q respectively of deadlock--freedom of *Sys*. We distinguish between local and global deadlocks in q. A deadlock is a state q which does not enable any interaction, i.e., a dead end in $\tilde{T}_{Sys}$. Deadlock-freedom of *Sys* simply describes the fact that no such dead end is reachable in *Sys*.

Definition 2.2.1. *Let Sys be an interaction system and let $q \in Q_{Sys}$ be a global state. We say that q is a* ***(global) deadlock (state)*** *of Sys if no interaction is enabled in q. Sys is* ***free of global deadlocks*** *or* ***deadlock-free*** *if it does not contain any reachable (global) deadlock.*

Let $D \subseteq K$ be a nonempty subset of components. D is a ***local deadlock in*** *q if for all $i \in D$ we have:*

$$\alpha \in Int(q_i) \Rightarrow \exists j \in D \cap comp(\alpha) \text{ with } j(\alpha) \not\subseteq en(q_j)$$

Sys is ***free of local deadlocks*** *if it does not contain any reachable local deadlock.*

A global deadlock in q is a local deadlock with $D = K$. If D constitutes a local deadlock in q then the components in D will never change their local state any more. Furthermore, whenever we consider a local deadlock D in a state q we can always find a minimal local deadlock $D' \subseteq D$ in q (in the sense that D' is a local deadlock but no proper subset of D' has this property). Thus, we may always assume that D is minimal.

According to Remark 2.1.2 we only consider interaction systems where the local transition systems do not contain any sink states. Furthermore, we have $\bigcup_{\alpha \in Int} = \mathcal{A}$. This means that there cannot be any deadlocks which are caused by the fact that one or more components have reached a local state that does not enable any port. Since we only consider finite systems it can therefore be seen that a deadlock can only occur because there are groups of components that mutually block each other. In particular, if there is a deadlock in q it is always possible to find a subset K' of at least two components such that there exists a cycle of waiting relations between the components in K'. This means that the components can be given an order such that each component wants to perform an interaction involving the next (calculating mod $|K'|$) component which in turn does not enable the required port. Of course, in a system where certain local states do not enable *any* port there can also be global states which do not enable any interaction because none of the local states offers a transition. We do not consider this kind of deadlock because usually such local sink states indicate some kind of successful termination of a process, and reaching a combination of such states should not be considered a fault of the system.

A global deadlock as above on the other hand indicates that there has been a conceptual mismatch in the combination of the components. One might ask whether it is possible that the system can indefinitely delay the execution of an enabled interaction and this way cause a "deadlock" even though interactions are enabled. However, interaction systems are not designed for such "malicious" behavior on the part of the system (which simply decides that it does not *want* to perform any interaction). The global semantics of an interaction system describes which global states are reachable by executing *any* possible finite sequence of enabled interactions. Then a deadlock is a state which *cannot* be left any more.

Example 2.1.1 continued: We take a closer look at the deadlock-behavior of Sys'_{phil_m}. For simplicity we consider $m = 3$. The following consideration can be repeated analogously for arbitrary m. By performing the interactions act, $\{enter_1\}$, $\{enter_2\}$, $\{enter_3\}$, $acquire^r_0$, $acquire^r_1$, and $acquire^r_2$ in Int'_{phil_m} the system will reach the state:

$$\left(q^1_{f_0}, q^3_{p_0}, q^1_{f_1}, q^3_{p_1}, q^1_{f_2}, q^3_{p_2}\right)$$

In this state every philosopher has one fork and waits for the other. This constitutes a deadlock, and Sys'_{phil_m} is not deadlock-free. This is not surprising, though, because it is well-known that the dining philosophers are afflicted by deadlocks if the system is not further restricted in some way.

Various strategies have been presented in order to avoid deadlocks in the system of the dining philosophers (cf. Ben-Ari [31], for example). The strategy we model here uses a control component c to make sure that at most $m-1$ philosophers are at the table at the same time. We extend the example as follows: Set $K_{phil_m} := K'_{phil_m} \cup \{c\}$ with $\mathcal{A}_c := \{enter, leave\}$. We get the new component system $CS_{phil_m} := (K_{phil_m}, \{\mathcal{A}_i\}_{i \in K_{phil_m}})$. The interaction set Int_{phil_m} is obtained from Int'_{phil_m} by replacing each interaction $\{enter_i\}$ by $\epsilon_i = \{enter_i, enter\}$ and each interaction $\{leave_i\}$ by $\lambda_i = \{leave_i, leave\}$. We get $IM_{phil_m} := \left(CS_{phil_m}, Int_{phil_m}\right)$. The local behavior of c is given in Figure 2.3. Again, in the figure we denote local states in $Q_c := \{q^x_c | 0 \leq x \leq m-1\}$ by their superindex x. Note that x denotes the number of philosophers that are currently in the dining room. In Sys_{phil_m} at most $m-1$ philosophers are able to acquire at least one fork at the same time because each philosopher

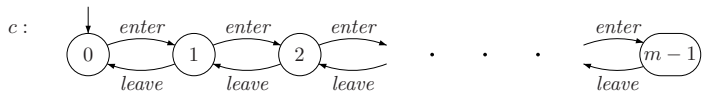

Figure 2.3: The local behavior of c

has to perform his $enter_i$ action before he can actually take a fork. This action has to be synchronized with *enter* of c which can be performed at most $m-1$ times without any philosopher leaving the table by means of an interaction λ_i. The pigeonhole-principle can be used to show that in this case there is always at least one philosopher who can pick up two forks and eat. The deadlock above cannot be reached. A simple case distinction shows that there is no other reachable deadlock. Therefore Sys_{phil_m} is deadlock-free.

The example shows that the definition of the allowed communications between components has to be handled with care because seemingly reasonable interaction rules (i.e., each philosopher may take any of his adjacent forks whenever it is free) may result in cyclic waiting relations and unpredictable problems in the global system. Deciding deadlock-freedom of a given system directly from the definition is difficult. Intuitively this is clear because any decision procedure for deadlock-freedom would have to involve a reachability analysis of the global state space which is obtained by taking the Cartesian product of the local state spaces. Therefore we encounter the state space explosion problem mentioned in Chapter 1. A reachability analysis is not feasible. This statement has been formally substantiated by showing that deciding virtually any property of interaction systems is PSPACE-complete. In detail, a reduction from 1-safe Petri nets to interaction systems was given by Majster-Cederbaum and Minnameier [99]. As a result it was possible to derive the PSPACE-hardness of deciding reachability and deadlock-freedom in interaction systems. Majster-Cederbaum and Minnameier [101] exploited and extended these results by giving a chain of reductions between various properties starting with reachability (which is PSPACE-hard) and ending in a property[5] which was shown to be in PSPACE. This showed that deciding

[5]This property is *availability of a component*. It states that on every run interactions involving the component in question are repeatedly enabled.

any property appearing in the chain is PSPACE-complete. These results yield a firm classification of the complexity of deciding the generic properties we investigate here. They show that we cannot expect to find a decision procedure for deadlock-freedom that can be realized in time polynomial in *Sys* and that works for *all* systems.

We turn to progress of a set of components. Intuitively, a component i makes progress if at any point during the execution of the system it will eventually participate in some interaction no matter how the system behaves. For a finite system deciding this property comes down to checking that there is no cycle in T_{Sys} not allowing i to participate. Again, it is clear that there is no efficient decision procedure for progress. In fact, progress is one of the properties in the chain of reductions mentioned above showing that deciding progress is PSPACE-complete. Here we consider the slightly more general notion of progress of a set of ports.

Definition 2.2.2. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a nonempty subset of ports. We say that $\mathcal{A}_0$* ***makes progress in*** *Sys if $\mathcal{A}_0$ participates in every run of Sys. Let $K' \subseteq K$ be a nonempty set of components. We say that K'* ***makes progress in*** *Sys if $\bigcup_{i \in K'} \mathcal{A}_i$ makes progress in Sys. If $K' = \{i\}$ we say that i* ***makes progress in*** *Sys.*

Recall that we only use the term "run" to denote infinite sequences starting in a reachable state. Thus, the definition above only makes a statement about T_{Sys}. We require *Sys* to be deadlock-free. This way we make sure that the definition is consistent with the intuitive meaning of progress of a component. The component should participate in some interaction eventually no matter how the system evolves. This condition is not satisfied if the system may reach a global deadlock. In the extreme case there are no runs at all in a system that contains deadlocks. Then any subset of components would make progress. This does not comply with the idea of progress.

Before continuing, we point out a correlation between local deadlocks and progress. If D is a local deadlock in q no component in D will participate in any interaction any more. This directly implies that D cannot make progress in *Sys* even if *Sys* is deadlock-free. No run starting in q allows D to participate. The converse is not true. If D does not make progress in *Sys*

there does not have to be a state q such that D is a local deadlock in q.

Example 2.1.1 continued: We discuss progress of an arbitrary philosopher in Sys_{phil_m}. Because every philosopher's behavior consists of a linear sequence in which the phases "think" and "eat" alternate, the requirement that he should make progress in the system can be interpreted as making sure that he will not starve to death. If a philosopher makes progress he regularly returns to the table and eats. Before going back to Sys_{phil_m} we want to understand why it is not a priori clear that a modeling obeying to the specification of the dining philosophers will not let any philosopher starve even if that particular modeling is deadlock-free: Even though the group of philosophers makes progress as a whole this does not mean that every philosopher will be able to eat eventually. This might simply be because one of the philosophers is extremely slow in picking up the forks so that the forks are always taken when he is trying to acquire them or because of an unfair scenario which favors one or more philosophers over others. For example, a setting which *never* allows philosopher p_{m-1} to sit down at the table certainly avoids a global deadlock because of circular waiting between the philosophers. On the other hand it is clear that this solution is far from being optimal because this setting basically results in a situation where the number of forks exceeds the number of philosophers which does not reflect the original specification of the problem.

Consider Sys_{phil_m}. Since all philosophers exhibit the same behavior we may investigate progress of philosopher p_0 and without loss of generality generalize the result to the other philosophers. We investigate progress of $\mathcal{A}_0 := \mathcal{A}_{p_i}$. At the moment we only informally argue why p_0 indeed makes progress. We have already seen that the system is deadlock-free. Before explaining why p_0 makes progress we note that for every $\alpha \in Int_{phil_m}$ at least one philosopher participates. This means that in every global step at least one philosopher changes his local state. Now assume that p_0 does not make progress. Let σ be a run of the system which p_0 does not participate in. Because there are only m philosophers but σ is infinite there must be a philosopher $p_j \neq p_0$ that participates infinitely often in σ. The local behavior of this philosopher shows that repeatedly he has to execute the transition labeled

$activate_j$. Globally this is only possible if all other philosophers, in particular p_0, join p_j to perform the interaction $act = \{activate_0, \ldots, activate_{m-1}\}$. This shows that p_0 participates in σ after all. We will return to the question whether p_0 makes progress in Chapter 4 after having introduced a framework allowing for a formal investigation of progress.

2.3 Conclusion and Discussion

This chapter accumulates the basic definitions of the notions needed to be able to speak about interaction systems. We summarized and partially extended the definitions given by Gössler, Sifakis, and others [71–74, 102]. We stated the definitions of an interaction system and of a subsystem of an interaction system with respect to a subset K' of components. Furthermore, we stated the definitions of deadlock-freedom and progress.

We have already mentioned that we did not always stick to the definitions and notations that were originally introduced. This should not be too confusing because it is still possible to relate the corresponding notions and definitions coming from the various sources. We only want to point out one detail which might cause confusion after all: Gössler et al. [74] define a notion of *liveness* which basically coincides with the notion of progress that we stated in this chapter. To make things even more complicated Gössler et al. [73, 74] also include definitions of a notion called progress. It should be noted that this notion of progress does *not* describe the same circumstances we treat here. This tangle of names is rather unfortunate but in the course of time it seemed more appropriate to follow the naming scheme we chose above because it is more coherent with the nomenclature chosen for corresponding properties in other formalisms[6].

We could have chosen a slightly different approach towards the definition of progress. It would have been possible to require $\mathcal{A}_0$ to participate *infinitely* often in every run. This yields a notion of progress which is equivalent to the formulation given in Definition 2.2.2: If $\mathcal{A}_0$ makes progress in *Sys* and

[6]The notion of liveness in Petri nets [110] is more closely related to the notion of progress defined by Gössler et al. [73, 74] than to progress as in Definition 2.2.2.

σ is a run then $\mathcal{A}_0$ participates in σ. We can write $\sigma = \sigma'\sigma''$ such that $\mathcal{A}_0$ participates in σ' and σ'' is again a run of *Sys*. Using the fact that $\mathcal{A}_0$ makes progress we see that $\mathcal{A}_0$ also participates in σ''. Repeating this argument, we see that $\mathcal{A}_0$ participates infinitely often in σ. The opposite implication clearly holds. We chose the formulation given in Definition 2.2.2 to make sure that the definition is more along the lines of the definition of a liveness property given Lamport [93]. There, a liveness property was described as a property stating that "something (good) *must* happen eventually". In our context the "good thing" that is supposed to happen is participation of $\mathcal{A}_0$ no matter how the system behaves. This definition was taken on and formalized by Alpern and Schneider [9]. Regarding a property of a system as a set of infinite executions of the system, Alpern and Schneider [9] define a liveness property $\mathcal{L}$ by requiring that any (finite) execution fragment α can be extended by an infinite execution α' such that $\alpha\alpha' \in \mathcal{L}$. Definition 2.2.2 can be interpreted in this context as follows: Any finite execution fragment of an interaction system *Sys* is a path σ in *Sys* which starts in q^0 and ends in some reachable state. The definition of progress of $\mathcal{A}_0$ states that any extension of σ allows $\mathcal{A}_0$ to participate, and deadlock-freedom of *Sys* guarantees that there is indeed at least one such extension. Note that various slightly different definitions of a liveness property have been given (cf. Sistla [130], for example). Alpern and Schneider [9] argue that their definition is the most general one and therefore we will not further discuss the other definitions. We would also like to mention that often liveness properties are investigated under certain fairness assumptions (e.g., strong or weak fairness) about a scheduler which coordinates the execution of the various enabled transitions. Roughly speaking, such fairness assumptions make sure that the scheduler must not exclude a transition, that is repeatedly/permanently enabled, from execution indefinitely. In our context we do not discuss such fairness assumptions. However, one can interpret progress of $\mathcal{A}_0$ in such a way that even an unfair scheduler which tries to *prevent* $\mathcal{A}_0$ from participating in a run will fail to do so and will have to execute an interaction involving $\mathcal{A}_0$ eventually. We refer to Kindler [88] for a more detailed discussion of liveness properties.

In Section 1.1 we named composability with other components as one

of the main characteristics a component should exhibit. For interaction systems composability is obtained by deriving an induced semantics for the global system from the local transition systems and the interaction set. The formal instruments described so far do not allow for composition of two interaction systems, though. Such a composition operator is desirable, as well, and it exists in the context of interaction systems, cf. Gössler and Sifakis [72] and Gössler et al. [73] for the more special setting treated there or Lambertz [92]. The results of this thesis do not use this operator, which is why we will not go into its details.

We conclude the discussion by mentioning one last issue: Using the concept of an interaction we only model the act of communication between several components. The formalism in its present form does not provide for passing of information between the components. This is not necessary for our considerations because the properties we investigate are mainly influenced (in that a system does or does not have the property) by the fact *that* and *with whom* the components communicate but not by the data they may exchange. Of course, for the verification of a specification that also refers to the information passed on it is necessary to extend the formalism such that it is also able to handle value passing. It is not difficult to do so, for example, by introducing local variables for the components that can be updated according to certain ports and interactions. Variables have been introduced into the data structure implemented in the BIP-tool [29], for example.

Chapter 3

Deadlock-Freedom for Component Systems with Architectural Constraints

3.1 Motivation

We derive a sufficient but efficiently checkable condition for deadlock-freedom by restricting the class of systems we investigate. We get a trade-off. We exclude many systems beforehand that cannot even be handled. However, by only treating a subclass of systems we have extra-information about the systems to be dealt with that can be incorporated into the considerations about the sufficient condition. We specify the subclass by restricting the (communication-)architecture. We will represent the interaction model by a graph on which we will impose the restriction that it should be a tree. A variety of interesting classes of systems can be obtained by alternatively restricting the graph to be, for example, a star, a ring, or a simple mesh. The subclass of tree-like systems is of interest because such systems naturally arise in many applications [30, 33, 79, 84, 103, 104, 115, 131]. Furthermore, tree-like architectures exist in systems with a hierarchical structure. In particular, networks that are built according to the master-slave operator respectively the principle of subordination are tree-like [38, 83]. The client/server systems considered by Abdulla et al. [7] also fall into that category. Roughly speaking, the authors consider such systems consisting of

disconnected stars where a number of clients is connected to a server in the center of the star. A client may be connected to at most one server but the connections between the nodes may change, i.e., clients, servers, respectively edges may be added and deleted. Sommerville [131] also considers tree-like systems resulting from client/server architectures.

The underlying idea of the condition that we derive from the tree-like architecture is the prospect of being able derive deadlock-freedom of the global system by only investigating subsystems consisting of two interacting components. Naively, one might presume that a tree-like architecture of a system alone is sufficient to ensure deadlock-freedom. However, this is by far not sufficient to ensure deadlock-freedom. Thus, conditions have to be established that can be checked for subsystems of size two. A conjecture which stands to reason is that it might suffice to check these subsystems for deadlock-freedom. However, the hope that this is sufficient for deadlock-freedom is in vain. Consider the system consisting of $K := \{i, j, k\}$, the transition systems depicted in Figure 3.1, and $Int := \{\{a_i, a_j\}, \{b_i, b_j\}, \{c_j, c_k\}, \{d_j, d_k\}\}$.

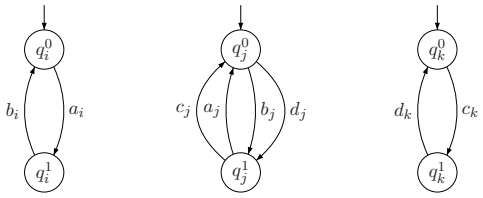

Figure 3.1: The local behavior of the components i, j, and k

Without having formally defined what this means, we content ourselves at the moment with accepting that in an intuitive sense the system has a tree-like communication architecture because a simple cycle of waiting relations involving all three components cannot occur. The subsystems consisting of i and j respectively of j and k are deadlock-free. On the other hand, the system is not deadlock-free because the initial state is a deadlock. We see that it is not sufficient to check for deadlock-freedom in the subsystems of a tree-like system. However, the example also already hints at what should be checked for instead. In each subsystem a pair of local states (namely the initial state of the subsystem) is reachable where the components want to cooperate while this is impossible because the actions

offered are not compatible. Combining these local states we end up with the deadlock in the global system. Amongst others, this observation will be exploited in the sufficient condition for deadlock-freedom. In the end of the chapter we will return to the point of origin, and we will discuss whether it is sufficient to check deadlock-freedom of subsystems of size two in order to ensure global deadlock-freedom if the system's structure is further restricted.

3.2 Reachability in Interaction Systems

The difficulty of deciding deadlock-freedom stems from the fact that it requires a reachability analysis on the global state-space. In fact, the PSPACE-completeness of deciding deadlock-freedom in interaction systems can be attributed to the PSPACE-completeness of deciding reachability of a state q. Having said that, it is clear that the two conditions occurring in the definition of a deadlock state (reachability of q vs. deadlock in q) are independent of each other. It makes sense to separate concerns. Reachability in interaction systems — or more generally in any kind of transition system for that matter — is an interesting question in itself, and we will treat it separately in this section.

The complexity result mentioned above shows that we cannot expect an efficient decision procedure for reachability that always works. We will exhibit a condition which can be checked efficiently and allows to exclude the possibility that certain global states are reachable. We want to motivate our approach by first taking a slightly more general view at the techniques that will be used. We want to derive results about the reachability of global states from investigating reachability in subsystems. The definition of $Sys \downarrow_{K'}$ for a subset K' of components makes sure that the projection $q \downarrow_{K'}$ of a reachable global state q is also reachable in the subsystem. In some cases, this observation may be used to show that a global state is not reachable. It is clear that computing $reach(Sys \downarrow_{K'})$ is less complex than analyzing the global system. On the other hand, by hiding components we loose information. Thus, we have to find a balance between precision and efficiency where the two extremes of the spectrum consist of a completely precise but completely inefficient approach (we investigate the subsystem defined by $K' = K$, i.e.,

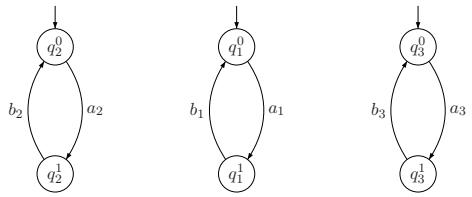

Figure 3.2: The local behavior of the components in K_1

Sys itself) on the one hand and a very efficient approach that does not convey any information at all (we investigate subsystems where K' consists of a single component, i.e., we investigate the local transition systems). As noted in the previous section, we want to focus on subsystems consisting of two components, only. Interestingly, it is possible to derive nontrivial information about reachability from these subsystems even though it might seem that this approach is still a very imprecise one.

In this chapter we consider systems where every port of every component occurs in exactly one interaction.

Definition 3.2.1. *Let $IM = (CS, Int)$ be an interaction model. IM has **strongly exclusive communication** if for $\alpha, \alpha' \in Int$ with $\alpha \neq \alpha'$ we have $\alpha \cap \alpha' = \emptyset$. If Sys is an interaction system whose interaction model has strongly exclusive communication we also say that Sys has **strongly exclusive communication**.*

Example 3.2.1. We introduce two running examples for this chapter.

1. Consider $K_1 := \{1, 2, 3\}$. The transition systems of the components are depicted in Figure 3.2. For each i the port set is understood to coincide with the set of labels of T_i. Define $Int_1 := \{\{a_1, a_2\}, \{a_1, a_3\}, \{b_1, b_2\}, \{b_1, b_3\}\}$. We denote the interactions (in the order above) by α_1 to α_4. The induced interaction system is called Sys_1.

2. Consider $K_2 = \{5, 6, 7, 8\}$. The transition systems of the components are depicted in Figure 3.3. Define $Int_2 := \{\{a_5, a_6, a_7\}, \{a_6, a_7, a_8\}, \{b_5, b_6, b_7\}, \{b_6, b_7, b_8\}\}$. Denote the interactions (in the order above) by β_1 to β_4. The induced interaction system is called Sys_2.

Neither Sys_1 nor Sys_2 have strongly exclusive communication. For example,

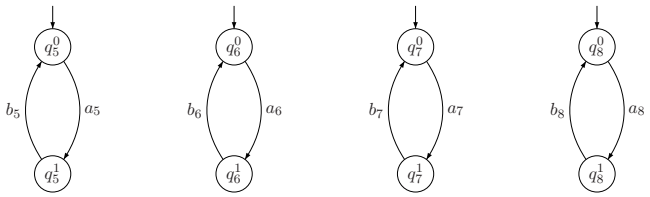

Figure 3.3: The local behavior of the components in K_2

we have $\alpha_1 \cap \alpha_2 \neq \emptyset$ for Sys_1 and $\beta_1 \cap \beta_2 \neq \emptyset$ for Sys_2.

At first glance the requirement about strongly exclusive communication seems to be rather strong. In particular, it appears to refute one of the features of interaction systems that we stressed in Chapter 2: the fact that one port may appear in various interactions. However, the following lemma and corollary show that this assumption is not a restriction after all. For an interaction system *Sys* we may construct in polynomial time a system $\overline{Sys}$ that essentially exhibits the same behavior as *Sys* and has strongly exclusive communication. Roughly speaking, $\overline{Sys}$ results from *Sys* by replacing in T_i every transition labeled a_i by a set of *parallel* transitions labeled by an indexed version of a_i such that there is one transition for every interaction that a_i occurs in. In order to be able to distinguish the transition relation in $\overline{Sys}$ from the transition relation in *Sys* we write $\rightharpoonup$ respectively $\rightarrow$ in this lemma. Contrary to our convention of moving proofs to the appendix of each chapter we include the proof of the lemma here because it is a constructive proof explaining how $\overline{Sys}$ is assembled.

Lemma 3.2.1. *Let Sys be an interaction system.*

There is an interaction system $\overline{Sys}$ with strongly exclusive communication such that $\overline{K} = K$, $\overline{Q}_i = Q_i$, $\left|\overline{Int}\right| = |Int|$, and for any $q, \tilde{q} \in \overline{Q} = Q$ there exists $\overline{\alpha} \in \overline{Int}$ with $q \xrightarrow{\overline{\alpha}} \tilde{q}$ if and only if there exists $\alpha \in Int$ with $q \xrightarrow{\alpha} \tilde{q}$. $\overline{Sys}$ can be constructed in time polynomial in the size of Sys.

Proof. We construct $\overline{Sys}$ in a step-by-step manner. Set $\overline{K} := K$ and consider $a_i \in \mathcal{A}_i$. For every $\alpha \in Int$ with $a_i \in \alpha$ we define a new port a_i^α. We set:

$$\overline{\mathcal{A}_i} := \bigcup_{\alpha \in Int, a_i \in \alpha} a_i^\alpha$$

Component i participates with at most one port in every interaction of *Sys*. Therefore, $|\overline{\mathcal{A}_i}| \in O(|Int|)$. We get $\overline{CS} := (\overline{K}, \{\overline{\mathcal{A}_i}\}_{i \in \overline{K}})$. Next we define the new interaction set $\overline{Int}$. Consider an interaction $\alpha = \{a_{i_1}, \ldots, a_{i_k}\} \in Int$. We define $\overline{\alpha} := \{a^{\alpha}_{i_1}, \ldots, a^{\alpha}_{i_k}\}$ and we set $\overline{Int} := \bigcup_{\alpha \in Int} \overline{\alpha}$. Clearly, $|\overline{Int}| = |Int|$. Finally, we define the local transition systems for the components in $\overline{K}$. Let $i \in \overline{K}$. We set $\overline{Q_i} := Q_i$ and $\overline{q_i^0} := q_i^0$. The new transition relation $\rightharpoonup_i \subseteq \overline{Q_i} \times \overline{\mathcal{A}_i} \times \overline{Q_i}$ is defined as follows. For $q_i, \tilde{q}_i \in \overline{Q_i}$ and $a_i^{\alpha} \in \overline{\mathcal{A}_i}$ we set $q_i \overset{a_i^{\alpha}}{\rightharpoonup}_i \tilde{q}_i$ if and only if $q_i \overset{a_i}{\rightarrow}_i \tilde{q}_i$. $\overline{Sys}$ has strongly exclusive communication.

We have to show that for any $q, \tilde{q} \in \overline{Q} = Q$ there is $\overline{\alpha} \in \overline{Int}$ with $q \overset{\overline{\alpha}}{\rightharpoonup} \tilde{q}$ if and only if there is $\alpha \in Int$ with $q \overset{\alpha}{\rightarrow} \tilde{q}$. Let $\overline{\alpha} = \{a^{\alpha}_{i_1}, \ldots, a^{\alpha}_{i_k}\}$. We have:

$$\begin{aligned} q \overset{\overline{\alpha}}{\rightharpoonup} \tilde{q} &\Leftrightarrow \forall i_j \in comp(\overline{\alpha}) : q_{i_j} \overset{a^{\alpha}_{i_j}}{\rightharpoonup}_{i_j} \tilde{q}_{i_j} \text{ and } \forall i \notin comp(\overline{\alpha}) : \tilde{q}_i = q_i \\ &\Leftrightarrow \forall i_j \in comp(\alpha) : q_{i_j} \overset{a_{i_j}}{\longrightarrow}_{i_j} \tilde{q}_{i_j} \text{ and } \forall i \notin comp(\alpha) : \tilde{q}_i = q_i \\ &\Leftrightarrow q \overset{\alpha}{\rightarrow} \tilde{q} \end{aligned}$$

Clearly, $\overline{Sys}$ can be constructed in time polynomial in the size of *Sys*. □

The following corollary is an immediate consequence. It states that *Sys* and $\overline{Sys}$ exhibit the same behavior with respect to freedom of local and global deadlocks.

Corollary 3.2.1. *Let Sys be an interaction system and let $\overline{Sys}$ be defined as above. Let $q \in Q = \overline{Q}$.*

1. *q is reachable in Sys if and only if it is reachable in $\overline{Sys}$.*
2. *q is a deadlock of Sys if and only if it is a deadlock of $\overline{Sys}$.*
3. *$D \subseteq K = \overline{K}$ is a local deadlock of Sys in q if and only if it is a local deadlock of $\overline{Sys}$ in q.*

Whenever we investigate freedom of local or global deadlocks of a system *Sys* we construct and investigate $\overline{Sys}$ instead. We want to stress that $\overline{Sys}$ is only constructed for the sake of proving deadlock-freedom. Once deadlock-freedom has been established for $\overline{Sys}$ (and consequently also for *Sys*) we discard $\overline{Sys}$ and deploy the original interaction system in whatever situation intended (in particular we maintain the possibility of using one port of a component for various cooperations).

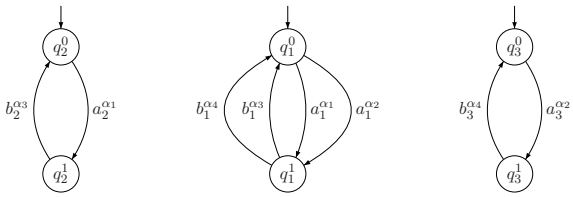

Figure 3.4: The local behavior of the components in $\overline{K_1}$

Example 3.2.1 continued: We trace the construction in the proof of Lemma 3.2.1 by constructing $\overline{Sys_1}$ and $\overline{Sys_2}$.

1. We get $\overline{K_1} = \{1, 2, 3\}$. The port sets are $\overline{\mathcal{A}_1} = \{a_1^{\alpha_1}, a_1^{\alpha_2}, b_1^{\alpha_3}, b_1^{\alpha_4}\}$, $\overline{\mathcal{A}_2} = \{a_2^{\alpha_1}, b_2^{\alpha_3}\}$, and $\overline{\mathcal{A}_3} = \{a_3^{\alpha_2}, b_3^{\alpha_4}\}$. We get $\overline{Int_1} = \{\{a_1^{\alpha_1}, a_2^{\alpha_1}\}, \{a_1^{\alpha_2}, a_3^{\alpha_2}\}, \{b_1^{\alpha_3}, b_2^{\alpha_3}\}, \{b_1^{\alpha_4}, b_3^{\alpha_4}\}\}$ where the interactions are named $\overline{\alpha_1}$ to $\overline{\alpha_4}$. The local transition systems are given in Figure 3.4 (note that the local behaviors of 2 and 3 have not changed up to renaming of the transition labels).

2. We have $\overline{K_2} = \{5, 6, 7, 8\}$. Furthermore, $\overline{\mathcal{A}_5} = \{a_5^{\beta_1}, b_5^{\beta_3}\}$, $\overline{\mathcal{A}_6} = \{a_6^{\beta_1}, a_6^{\beta_2}, b_6^{\beta_3}, b_6^{\beta_4}\}$, $\overline{\mathcal{A}_7} = \{a_7^{\beta_1}, a_7^{\beta_2}, b_7^{\beta_3}, b_7^{\beta_4}\}$, and $\overline{\mathcal{A}_8} = \{a_8^{\beta_2}, b_8^{\beta_4}\}$. The interaction set is $\overline{Int_2} := \{\{a_5^{\beta_1}, a_6^{\beta_1}, a_7^{\beta_1}\}, \{a_6^{\beta_2}, a_7^{\beta_2}, a_8^{\beta_2}\}, \{b_5^{\beta_3}, b_6^{\beta_3}, b_7^{\beta_3}\}, \{b_6^{\beta_4}, b_7^{\beta_4}, b_8^{\beta_4}\}\}$ where the interactions are named $\overline{\beta_1}$ to $\overline{\beta_4}$. The local transition systems are given in Figure 3.5.

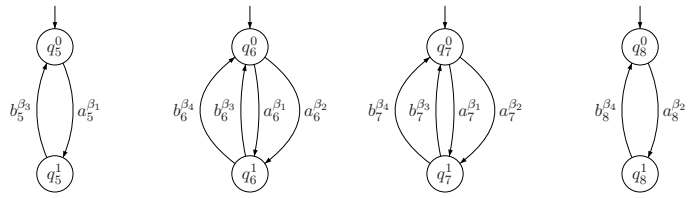

Figure 3.5: The local behavior of the components in $\overline{K_2}$

In the following we will make extensive use of the notion of a subsystem of *Sys* with respect to K'. We have the following lemma.

Lemma 3.2.2. *Let Sys be an interaction system. Let $K' \subseteq K$ and $q, q' \in Q$. Let $\alpha \in Int$ be an interaction with $comp(\alpha) \cap K' \neq \emptyset$ and $q' \xrightarrow{\alpha} q$.*

1. *There is a transition $q' \downarrow_{K'} \xrightarrow{\alpha\downarrow_{K'}} q \downarrow_{K'}$ in $\tilde{T}_{Sys\downarrow_{K'}}$.*

2. If q is reachable in Sys then $q\downarrow_{K'}$ is reachable in Sys $\downarrow_{K'}$.

Summarizing the bottom line of the previous lemma, we conclude that the projection to K' of *reach*(*Sys*) is contained in *reach*(*Sys* $\downarrow_{K'}$). We say that *reach*(*Sys* $\downarrow_{K'}$) is an over-approximation of *reach*(*Sys*) $\downarrow_{K'}$. If a state $(q_i)_{i\in K'}$ is not reachable in *Sys* $\downarrow_{K'}$ then there cannot be any reachable global state q with $q\downarrow_{K'}=(q_i)_{i\in K'}$. Moreover, the interactions in *Int* leading to a reachable state in *Sys* must be consistent with the interactions in *Int* $\downarrow_{K'}$ leading to the projection of the state in *Sys* $\downarrow_{K'}$. These facts can be used to exclude the possibility that global states are reachable. Since it is not feasible to compute *reach* (*Sys* $\downarrow_{K'}$) for all subsets $K'\subseteq K$ we consider those subsystems obtained by projecting *Sys* to pairs of interacting components. By means of $\overline{Sys_1}$ we illustrate the ideas we use to obtain results about reachability of global states by only looking at pairs of components.

Example 3.2.1 continued: Consider $\overline{Sys_1}$. We want to ensure that the state $\left(q_1^1,q_2^0,q_3^0\right)$ is not globally reachable, but we only want to examine subsystems of size two. When we consider the subsystems $\overline{Sys_1}\downarrow_{\{1,2\}}$, $\overline{Sys_1}\downarrow_{\{1,3\}}$, and $\overline{Sys_1}\downarrow_{\{2,3\}}$ we see that all projections of $\left(q_1^1,q_2^0,q_3^0\right)$ to the corresponding pairs of components are reachable. We cannot use the second statement of Lemma 3.2.2 to exclude the possibility that $\left(q_1^1,q_2^0,q_3^0\right)$ is reachable in $\overline{Sys_1}$. On the other hand, in $\overline{Sys_1}\downarrow_{\{1,2\}}$ the state $\left(q_1^1,q_2^0\right)$ can *only* be reached by executing the port $a_1^{\alpha_2}$ while in $\overline{Sys_1}\downarrow_{\{1,3\}}$ the state $\left(q_1^1,q_3^0\right)$ can *only* be reached by executing $a_1^{\alpha_1}$. The first statement of the lemma shows that $\left(q_1^1,q_2^0,q_3^0\right)$ is not globally reachable because the ports of 1 that would have to be executed are not consistent. Such comparisons of ports, that can be used to reach states in subsystems, may help to exclude the possibility that a global state is reachable. The comparisons can be performed efficiently since we only consider pairs of components.

On the other hand, there are limitations to the exactness of this technique. We replace the local behavior of component 1 by the transition system given in Figure 3.6 (note that implicitly we also change $\overline{\mathcal{A}_1}$ because we introduce new ports), and we add the new component 4 whose behavior is also given in Figure 3.6. We write $\overline{K_3}:=\{1,2,3,4\}$ and $\overline{Int_3}:=\overline{Int_1}\cup\{\{c_1^{\alpha_5},c_4^{\alpha_5}\},\{d_1^{\alpha_6},d_4^{\alpha_6}\}\}$. We name the new interactions $\overline{\alpha_5}$ and $\overline{\alpha_6}$.

Denote the induced interaction system by $\overline{Sys_3}$.

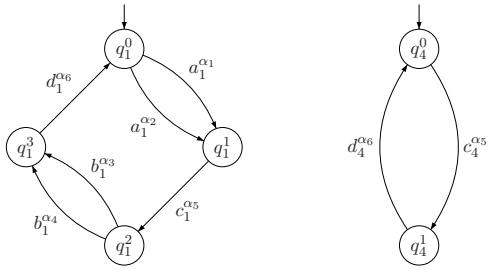

Figure 3.6: The local behaviors of components 1 and 4 in $\overline{K_3}$

We are now interested in the question whether there is $q \in \mathit{reach}(\overline{Sys_3})$ with $q \downarrow_{\{1,2,3\}} = \left(q_1^2, q_2^0, q_3^0\right)$. Repeating the considerations above, we see that in $\overline{Sys_3} \downarrow_{\{1,2\}}$ and in $\overline{Sys_3} \downarrow_{\{1,3\}}$ the states $\left(q_1^2, q_2^0\right)$ respectively $\left(q_1^2, q_3^0\right)$ can both be reached by executing the interaction $\{c_1^{\alpha_5}\}$ which is in $\overline{Int_3} \downarrow_{\{1,2\}}$ as well as in $\overline{Int_3} \downarrow_{\{1,3\}}$. At first glance, we cannot as above exclude the possibility that q_1^2, q_2^0, and q_3^0 are globally reachable in that combination even though they are not. However, when we only focus on 1, 2, and 3, it is clear that $c_1^{\alpha_5}$ can *always* be performed by 1 when it is enabled because it *never* has to wait for 2 or 3. Starting from q_1^2 we search for all local states that can be reached by going *backwards* along those transitions labeled with ports, that are *not* meant for communication with 2 or 3. This "backward-search" results in the local states q_1^2 and q_1^1. As above we then compare the ports of 1 (which were not used for the backward-search) that can be used in $\overline{Sys_3} \downarrow_{\{1,2\}}$ and $\overline{Sys_3} \downarrow_{\{1,3\}}$ to reach q_2^0 respectively q_3^0 in combination with one of these local states. The only possible such port when we observe 1 in parallel with 2 is $a_1^{\alpha_2}$, while the only possible such port when we observe 1 in parallel with 3 is $a_1^{\alpha_1}$. Again, by only looking at pairs of components we may conclude that the combination of the three states in question is not globally reachable.

We now formally define the notions needed in order to substantiate the ideas motivated above. We start by defining the backward-search operator BWS for a local state q_i and a subset K' of components. K' is used to parametrize the operator, and it has to be chosen accordingly to the situation. All ports of i that are *not* used for communication with any component

in K' are eligible for the computation of BWS.

Definition 3.2.2. *Let Sys be an interaction system with strongly exclusive communication. Let $i \in K$ and $K' \subseteq K \setminus \{i\}$. For $q_i \in Q_i$ we define*

$$BWS(q_i, K') := \{q_i' | \exists\ q_i' \xrightarrow{a_i^1} \ldots \xrightarrow{a_i^m} q_i \text{ in } T_i \text{ such that } m \geq 0 \text{ and } \forall l : a_i^l \notin \bigcup_{k \in K'} \mathit{comm}_i(k)\}$$

the set of local states of i from which q_i can be reached by only performing actions that are not used for cooperation with components in K'.

Let $Q_i' \subseteq Q_i$ and $\mathfrak{K}(Q_i') = \{K_{q_i}\}_{q_i \in Q_i'}$ be a family of subsets of components such that $K_{q_i} \subseteq K \setminus \{i\}$ for all $q_i \in Q_i'$. We define

$$BWS(Q_i', \mathfrak{K}(Q_i')) := \bigcup_{q_i \in Q_i'} BWS(q_i, K_{q_i}).$$

If $K_{q_i} = K'$ for all $q_i \in Q_i'$ we write $BWS(Q_i', K')$ instead of $BWS(Q_i', \mathfrak{K}(Q_i'))$.

The idea of $BWS(q_i, K')$ is to find those local states q_i' from which q_i can be reached *without* communication with any component in K'. In other words, if Sys has reached a state q' with $q_i' \in BWS(q_i, K')$ then the components in K' cannot prevent the system from moving to a state q whose entry for i coincides with q_i.

Example 3.2.1 continued: Because paths of length 0 are admitted in the definition, for all local states q_i and all permitted K' we get $q_i \in BWS(q_i, K')$. Further, for all local states q_i the set $BWS(q_i, \emptyset)$ coincides with the set of local states of i from which q_i is reachable. Let i be a component that only communicates with one other component, i.e., there exists a component j such that for all $\alpha \in Int$ with $i(\alpha) \neq \emptyset$ we have $comp(\alpha) = \{i, j\}$. This implies that all transitions in T_i are labeled with ports in $\mathit{comm}_i(j)$. Furthermore, $need(q_i) = \{j\}$ for all $q_i \in Q_i$ and therefore $BWS(q_i, need(q_i)) = \{q_i\}$.

1. We compute BWS for some local states in $\overline{Sys_3}$. Consider the local transition systems of components 2 and 3 and of components 1 and 4 as depicted in Figure 3.4 (p. 43) respectively Figure 3.6 (p. 45). We have $BWS(q_1^2, \{2,3\}) = \{q_1^2, q_1^1\}$. Similarly, we get $BWS(q_1^0, \{2,3\}) =$

$\{q_1^0, q_1^3\}$. On the other hand, $BWS(q_1^2, \{2\}) = Q_1$ because then the edges labeled $a_1^{\alpha_2}$ respectively $b_1^{\alpha_4}$ can also be used for the backward-search. For $i \in \{2,3,4\}$ and $q_i \in Q_i$ we get $BWS(q_i, K') = \{q_i\}$ if $1 \in K'$ and $BWS(q_i, K') = Q_i$ otherwise because the components contained in $\{2,3,4\}$ only communicate with 1. In particular, for $i \in \{2,3,4\}$ and $q_i \in Q_i$ we have $BWS(q_i, need(q_i)) = \{q_i\}$ because $need(q_i) = \{1\}$. For $q_1 \in Q_1$ the set $BWS(q_1, need(q_1))$ contains q_1 and its predecessor in T_1.

2. We similarly deal with the local states in $\overline{Sys_2}$. Consider the local transition systems of the components as depicted in Figure 3.5 (p. 43). For $q_5 \in Q_5$ we have $BWS(q_5, K') = \{q_5\}$ if $K' \cap \{6,7\} \neq \emptyset$ and $BWS(q_5, K') = Q_5$ otherwise because for all $q_5 \in Q_5$ there is only one ingoing transition. This transition is labeled with a port that is used for communication with 6 and 7. An analogous statement holds for $q_8 \in Q_8$. Note that $\overline{Int_2}$ contains interactions of size 3. Therefore the port $a_6^{\beta_1}$, for example, occurs in $\boldsymbol{comm}_6(5)$ and $\boldsymbol{comm}_6(7)$. When we consider $BWS(q_6^1, K')$ this port is only eligible for the computation of the backward-search if neither 5 nor 7 are in K'. An analogous statement holds for $a_6^{\beta_2}$. In detail, we get $BWS(q_6^1, K') = \{q_6^1\}$ if $7 \in K'$ or if $\{5,8\} \subseteq K'$. Otherwise we have $BWS(q_6^1, K') = Q_6$. The local state q_6^0 and $q_7 \in Q_7$ can be treated likewise. This means that, for all local states q_i in $\overline{Sys_2}$ we have $BWS(q_i, need(q_i)) = \{q_i\}$. This results from the facts that for $i \in \{5,8\}$ and $q_i \in Q_i$ we have $need(q_i) = \{6,7\}$, for $q_6 \in Q_6$ we have $need(q_6) = \{5,7,8\}$, and for $q_7 \in Q_7$ we have $need(q_i) = \{5,6,8\}$.

For a local state q_i and a subset Q'_j of local states we define the notion of entry actions. The BWS-operator appears in the context of q_i and in the context of Q'_j. Thus, the entry actions must be parametrized with a subset K' of components (for BWS with respect to q_i) and a family $\mathfrak{K}(Q'_j) = \{K_{q_j}\}_{q_j \in Q'_j}$ of subsets of components (for BWS with respect to Q'_j).

Definition 3.2.3. *Let Sys be an interaction system with strongly exclusive communication. Let $i \in K$ and $K' \subseteq K \setminus \{i\}$. Let $j \in K \setminus \{i\}$. Consider*

$q_i \in Q_i$ and $Q'_j \subseteq Q_j$. Further let $\mathfrak{K}(Q'_j) = \{K_{q_j}\}_{q_j \in Q'_j}$ be a family of subsets of components with $K_{q_j} \subseteq K \setminus \{j\}$.

We define

$$\mathcal{EA}(q_i, K', Q'_j, \mathfrak{K}(Q'_j)) := \{a_i \in \bigcup_{k \in K'} comm_i(k) \mid$$
$$\bullet\ \exists (q'_i, q'_j) \in BWS(q_i, K') \times BWS(Q'_j, \mathfrak{K}(Q'_j))$$
$$\bullet\ \exists (q''_i, q''_j) \text{reachable from} (q^0_i, q^0_j)$$
$$\bullet\ \exists \alpha \in Int \downarrow_{\{i,j\}} \text{such that:}$$
$$a_i \in \alpha \text{ and } (q''_i, q''_j) \xrightarrow{\alpha} (q'_i, q'_j)\}$$

as the set of ***entry actions of*** *i* ***with respect to*** *q_i,* ***K', Q'_j, and*** *$\mathfrak{K}(Q'_j)$.*

If $K_{q_j} = K''$ for all $q_j \in Q'_j$ we write $\mathcal{EA}(q_i, K', Q'_j, K'')$ instead of $\mathcal{EA}(q_i, K', Q'_j, \mathfrak{K}(Q'_j))$. If $Q'_j = \{q_j\}$ we write $\mathcal{EA}(q_i, K', q_j, K_{q_j})$ instead of $\mathcal{EA}(q_i, K', Q'_j, \mathfrak{K}(Q'_j))$.

An entry action $a_i \in \mathcal{EA}(q_i, K', Q'_j, \mathfrak{K}(Q'_j))$ is a port which is for communication with a component in K' and which can be used in $Sys \downarrow_{\{i,j\}}$ to reach a state (q'_i, q'_j) where $q'_i \in BWS(q_i, K')$ (and $q'_j \in BWS(Q'_j, \mathfrak{K}(Q'_j))$ for that matter). Intuitively, this means that a_i is the last port on a path ending in q_i whose execution can be influenced by one or more components in K'. Thus, at least from the point of view of K', it has to be assumed that Sys will reach a state q whose projection to i and j is contained in $\{q_i\} \times Q'_j$ once an interaction involving a_i has been executed.

Example 3.2.1 continued: We compute this notion for our examples:

1. Using the results that we computed for the BWS-operator above, for $\overline{Sys_3}$ (cf. Figure 3.4 (p. 43) respectively Figure 3.6 (p. 45)) we obtain $\mathcal{EA}\left(q^2_1, need(q^2_1), q^0_2, need(q^0_2)\right) = \{a^{\alpha_2}_1\}$ on the one hand as well as $\mathcal{EA}\left(q^2_1, need(q^2_1), q^0_3, need(q^0_3)\right) = \{a^{\alpha_1}_1\}$ on the other. This is because in the corresponding subsystems $\left(q^1_1, q^0_2\right)$ and $\left(q^1_1, q^0_3\right)$ can be reached by executing these ports. Similarly, $\mathcal{EA}\left(q^0_1, need(q^0_1), q^1_2, need(q^1_2)\right) = \{b^{\alpha_4}_1\}$ and $\mathcal{EA}\left(q^0_1, need(q^0_1), q^1_3, need(q^1_3)\right) = \{b^{\alpha_3}_1\}$. Note that switching the order of the states in $\mathcal{EA}$ changes the result. For example, we have $\mathcal{EA}\left(q^0_2, need(q^0_2), q^2_1, need(q^2_1)\right) = \emptyset$ because even though $\left(q^0_2, q^1_1\right)$

is reachable in the corresponding subsystem there is no port in $\overline{\mathcal{A}_2}$ that can be used to enter it. The analogous result is obtained when we switch the order of the states in the other sets of entry actions computed above. On the other hand, we get $\mathcal{EA}\left(q_1^2, \{2\}, q_2^0, \{1\}\right) = \{b_1^{\alpha_3}\}$. The only ports that can possibly be contained in this set are the ports in $\mathit{comm}_1(2) = \{a_1^{\alpha_1}, b_1^{\alpha_3}\}$. We have already seen above that $BWS\left(q_1^2, \{2\}\right) = Q_1$ and that $BWS(q_2^0, \{1\}) = \{q_2^0\}$. In $\overline{Sys_3}\downarrow_{\{1,2\}}$ the state $\left(q_3^1, q_2^0\right)$ is reached by performing $\{b_1^{\alpha_3}, b_2^{\alpha_3}\}$. Therefore $b_1^{\alpha_3} \in \mathcal{EA}\left(q_1^2, \{2\}, q_2^0, \{1\}\right)$. The port $a_1^{\alpha_1}$ is not contained in the set of entry actions. Whenever it is executed in $\overline{Sys_3}\downarrow_{\{1,2\}}$ it must synchronize with $a_2^{\alpha_1}$ causing component 2 to move to q_2^1. For local states of components other than 1 the entry actions are computed likewise.

We point out one detail: Let $i \in \overline{K_3}\setminus\{1\}$. Then i only communicates with 1. We get $\mathit{comm}_i(1) = \mathcal{A}_i$ and $\mathit{comm}_i(j) = \emptyset$ for any other component j. Computing a set of entry actions only those ports that are for communication with one of the components in K', i.e., the ports in $\bigcup_{k\in K'} \mathit{comm}_i(k)$, are eligible. For $1 \notin K'$ we therefore always get $\mathcal{EA}(q_i, K', Q'_j, \mathfrak{K}(Q'_j)) = \emptyset$ for any choice of q_i, Q'_j, and $\mathfrak{K}(Q'_j)$.

2. Next, consider $\overline{Sys_2}$ (cf. Figure 3.5 (p. 43)). For $j \in \{6, 7\}$ we obtain $\mathcal{EA}(q_5^0, \mathit{need}(q_5^0), q_j^1, \mathit{need}(q_j^1)) = \emptyset$. We have already computed the necessary BWS-sets above. Indeed, the state (q_5^0, q_j^1) is reachable in the corresponding subsystem, but it can *only* be reached by performing the port $a_j^{\beta_2}$. It cannot be reached by any port of component 5. Therefore the set of entry actions is empty. We get a different result by changing the order of the states: $\mathcal{EA}(q_j^1, \mathit{need}(q_j^1), q_5^0, \mathit{need}(q_5^0)) = \{a_j^{\beta_2}\}$. Analogously, we get $\mathcal{EA}(q_5^1, \mathit{need}(q_5^1), q_j^0, \mathit{need}(q_j^0)) = \emptyset$ on the one hand, and $\mathcal{EA}(q_j^0, \mathit{need}(q_j^0), q_5^1, \mathit{need}(q_5^1)) = \{b_j^{\beta_4}\}$, on the other. Replacing the local state of 5 in the computations above by the corresponding local state of 8 we get analogous results where the two nonempty sets of entry actions are equal to $\{a_j^{\beta_1}\}$ respectively $\{b_j^{\beta_3}\}$.

The following lemma summarizes the observations made for the example we used to motivate the backward-search and the entry actions. It shows

how observing the entry actions can help to exclude the possibility that global states are reachable if K' and $\mathfrak{K}(Q'_j)$ are chosen appropriately.

Lemma 3.2.3. *Let Sys be an interaction system with strongly exclusive communication. Let* $i \in K$, $\tilde{K} \subseteq K \setminus \{i\}$, *and* q_i *such that* $q_i^0 \notin BWS(q_i, \tilde{K})$. *For each* $j \in \tilde{K}$ *let* $Q'_j \subseteq Q_j$ *be a nonempty subset of local states.*

If $\bigcap_{j\in\tilde{K}} \mathcal{EA}(q_i, \tilde{K}, Q'_j, \{i\}) = \emptyset$ *then there is no* $q \in reach\,(Sys)$ *with* $q \downarrow_{(\{i\}\cup\tilde{K})} \in \{q_i\} \times \prod_{j\in\tilde{K}} Q'_j$.

Taking the intersection of the sets of entry actions in the lemma corresponds to the comparison between the ports that could be used to reach certain states of subsystems of Sys_3 that concluded the informal argument before the introduction of *BWS*.

Example 3.2.1 continued: We formally substantiate the observations made for $\overline{Sys_3}$ on pp. 44ff. We want to know if there is $q \in reach(\overline{Sys_3})$ with $q \downarrow_{\{1,2,3\}} = \left(q_1^2, q_2^0, q_3^0\right)$. We apply the lemma with $\tilde{K} = \{2,3\}$ and $Q'_j = \{q_j^0\}$ for $j \in \tilde{K}$. We already know $\mathcal{EA}\left(q_1^2, need(q_1^2), q_2^0, need(q_2^0)\right) = \{a_1^{\alpha_2}\}$ respectively $\mathcal{EA}\left(q_1^2, need(q_1^2), q_3^0, need(q_3^0)\right) = \{a_1^{\alpha_1}\}$ (note that $\tilde{K} = \{2,3\} = need(q_1^2)$ and $need(q_2^0) = need(q_3^0) = \{1\}$). The intersection of these sets is empty and the lemma implies that no such global state is reachable.

3.3 Architectural Patterns

In this section we introduce the architectural pattern defining the subclass of tree-like interaction systems for which we will exhibit a sufficient condition for deadlock-freedom. The pattern only refers to the *interaction model* of an interaction system, i.e., it restricts the way in which the components may interact. For an interaction model we introduce two (undirected) interaction graphs named G^* and G. They give a simple description of the possible interactions between the components. In a first intuitive approach we simply represent each component by a node, and we introduce an edge between two components if they interact. This is the definition of G^* which describes all *direct* cooperations that are allowed between the components. Requiring G^* to be a tree results in a sufficiently interesting class of interac-

tion systems. On the other hand, no system with interactions involving more than two components falls into this class. Following the informal description of G^* it is clear that every interaction results in a complete subgraph of G^*. Such a subgraph contains cycles if it involves more than two components. Thus, an interaction model for which G^* is a tree only contains singleton or binary interactions which is a rather strong limitation. This motivates the definition of the more general graph G which allows the definition of tree-like architectures even for systems with multiway cooperation. Again there is one node for every component. Further, we also introduce a second type of node. These nodes represent sets of components which are obtained by intersecting the sets of components participating in two (not necessarily distinct) interactions. Roughly speaking, the edges of G represent "subset inclusion" of sets of components.

Definition 3.3.1. *Let IM be an interaction model.*

1. *Define $G := (V, E)$. The set of nodes is defined by $V := V_1 \cup V_2$ where $V_1 := \{\{i\} | i \in K\}$ and $V_2 := \{comp(\alpha) \cap comp(\alpha') \,|\alpha, \alpha' \in Int$ and $comp(\alpha) \cap comp(\alpha') \neq \emptyset\}$. For $K', K'' \in V$ there is an edge $\{K', K''\} \in E$ if and only if $K' \subsetneq K''$ and there is no $\bar{K} \in V$ with $K' \subsetneq \bar{K} \subsetneq K''$. G is the **interaction graph of** IM. If Sys is an interaction system we also speak of the **interaction graph of** Sys. If G is a tree we say that IM respectively Sys is **tree-like**.*

2. *Define $G^* := (K, E^*)$. For $i, j \in K$ there is an edge $\{i, j\} \in E^*$ if and only if there is an interaction $\alpha \in Int$ with $i, j \in comp(\alpha)$. If G^* is a tree we say that IM is **strongly tree-like**. Sys is a **strongly tree-like** interaction system if G^* is a tree for IM.*

Example 3.2.1 continued: G and G^* of $\overline{Sys_3}$ (cf. Figure 3.4 (p. 43) respectively Figure 3.6 (p. 45)) are depicted in Figure 3.7. $\overline{Sys_3}$ is strongly tree-like. For each $\alpha \in \overline{Int_3}$ there is a node $comp(\alpha)$ in G. This is true in general, because the definition of V_2 does not exclude the possibility $\alpha = \alpha'$.

Figure 3.8 a) shows G^* of $\overline{Sys_2}$ (cf. Figure 3.5 (p. 43)). The system is not strongly tree-like because there are interactions involving three components. $\overline{Sys_2}$ is tree-like, though, as can be seen in Figure 3.8 b). Note that there is

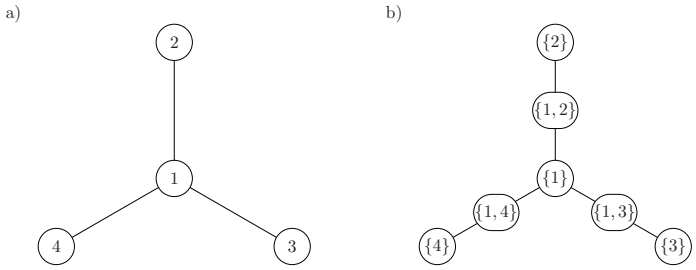

Figure 3.7: a) G^* and b) G for $\overline{Sys_3}$

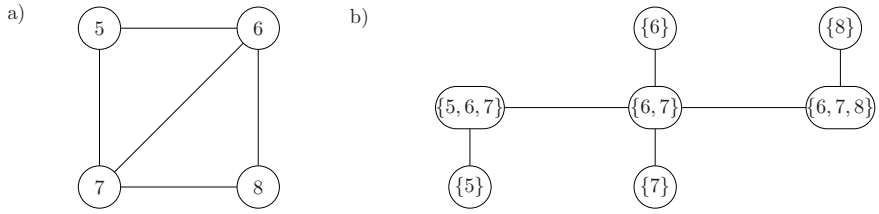

Figure 3.8: a) G^* and b) G for $\overline{Sys_2}$

a node $\{6, 7\} \in V_2$ even though there is no $\alpha \in \overline{Int_2}$ with $comp(\alpha) = \{6, 7\}$. This node stems from the intersection of the two nodes representing $\{5, 6, 7\}$ and $\{6, 7, 8\}$.

In the following we state some lemmas that summarize a few simple results about the interrelations between the two graphs and the consequences of the definition. We first formalize the observation that every interaction in a strongly tree-like interaction model involves two components at most.

Lemma 3.3.1. *Let IM be an interaction model.*

If IM is strongly tree-like then $|\alpha| \leq 2$ *for all* $\alpha \in Int$.

The lemma justifies the distinction between G^* and G. If we only considered (the more simple) strongly tree-like interaction models there would be systems which we would not be able to treat even though they have a natural tree-like structure. It is possible to transform an arbitrary interaction system to an interaction system where all interactions involve at most two components, but this approach is not feasible in the context of tree-like communication architectures: In general, the techniques used to transform

a system result in systems for which G^* contains cycles even if G is a tree. Thus, in order to be able to speak about tree-like architectures for systems where interactions of arbitrary size are allowed we need G.

It is intuitively clear that a strongly tree-like interaction model is tree-like and that a tree-like interaction model where all interactions are of size one or two is strongly tree-like:

Lemma 3.3.2. *Let IM be an interaction model.*

1. *If IM is strongly tree-like then it is tree-like.*
2. *If for all $\alpha \in Int$ we have $|\alpha| \leq 2$ and IM is tree-like then IM is strongly tree-like.*

The first part of the following lemma formalizes the observation that for all $\alpha \in Int$ there is a node $comp(\alpha) \in V_2$. Further, the lemma describes the structure of paths in G between interacting components. In general, there can be several such paths. If G is a tree, though, the paths are unique.

Lemma 3.3.3. *Let IM be an interaction model.*

1. *For all $\alpha \in Int$ there is a node $comp(\alpha) \in V_2$.*
2. *Let $\alpha \in Int$ and $i \in comp(\alpha)$. There is a simple path $\pi_{i,\alpha}$ connecting $\{i\}$ and $comp(\alpha)$ in G. All nodes on $\pi_{i,\alpha}$ are subsets of $comp(\alpha)$. If $|comp(\alpha)| \geq 2$ then all nodes on $\pi_{i,\alpha}$ except $\{i\}$ contain two components at least.*
3. *Let $\alpha \in Int$ and $i, j \in comp(\alpha)$. There is a simple path $\pi^{\alpha}_{i,j}$ connecting $\{i\}$ and $\{j\}$ in G. All nodes on $\pi^{\alpha}_{i,j}$ are subsets of $comp(\alpha)$. All nodes on $\pi_{i,\alpha}$ except $\{i\}$ and $\{j\}$ contain two components at least. The paths $\pi^{\alpha}_{i,j}$, $\pi_{i,\alpha}$, and $\pi_{j,\alpha}$ can be chosen such that there is a node K' on $\pi^{\alpha}_{i,j}$ such that the sub-path of $\pi^{\alpha}_{i,j}$ from $\{i\}$ to K' is a sub-path of $\pi_{i,\alpha}$ and the sub-path of $\pi^{\alpha}_{i,j}$ from K' to $\{j\}$ is a sub-path of $\pi_{j,\alpha}$.*
4. *Let $\alpha, \alpha' \in Int$ with $|comp(\alpha) \cap comp(\alpha')| \geq 2$ and let $i \in comp(\alpha)$ and $j \in comp(\alpha')$. There is a simple path $\pi^{\alpha,\alpha'}_{i,j}$ connecting $\{i\}$ and $\{j\}$ in G. All nodes on $\pi^{\alpha,\alpha'}_{i,j}$ are subsets of $comp(\alpha)$ or $comp(\alpha')$. The first node after $\{i\}$ is contained in $comp(\alpha)$ and the last node before*

$\{j\}$ is contained in comp(α'). All nodes on $\pi_{i,j}^{\alpha,\alpha'}$ except $\{i\}$ and $\{j\}$ contain two components at least.

The motivation for using a tree-like architecture was the prospect of only having to investigate two components at a time. Strictly speaking, we do not need the assumption that G respectively G^* are connected for this purpose. It suffices to require that the graphs are free of cycles. However, an interaction model whose interaction graph is not connected describes several independent systems corresponding to the connected subgraphs. It would make sense to decompose such a system into its independent subsystems and to apply the results to each of these subsystems. Thus, we do not infringe on generality by assuming that G is connected.

3.4 Deadlock-Freedom for Tree-Like Component Architectures

We will now develop a methodology for checking deadlock-freedom of tree-like interaction systems. It allows to derive information about the *global* property deadlock-freedom by means of a compositional analysis which only investigates the system *locally*, i.e., subsystems of size two. We present a sufficient condition for deadlock-freedom. One might ask why this is even necessary. If we only consider this subclass of interaction systems it might be the case that there is an efficient decision procedure for deadlock-freedom. Results presented by Semmelrock and Majster-Cederbaum [128] show that this is not the case: Even for strongly tree-like interaction systems deciding deadlock-freedom is PSPACE-complete.

This section contains two subsections. The first subsection develops a criterion for deadlock-freedom of tree-like interaction systems. In the second subsection we show how the notions and conditions can be simplified to obtain a criterion for strongly tree-like systems.

3.4.1 Interaction Systems with Multiway Cooperation

The requirement that the interaction graph should be a tree restricts the architecture of the *interaction model*. Using this extra information we

will now exhibit conditions that take hold at the level of the *local transition systems* of the components. These conditions help to detect combinations of local states that could result in deadlocks. This separation of concerns reflects the two layers of description for an interaction system mentioned in Chapter 2. The layers help to take a precise approach to solving the problem of checking deadlock-freedom. In this subsection *Sys* always denotes a tree-like interaction system with strongly exclusive communication.

Since we only want to investigate pairs of interacting components, we first have to characterize those pairs of states that may possibly be involved in a deadlock. Once we have found a suitable characterization of such a "problematic" pair of states we use the techniques presented in Section 3.2 to check whether combinations of such pairs which could result in deadlocks are possibly reachable in Sys. If this is not the case we conclude that Sys is deadlock-free. Since all conditions only refer to subsystems of size two the overall procedure causes cost polynomial in the size of *Sys*.

The notion of problematic states is central to our considerations. Before giving the formal definition, we want to motivate how this notion evolves from the architectural constraint by means of the following remark.

Remark 3.4.1. *Consider an interaction system Sys and a deadlock in the global state q. Because of the assumption that every local state of every component enables at least one port, every component i is ready to participate in at least one interaction α. On the other hand, because q is a deadlock, for every α there is $j \in comp(\alpha)$ such that q_j does* not *enable $j(\alpha)$. Thus, it is possible to construct sequences*

1. *of components $i_0, i_1, \ldots, i_{r-1}$ and*
2. *of interactions $\alpha_0, \alpha_1, \ldots, \alpha_{r-1}$*

such that for all $0 \leq s \leq r-1$ we have (writing $i_r = i_0$ and $\alpha_r = \alpha_0$):

- *$\alpha_s \in Int(q_{i_s})$, i.e., i_s is ready to perform α_s in the local state q_{i_s},*
- *$i_{s+1}(\alpha_s) \neq \emptyset$, i.e., i_{s+1} is needed by i_s in order to perform α_s,*
- *but $i_{s+1}(\alpha_s) \not\subseteq en\left(q_{i_{s+1}}\right)$, i.e., i_{s+1} does not enable its port in α_s.*

For simplicity, we say that in q_{i_s} component i_s waits for i_{s+1} with respect to α_s. Informally speaking, these sequences constitute a cycle of waiting relations between the components. Such a cycle is depicted in Figure 3.9 a).

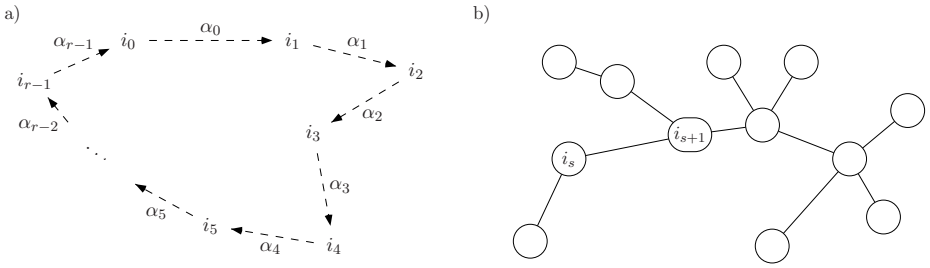

Figure 3.9: a) Cycle of waiting relations, b) Part of G^*

In order to accentuate the idea that we want to motivate we take a short digression by considering strongly *tree-like interaction systems first. We want to retrace the cycle in G^*. Therefore, we contrast it with G^* in Figure 3.9* b). *Because the system is strongly tree-like the cycle is represented by edges in G^* that are first traveled in one direction and later in the opposite direction. This is schematically depicted in Figure 3.10. In particular, it can*

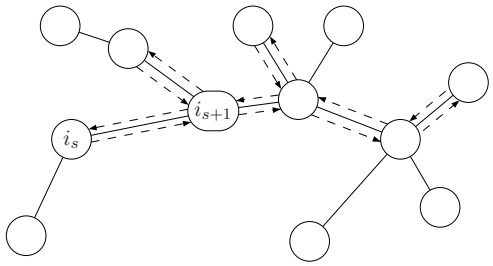

Figure 3.10: Superimposing the images in Figure 3.9

be seen that there must be components i_s and i_{s+1} that wait for each other, i.e., $i_s \in need(q_{i_{s+1}})$ and vice versa, but no interaction is enabled because the ports that are offered are not compatible. The existence of such a pair of components is a necessary *condition for q to be a deadlock in Sys if the system is strongly tree-like. We now elaborate on this observation in order to derive a similar condition for tree-like interaction systems.*

Before continuing, note that the sequences above can be chosen such that the components respectively the interactions are pairwise distinct. As above, we retrace the cycle of waiting relations in G. Figure 3.11 a) shows the part of the cycle that is concerned with i_s and i_{s+1}. Figure 3.11 b) depicts part of the interaction graph[1]*. Superimposing the two images in Figure 3.12, we*

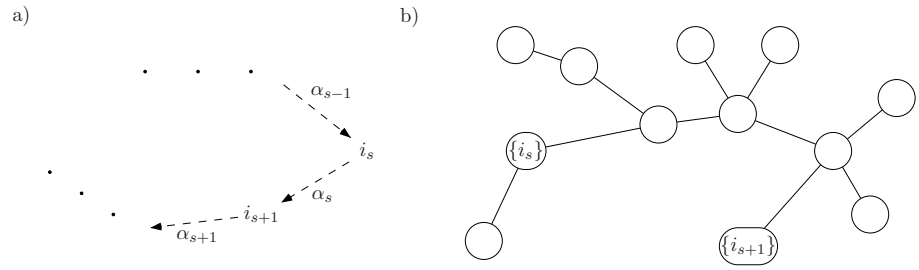

Figure 3.11: a) Part of the cycle, b) Part of G

schematically represent the fact that i_s waits for i_{s+1} because of α_s by a directed path which follows the unique *simple non-directed path between $\{i_s\}$ and $\{i_{s+1}\}$ in G. The set K' that has been highlighted contains two*

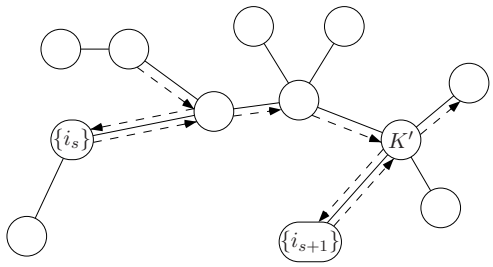

Figure 3.12: Superimposing the images in Figure 3.11

components at least, and it is contained in comp(α_s) and *in comp*(α_{s+1}).

We will formalize this observation. Lemma 3.3.3 shows that for each i_s and i_{s+1} there exists a simple path $\pi^{\alpha_s}_{i_s,i_{s+1}}$ in the interaction graph of Sys connecting $\{i_s\}$ and $\{i_{s+1}\}$. It only visits nodes (except $\{i_s\}$ and $\{i_{s+1}\}$) that contain two components at least and that are subsets of comp (α_s). Denote

[1]The nodes of G are sets of components which do not necessarily contain only one component.

this path by π_s. *Walking the paths* π_s *consecutively for* $s = 0, \ldots, r-1$ *we obtain a path* π *in* G *which starts in* $\{i_0\}$, *visits all components above exactly once, and then returns to* $\{i_0\}$. *In between the nodes representing components this path only visits nodes that contain two components at least. We consider*

$$\pi_s = \{i_s\} \text{ — } \ldots \text{ — } K' \overset{e'}{\text{—}} \{i_{s+1}\}$$

and

$$\pi_{s+1} = \{i_{s+1}\} \overset{e''}{\text{—}} K'' \text{ — } \ldots \text{ — } \{i_{s+2}\},$$

two consecutive parts of π. *We argue that* e' — *the last edge on* π_s — *and* e'' — *the first edge on* π_{s+1} — *must be equal. Assume that this is not the case. Starting from* $\{i_{s+1}\}$, *we walk* π *in either direction (the first edge traveled in one direction is* e' *and the first edge traveled in the other direction is* e''*). We denote by* $\bar{K}$ *the first node that occurs on both paths. It is clear that* $\bar{K}$ *exists because both paths end in* $\{i_0\}$. *We obtain two paths connecting* $\{i_{s+1}\}$ *and* $\bar{K}$. *From these two paths we can construct simple paths* π' *and* π'' *connecting* $\{i_{s+1}\}$ *and* $\bar{K}$. *Note that* π' *and* π'' *do not share any nodes other than* $\{i_{s+1}\}$ *and* $\bar{K}$ *because* $\bar{K}$ *was chosen to be the first node occurring on both paths above. Because the sequence of components was chosen such that the components are pairwise distinct the first edge of* π' *must be* e' *and the first edge of* π'' *must be* e''. *Since these edges are different from each other by assumption we see that* π' *and* π'' *are also different from each other. This is a contradiction because* G *is a tree. We see that* $e' = e''$ *and in particular* $K' = K''$. *By construction of* π_s *respectively* π_{s+1} *we know that this node is contained in* $comp(\alpha_s)$ *as well as in* $comp(\alpha_{s+1})$. *We conclude* $|comp(\alpha_s) \cap comp(\alpha_{s+1})| \geq 2$.

Summarizing, we see that if *Sys is tree-like* and a *deadlock exists in Sys then there must be sequences as above where a) each component waits for the successive component because of the corresponding interaction and b) two consecutive interactions have at least two components in common.*

Negating this necessary condition for q to be a deadlock, yields a sufficient condition which ensures for a local state of a component that it cannot be part of a deadlock state: If we can show the following Condition 3.4.1 for a local state q_i then we can be sure that q_i will never be part of a cycle of

waiting conditions as in the remark.

(3.4.1) for all $\alpha \in \mathit{Int}(q_i)$ and all $j \in \mathit{comp}(\alpha)\backslash\{i\}$ no q_j exists such that in (q_i, q_j) component i waits for j because of α and such that there is some $\alpha' \in \mathit{Int}(q_j)$ with $|\mathit{comp}(\alpha) \cap \mathit{comp}(\alpha')| \geq 2$

Given Condition 3.4.1 q_i will never *cause* a global deadlock (by participating in a cycle as above). Note that q_i may *be part* of a global deadlock all the same, because it is possible that i waits for other components which are part of a cycle as above. However, for our purposes it suffices to identify for each component those states that may cause a deadlock in the sense above. Condition 3.4.1 is still rather restrictive as it requests that for q_i *no* interaction $\alpha \in \mathit{Int}(q_i)$ can ever be part of a cycle of waiting relations as above. In order to make sure that q_i never causes a deadlock in a global state q it actually suffices to ensure that for each q_i there is *at least* one interaction $\alpha \in \mathit{Int}(q_i)$ that can never be part of such a cycle. Thus, for $i \in K$ and $q_i \in Q_i$ we may substitute the condition by Condition 3.4.2 below where we replace the universal quantifier for $\alpha \in \mathit{Int}(q_i)$ by an existential quantifier:

(3.4.2) there is $\alpha \in \mathit{Int}(q_i)$ such that for all $j \in \mathit{comp}(\alpha)\backslash\{i\}$ no q_j exists such that in (q_i, q_j) component i waits for j because of α and such that there is $\alpha' \in \mathit{Int}(q_j)$ with $|\mathit{comp}(\alpha) \cap \mathit{comp}(\alpha')| \geq 2$

Note that j participates in α as well as in α'. Thus, the requirement that there should be an interaction $\alpha' \in \mathit{Int}(q_j)$ with $|\mathit{comp}(\alpha) \cap \mathit{comp}(\alpha')| \geq 2$ is equivalent to demanding that there should be $\alpha' \in \mathit{Int}(q_j)$ such that at least one component other than j participates in α and in α'. Finally, since $\alpha' \in \mathit{Int}(q_j)$ this requirement is equivalent to $\mathit{need}(q_j) \cap \mathit{comp}(\alpha) \neq \emptyset$. This last condition is used in the definition below. In a first approach for q_i, $\alpha \in \mathit{Int}(q_i)$, and $j \in \mathit{comp}(\alpha)\backslash\{i\}$ a state q_j could be called problematic with respect to q_i and α if $\mathit{need}(q_j) \cap \mathit{comp}(\alpha) \neq \emptyset$ and in q_i component i waits for j because of α. However, it can be seen that a state q_j is not problematic with respect to q_i and α after all if there is at least one interaction $\beta \in \mathit{Int}(q_j)$ for which no problematic states exist (for none of the components in $\mathit{comp}(\beta)$). Starting with those states of j that violate Condition 3.4.2, we define for q_i, $\alpha \in \mathit{Int}(q_i)$, and $j \in \mathit{comp}(\alpha)$ a descending sequence of superscripted sets

of problematic states by eliminating those states q_j that do not satisfy this additional requirement. The intersection of these sets is then defined to be the set of states of j that are problematic with respect to q_i and α.

Definition 3.4.1. *Let Sys be a tree-like interaction system. For $i \in K$, $q_i \in Q_i$ incomplete, $\alpha \in Int(q_i)$ and $j \in comp(\alpha) \setminus \{i\}$ we inductively define a descending sequence of sets by:*

$$\begin{aligned} PS_j^0(q_i,\alpha) := \{q_j \mid & \bullet\ q_j \text{ incomplete} \\ & \bullet\ need(q_j) \cap comp(\alpha) \neq \emptyset \\ & \bullet\ (q_i,q_j) \text{ is reachable in } Sys\downarrow_{\{i,j\}} \\ & \bullet\ j(\alpha) \not\subseteq \mathcal{en}(q_j) \\ & \bullet\ \nexists \tilde{\alpha} \in Int(q_i) \cap Int(q_j) \text{ with } |\tilde{\alpha}| = 2\} \end{aligned}$$

$$\begin{aligned} PS_j^{l+1}(q_i,\alpha) := \{q_j | & q_j \in PS_j^l(q_i,\alpha) \text{ and } \forall \beta \in Int(q_j) \exists k \in comp(\beta) \setminus \{j\} \\ & \text{with } PS_k^l(q_j,\beta) \neq \emptyset\} \end{aligned}$$

The set of states of j that are ***problematic with respect to*** *q_i* ***and*** *α is*

$$PS_j(q_i,\alpha) := \bigcap_{l \in \mathbb{N}} PS_j^l(q_i,\alpha).$$

We obtain a descending sequence of sets. Furthermore, there is an index l for which the sequence becomes stationary such that the computation can be stopped. By way of imprecision we sometimes also say that a state contained in one of the superscripted sets is problematic when there is no danger of resulting confusion. The definition for the 0-th instance of the set of problematic states of component j restates the conditions derived above. We further elaborated this condition in two ways. First, we require that (q_i, q_j) should be reachable in the subsystem consisting of i and j. If this is not the case Lemma 3.2.2 shows that there is no global state $q \in reach(Sys)$ with $q \downarrow_{\{i,j\}} = (q_i, q_j)$. Therefore this pair of states will not cause a reachable deadlock. Second, we treat a pair of states offering an interaction of the global system of size one or two differently. Complete local states are excluded from the definition. It is clear that a global state q such that q_i is complete for at least one component i is not a deadlock. The conditions for $\tilde{\alpha}$ simply say that there must not be an interaction which

only involves i and j and which is enabled in (q_i, q_j). Again it is clear that such a pair of states would never be part of a deadlock because $\tilde{\alpha}$ could be performed globally. We cannot argue analogously for interactions of size greater than two if i and j both enable their part of the interaction because we cannot be sure whether all other required components enable their ports.

Example 3.2.1 continued: Consider $\overline{Sys_2}$ (cf. Figure 3.5 (p. 43)). The only interaction in $Int(q_5^0)$ is $\overline{\beta_1}$. Therefore $need(q_5^0) = \{6, 7\}$. Let $j \in \{6, 7\}$. The state (q_5^0, q_j^1) is reachable in $\overline{Sys_2} \downarrow_{\{5,j\}}$ and $j\left(\overline{\beta_1}\right) = \{a_j^{\beta_1}\}$ is not enabled. We have $5 \in need(q_j^1)$, and therefore $need(q_j^1) \cap comp(\overline{\beta_1}) \neq \emptyset$ (depending on the choice of j component 6 or 7 is also contained in that intersection). Finally, no global interaction of size one or two is enabled in (q_5^0, q_j^1). We conclude $q_j^1 \in PS_j^0\left(q_5^0, \overline{\beta_1}\right)$. It is the only state in this set because $j\left(\overline{\beta_1}\right) \subseteq \mathcal{en}(q_j^0)$. The other states can be treated likewise. For $j \in \{6, 7\}$ we have $PS_j^0(q_5^1, \overline{\beta_3}) = \{q_j^0\}$, and analogously $PS_j^0\left(q_8^0, \overline{\beta_2}\right) = \{q_j^1\}$ and $PS_j^0(q_8^1, \overline{\beta_4}) = \{q_j^0\}$. Further, $PS_5^0(q_j^0, \overline{\beta_1}) = \left\{q_5^1\right\}$ and $PS_5^0(q_j^1, \overline{\beta_3}) = \left\{q_5^0\right\}$ respectively $PS_8^0(q_j^0, \overline{\beta_2}) = \left\{q_8^1\right\}$ and $PS_8^0(q_j^1, \overline{\beta_4}) = \left\{q_8^0\right\}$. Note that $PS_7^0(q_6, \overline{\beta}) = PS_6^0(q_7, \overline{\beta}) = \emptyset$ for any $q_6 \in Q_6$ and $\overline{\beta} \in Int(q_6)$ respectively $q_7 \in Q_7$ and $\overline{\beta} \in Int(q_7)$. Treating q_6^0 and $\overline{\beta_1} \in Int(q_6^0)$, for example, we compute $PS_7^0(q_6^0, \overline{\beta_1})$. The components 6 and 7 always have to cooperate. Therefore, the only reachable states in $\overline{Sys_2} \downarrow_{\{6,7\}}$ are $\left(q_6^0, q_7^0\right)$ and $\left(q_6^1, q_7^1\right)$. The conditions of the definition are not satisfied for q_7^0. Thus, it is not in $PS_7^0(q_6^0, \overline{\beta_1})$, and this set is empty.

For $\overline{Sys_2}$ iterating the computation of the problematic states does not have any effect: Even though some sets of problematic states of the 0-th stage are empty, for every q_j and $\beta \in Int(q_j)$ there is at least one component in $comp(\beta)\backslash\{j\}$ for which the 0-th stage set of problematic states is not empty. Thus, for all combinations of local states and interactions the 0-th stage set of problematic states is equal to the final set of problematic states.

In the following we combine the concepts about reachability developed in Section 3.2 with the notion of problematic states. We merge the notions by computing the entry actions with respect to q_i and $PS_j(q_i, \alpha)$ for $\alpha \in Int(q_i)$ and $j \in comp(\alpha) \setminus \{i\}$. We then use an idea similar to the one suggested in Lemma 3.2.3 to make sure that no combinations of states are

reachable where for all interactions in $Int(q_i)$ at least one of the needed communication partners is in a problematic state with respect to q_i and the corresponding interaction. All interactions in $Int(q_i)$ are blocked in such a combination of states, and therefore it might ultimately cause a deadlock. In the following definition the BWS-operator for q_i and $PS_j(q_i, \alpha)$ is parametrized with $need(q_i)$ respectively $need(q_j)$ for every $q_j \in PS_j(q_i, \alpha)$. We use the following notation for a subset $Q'_j \subseteq Q_j$:

$$\mathfrak{need}\left(Q'_j\right) := \{need(q_j)\}_{q_j \in Q'_j}$$

We call the entry actions computed with respect to q_i and $PS_j(q_i, \alpha)$ the actions that are problematic with respect to q_i, α, and j because they may be used to reach $BWS(q_i, need(q_i)) \times BWS(PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$ in $Sys\downarrow_{\{i,j\}}$. Once the system has reached a state in this set we have to assume that it is also possible to reach a state in $\{q_i\} \times PS_j(q_i, \alpha)$.

Definition 3.4.2. *Let Sys be a tree-like interaction system with strongly exclusive communication. Let $i \in K$, $q_i \in Q_i$ incomplete, $\alpha \in Int(q_i)$, and $j \in comp(\alpha) \setminus \{i\}$. We define*

$$\mathcal{PA}(q_i, \alpha, j) := \mathcal{EA}(q_i, need(q_i), PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$$

the set of ***problematic actions of i with respect to q_i, α, and j.***

Example 3.2.1 continued: Using the results gathered above we can now specify the sets of problematic actions for $\overline{Sys_2}$ (cf. Figure 3.5 (p. 43)). It becomes apparent now, that the combinations of states for which we computed the BWS-operator and the entry actions (cf. pp. 47 and 49) were those combinations that are relevant for stating the problematic actions. It is clear that $\mathcal{PA}\left(q_6, \overline{\beta}, 7\right) = \mathcal{PA}\left(q_7, \overline{\beta}, 6\right) = \emptyset$ for any $q_6 \in Q_6$ and $\overline{\beta} \in Int(q_6)$ respectively $q_7 \in Q_7$ and $\overline{\beta} \in Int(q_7)$ because the corresponding sets of problematic states are empty. For $q_5 \in Q_5$ and $\overline{\beta} \in Int(q_5)$ respectively $q_8 \in Q_8$ and $\overline{\beta} \in Int(q_8)$ and $j \in \{6, 7\}$ we get $\mathcal{PA}\left(q_5, \overline{\beta}, j\right) = \mathcal{PA}\left(q_8, \overline{\beta}, j\right) = \emptyset$ because the corresponding sets of entry actions are empty. Switching the roles of the components we obtain $\mathcal{PA}(q_j^0, \overline{\beta_1}, 5) = \{b_j^{\beta_4}\}$, $\mathcal{PA}(q_j^0, \overline{\beta_2}, 8) = \{b_j^{\beta_3}\}$, $\mathcal{PA}(q_j^1, \overline{\beta_3}, 5) = \{a_j^{\beta_2}\}$, and $\mathcal{PA}(q_j^1, \overline{\beta_4}, 8) = \{a_j^{\beta_1}\}$ where $j \in \{6, 7\}$.

Finally, we explain how to combine all these notions in order to derive information about deadlock-freedom. Consider an arbitrary local state q_i. An interaction $\alpha \in \mathit{Int}(q_i)$ is blocked if *at least one* of the components j in $\mathit{comp}(\alpha) \setminus \{i\}$ is in a local state q_j with $j(\alpha) \not\subseteq \mathcal{en}(q_j)$. Therefore, when we want to know which actions of i might lead to a state where α is blocked we have to take the *union* of all sets $\mathcal{PA}(q_i, \alpha, j)$ where j ranges over $\mathit{comp}(\alpha) \setminus \{i\}$. A state q_i can only be part of a global deadlock if *all* interactions in $\mathit{Int}(q_i)$ are blocked. This can only be the case if there is a port of i that is problematic with respect to all $\alpha \in \mathit{Int}(q_i)$ and at least one $j \in \mathit{comp}(\alpha) \setminus \{i\}$. If there is no such action then the global system will never be able to reach a state where all $\alpha \in \mathit{Int}(q_i)$ are blocked and q_i will never cause a global deadlock. Therefore, we take the *intersection* over all $\alpha \in \mathit{Int}(q_i)$ of the unions mentioned above. This intersection contains those ports of i that may lead to a state where all $\alpha \in \mathit{Int}(q_i)$ are blocked.

The first condition below caters for the case that there are local states q_i with $q_i^0 \in BWS(q_i, \mathit{need}(q_i))$. This situation has to be treated separately because the reachability of q_i in combination with problematic states cannot be restricted by the consistency check for problematic actions described above because q_i may be reached without performing any such action.

Proposition 3.4.1. *Let Sys be a tree-like interaction system with strongly exclusive communication.*

If the following two conditions hold then Sys is deadlock-free:

1. $\forall i \forall q_i : q_i$ *complete* $\vee\, q_i^0 \notin BWS(q_i, \mathit{need}(q_i)) \vee \exists \alpha \in \mathit{Int}(q_i)$ *such that* $\forall j \in \mathit{comp}(\alpha) \setminus \{i\} : q_j^0 \notin BWS(PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$
2. $\forall \tilde{\alpha} \in \mathit{Int}$ *with* $|\mathit{comp}(\tilde{\alpha})| \geq 2\ \exists i \in \mathit{comp}(\tilde{\alpha})$ *such that for all* $q_i \in Q_i$ *that are incomplete:*

$$i(\tilde{\alpha}) \not\subseteq \bigcap_{\alpha \in \mathit{Int}(q_i)} \bigcup_{j \in \mathit{comp}(\alpha) \setminus \{i\}} \mathcal{PA}(q_i, \alpha, j)$$

The conditions can be checked in time polynomial in the size of Sys.

Example 3.4.1. Example: Proposition 3.4.1 shows that $\overline{Sys_2}$ (cf. Figure 3.5 (p. 43)) is deadlock-free. Recall that for all local states q_i of all components $i \in \overline{K_2}$ we computed $BWS(q_i, \mathit{need}(q_i)) = \{q_i\}$ (cf. p. 47). Thus, the

first condition above is definitely satisfied for all local states other than the local initial states. On the other hand, consider $i \in \overline{K_2}$, q_i^0, and $\overline{\beta} \in \mathit{Int}(q_i^0)$. We have already seen above that for any $j \in \mathit{comp}(\overline{\beta})$ and $q_j \in PS_j(q_i^0, \overline{\beta})$ we have $q_j \neq q_j^0$. Thus, the first condition also holds for q_i^0.

Considering the sets of problematic actions computed above we see that all the intersections that have to be computed for the second condition are empty. Therefore, this condition is also satisfied.

3.4.2 Strongly Tree-Like Interaction Systems

In this subsection we address deadlock-freedom of strongly tree-like interaction systems. Lemma 3.3.2 showed that a strongly tree-like interaction system is tree-like. Therefore we can apply Proposition 3.4.1 to strongly tree-like systems. We will see now that restricting the interactions to size two or one results in a further simplification of the notions. We formulate an equivalent but more simple version of Proposition 3.4.1 for strongly tree-like interaction systems. In this subsection *Sys* denotes a strongly tree-like interaction system with strongly exclusive communication. For this kind of system, a component i, and two other components $j \neq k$ that interact with i we get $\mathit{comm}_i(j) \cap \mathit{comm}_i(k) = \emptyset$ because for every interaction α with $i(\alpha) \neq \emptyset$ there is at most one other component that also participates in α. Since all interactions are pairwise disjoint $i(\alpha)$ can only be contained in at most one set $\mathit{comm}_i(j)$.

We define a notion of problematic states which is more simple than the one from Definition 3.4.1. In particular, it is independent of $\alpha \in \mathit{Int}(q_i)$.

Definition 3.4.3. *Let Sys be a strongly tree-like interaction system. For $i \in K$, $q_i \in Q_i$ incomplete, and $j \in \mathit{need}(q_i)$ we inductively define a descending sequence of sets by:*

$$\begin{aligned} PS_j^0(q_i) = \{q_j \mid & \bullet\ q_j \text{ incomplete} \\ & \bullet\ i \in \mathit{need}(q_j) \\ & \bullet\ (q_i, q_j) \text{ is reachable in } \mathit{Sys} \downarrow_{\{i,j\}} \\ & \bullet\ \mathit{Int}(q_i) \cap \mathit{Int}(q_j) = \emptyset\} \end{aligned}$$

$$PS_j^{l+1}(q_i) = \{q_j \mid q_j \in PS_j^l(q_i) \text{ and } \forall k \in \mathit{need}(q_j) : PS_k^l(q_j) \neq \emptyset\}$$

The set of states of j that are ***problematic with respect to*** q_i *is*

$$PS_j(q_i) := \bigcap_{l \in \mathbb{N}} PS_j^l(q_i).$$

Note how this definition differs from Definition 3.4.1. We have already mentioned that it is independent of $\alpha \in Int(q_i)$. This means that all conditions in Definition 3.4.1 that refer to α have to be replaced. On the one hand, we write $i \in need(q_j)$ instead of $comp(\alpha) \cap need(q_j) \neq \emptyset$. Note that this condition corresponds to the one that we derived in the beginning of Remark 3.4.1 (p. 55). On the other hand, the last condition above replaces the two conditions $j(\alpha) \not\subseteq \mathcal{en}(q_j)$ and $\nexists \tilde{\alpha} \in Int(q_i) \cap Int(q_j)$ with $|\tilde{\alpha}| = 2$ that were stated in Definition 3.4.1. It suffices to require $Int(q_i) \cap Int(q_j) = \emptyset$: Any interaction $\tilde{\alpha} \in Int(q_i) \cap Int(q_j)$ would automatically satisfy $|\tilde{\alpha}| = 2$ because all interactions involve two components at most. If we combine the conditions about $\tilde{\alpha}$ and about the incompleteness of q_i and q_j with the fact that all interactions involve two components at most we get an equivalent formulation which simply states that no interaction $\alpha \in Int$, i.e., no *global* interaction, is enabled in (q_i, q_j).

Example 3.2.1 continued: We compute some sets of problematic states for $\overline{Sys_3}$ (cf. Figure 3.4 (p. 43) respectively Figure 3.6 (p. 45)). Consider $\overline{Sys_3} \downarrow_{\{1,4\}}$ first. In this system no pair (q_1, q_4) can be reached where q_1 is problematic with respect to q_4 or vice versa. For any such pair either $4 \notin need(q_1)$ or an interaction in $\overline{Int_3}$ is enabled. This implies that all sets of problematic states of the 0-th instance are empty for 1 and 4. We conclude $PS_1(q_4^0) = PS_1(q_4^1) = \emptyset$ and $PS_4(q_1^1) = PS_4(q_1^3) = \emptyset$.

For $j \in \{2, 3\}$ the state (q_1^2, q_j^0) is reachable in $\overline{Sys_3} \downarrow_{\{1,j\}}$ (by first performing $\{a_1^{\alpha_2}\}$ respectively $\{a_1^{\alpha_1}\}$ depending on whether $j = 2$ or $j = 3$ followed by the transition labeled $\{c_1^{\alpha_5}\}$). We have $1 \in need(q_j^0)$ and $j \in need(q_1^2)$, and no interaction in $\overline{Int_3}$ is enabled in (q_1^2, q_j^0). This means $q_1^2 \in PS_1^0(q_j^0)$ and $q_j^0 \in PS_j^0(q_1^2)$. Analogously, we see $q_1^0 \in PS_1^0(q_j^1)$ and $q_j^1 \in PS_j^0(q_1^0)$. Iterating the computation of the problematic states does not have any effect for the sets above. All sets of problematic states are equal to the 0-th stage sets of problematic states above.

It can be seen that q_j is problematic with respect to q_i if and only if it

is problematic with respect to q_i and all $\alpha \in Int(q_i)$ with $j(\alpha) \neq \emptyset$:

Lemma 3.4.1. *Let Sys be a strongly tree-like interaction system. Let $i \in K$, $q_i \in Q_i$ incomplete, and $j \in need(q_i)$. Let $q_j \in Q_j$.*

We have $q_j \in PS_j(q_i)$ if and only if for all $\alpha \in Int(q_i)$ with $j(\alpha) \neq \emptyset$ we have $q_j \in PS_j(q_i, \alpha)$.

The lemma shows that for strongly tree-like interaction systems the simplification of the definition of a problematic state (in particular the independence of α) does not come at the expense of generality. It is clear that the change in the definition of the sets of problematic states also results in a change in the definition of the sets of problematic actions. Since $\alpha \in Int(q_i)$ does not occur in the definition of $PS_j(q_i)$ any more it can be canceled from the definition of $\mathcal{PA}(q_i, \alpha, j)$, as well.

Definition 3.4.4. *Let Sys be a strongly tree-like interaction system with strongly exclusive communication. For $i \in K$, $q_i \in Q_i$ incomplete, and $j \in need(q_i)$ we define*

$$\mathcal{PA}(q_i, j) := \mathcal{EA}(q_i, need(q_i), PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$$

the set of ***problematic actions of*** *i* ***with respect to*** *q_i* ***and*** *j.*

Example 3.2.1 continued: Using the results gathered above, we specify the sets of problematic actions for $\overline{Sys_3}$ (cf. Figure 3.4 (p. 43) respectively Figure 3.6 (p. 45)). The local states for which we computed the *BWS*-operator and the entry actions (cf. pp. 46 respectively 48) were those local states that are significant for stating the problematic actions now. For $q_4 \in Q_4$ and $q_1 \in Q_1$ we have $\mathcal{PA}(q_4, 1) = \mathcal{PA}(q_1, 4) = \emptyset$ because the corresponding sets of problematic states are empty. For $j \in \{2, 3\}$ and $q_j \in Q_j$ we have $\mathcal{PA}(q_j, 1) = \emptyset$, even though the corresponding sets of problematic states are non empty. This is because in $\overline{Sys_3} \downarrow_{\{1,j\}}$ a relevant pair of states can only be entered by performing interactions in $\overline{Int_3} \downarrow_{\{1,j\}}$ that *only* involve ports of component 1. These ports are exactly the ones listed in the following sets of problematic actions. We get $\mathcal{PA}(q_1^0, 2) = \{b_1^{\alpha_4}\}$, $\mathcal{PA}(q_1^0, 3) = \{b_1^{\alpha_3}\}$, $\mathcal{PA}(q_1^2, 2) = \{a_1^{\alpha_2}\}$, and $\mathcal{PA}(q_1^2, 3) = \{a_1^{\alpha_1}\}$.

The following corollary follows from the definitions and Lemma 3.4.1.

Corollary 3.4.1. *Let Sys be a strongly tree-like interaction system with strongly exclusive communication. Let $i \in K$, $q_i \in Q_i$ incomplete, and $j \in need(q_i)$. Let $a_i \in \mathcal{A}_i$.*

We have $a_i \in \mathcal{PA}(q_i, j)$ if and only if for all $\alpha \in Int(q_i)$ with $j(\alpha) \neq \emptyset$ we have $a_i \in \mathcal{PA}(q_i, \alpha, j)$.

We rephrase Proposition 3.4.1 for strongly tree-like interaction systems.

Corollary 3.4.2. *Let Sys be a strongly tree-like interaction system with strongly exclusive communication*

If the following two conditions hold then Sys is deadlock-free:

1. $\forall i \forall q_i : q_i$ *complete* $\vee\, q_i^0 \notin BWS(q_i, need(q_i)) \vee$
   $\exists j \in need(q_i)$ *with* $q_j^0 \notin BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$

2. $\forall \tilde{\alpha} \in Int$ *with* $|comp(\tilde{\alpha})| = 2$ $\exists i \in comp(\tilde{\alpha})$ *such that for all* $q_i \in Q_i$ *that are incomplete:*

$$i(\tilde{\alpha}) \not\subseteq \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k)$$

The conditions can be checked in time polynomial in the size of Sys.

Checking the technicalities and details of Proposition 3.4.1 and Corollary 3.4.2 is rather tedious and certainly not suitable to be done by hand for large systems. It is essential to provide an adequate tool-support allowing to automatically check whether systems are tree-like and whether the conditions are satisfied. Such tool-support is also important with regard to testing how the procedures scale for large systems and how they compare to other automated verification methods. Building on the tool PrInSESSA [127], a trial version of an algorithm based on the corollary has been implemented. It is in the testing phase. An extension of the tool is planned. In particular, an algorithm checking the conditions of Proposition 3.4.1 shall be provided. Furthermore, a data-structure for interaction systems based on binary decision diagrams [36, 42, 43] is being implemented which allows for a very efficient implementation of the necessary checks.

Example 3.2.1 continued: Deadlock-freedom of $\overline{Sys_3}$ (cf. Figure 3.4 (p. 43) respectively Figure 3.6 (p. 45)) follows from the corollary. From the

sets of problematic actions computed above it is clear that all intersections that have to be considered in the second condition are empty. Therefore, the second condition is satisfied. The first condition is also satisfied. This follows from the sets of problematic states computed above and from the results computed for the *BWS*-operator (cf. p. 46).

We conclude by returning to the original idea discussed in Section 3.1 where we posed the question whether it is possible to derive deadlock-freedom of a tree-like system from deadlock-freedom of all subsystems consisting of two interacting components. We saw that in general this is not possible. However, having developed a formal framework for tree-like interaction systems, we are now in a position to state under what further circumstances it is possible to do so: If a strongly tree-like interaction system has the property that for every local state of every component there is at most one possible communication partner (i.e., $|need\,(q_i)| \leq 1$ for all q_i) then checking deadlock-freedom of *Sys* comes down to checking deadlock-freedom of $Sys\downarrow_{\{i,j\}}$ for every edge $\{i,j\}$ in G^*. This requirement is not as strong a restriction as one might think. There are various results concerning systems where each process can only communicate with exactly one other process in every state (cf. Brookes and Roscoe [38] or Dijkstra and Scholten [58], for example). It is possible to model various interesting systems according to this requirement. In this situation Corollary 3.4.3 below constitutes a further compactification of our results. Interestingly, Brookes and Roscoe [38] present a sufficient condition for deadlock-freedom that also checks deadlock-freedom of all subsystems of size two. This condition is based on the same assumptions as the corollary. In particular, it also requires a tree-like communication architecture and a condition corresponding to the additional requirement $|need(q_i)| \leq 1$ we impose here.

Corollary 3.4.3. *[38] Let Sys be a strongly tree-like interaction system with strongly exclusive communication and* $|need(q_i)| \leq 1$ *for all* $i \in K$ *and for all* $q_i \in Q_i$*.*

If $Sys\downarrow_{\{i,j\}}$ *is deadlock-free for all* $\{i,j\} \in E^*$ *then Sys is deadlock-free.*

We make a final remark about the Corollaries 3.4.2 and 3.4.3. They

state sufficient conditions for deadlock-freedom of strongly tree-like interaction systems. However, they actually comprise stronger statements: In the proofs we will not only show that the conditions entail deadlock-freedom. Instead, we will show that for a strongly tree-like system the conditions of Corollary 3.4.2 are equivalent to those of Proposition 3.4.1. Thus, there is no strongly tree-like interaction system where Corollary 3.4.2 fails to prove deadlock-freedom while the conditions of Proposition 3.4.1 are met. The additional restriction of the architecture causes a simplification of notions and a reduction of complexity without harming the generality of the criterion. The same is true for Corollaries 3.4.3 and 3.4.2 with respect to strongly tree-like interaction systems satisfying the additional requirement above. We did not explicitly include these facts into the formulation of the corollaries in order to get more concise statements.

3.5 Freedom of Local Deadlocks for Tree-Like Component Architectures

We complete the results by a short section about freedom of local deadlocks of tree-like systems. The approach does not have to be altered very much in order to be able to handle freedom of local deadlocks, as well. We state if and how the results have to be adapted to deal with this slightly different problem. We treat strongly tree-like interaction systems first because for this class of systems the conditions in Corollary 3.4.2 already ensure freedom of local deadlocks. In the general case we have to sharpen the conditions of Proposition 3.4.1. Since freedom of local deadlocks is not the main concern of this thesis and because the approach only has to be changed a little we simply state the results without further explanation. In this section D denotes a local deadlock in q.

3.5.1 Strongly Tree-Like Interaction Systems

We state the following result regarding freedom of local deadlocks for strongly tree-like interaction systems:

Corollary 3.5.1. *Let Sys be a strongly tree-like interaction system with*

strongly exclusive communication.

If the following two conditions hold then Sys does not contain any local deadlock:

1. $\forall i\ \forall q_i : q_i\ \textit{complete} \vee q_i^0 \notin BWS(q_i, need\,(q_i)) \vee$
   $\exists j \in need\,(q_i)\ \ \textit{with}\ q_j^0 \notin BWS(PS_j\,(q_i)\,, \mathfrak{need}\,(PS_j\,(q_i)))$

2. $\forall \tilde{\alpha} \in Int$ *with* $|comp(\tilde{\alpha})| = 2\ \exists i \in comp\,(\tilde{\alpha})$ *such that for all* $q_i \in Q_i$ *that are incomplete:*

$$i\,(\tilde{\alpha}) \not\subseteq \bigcap_{k \in need(q_i)} \mathcal{PA}\,(q_i, k)$$

The conditions can be checked in time polynomial in the size of Sys.

The two conditions in the corollary coincide with those in Corollary 3.4.2. As far as strongly tree-like interaction systems are concerned the approach does not have to be changed in order to treat freedom of local deadlocks.

3.5.2 Interaction Systems with Multiway Cooperation

For tree-like interaction systems it is not possible to simply adopt Proposition 3.4.1. The second condition has to be sharpened in order to be able to also handle local deadlocks. The reason for this phenomenon can be motivated as follows: For a strongly tree-like interaction system we know that all interactions in *Int* involve two components, at most. Thus, if D is a local deadlock in q then for $i \in D$ all $\alpha \in Int(q_i)$ involve *exactly* two components. Otherwise q_i would be complete, and no complete local state can be part of a local deadlock. Using this observation, it can be seen that for a local deadlock D in q we have $comp(\alpha) \subseteq D$ for all $i \in D$ and all $\alpha \in Int(q_i)$. Every $\alpha \in Int(q_i)$ is blocked by a component in $comp(\alpha) \cap D$. There is exactly one choice for this component namely the component that participates in α other than i. Therefore, this component must be contained in D. This fact will be needed in the proof of Corollary 3.5.1. For tree-like interaction systems we cannot argue the same way any more. Given a local deadlock D in q we only know that for all $i \in D$ and all $\alpha \in Int\,(q_i)$ there has to be *at least one* other component in $comp\,(\alpha)$ that is also contained in D (and that blocks

α). We do not know whether the other components that participate in α are contained in D or not. However, it is still possible to state a criterion for freedom of local deadlocks in tree-like interaction systems which is similar to Proposition 3.4.1. The first condition of the proposition can be adopted without change. We replace the second condition by the requirement that no $\tilde{\alpha} \in Int$ contains a port a_i for which there is an incomplete local state q_i such that a_i is problematic with respect to all $\alpha \in Int(q_i)$. This simply results in the requirement that all intersections in the second condition of the proposition must be empty.

Proposition 3.5.1. *Let Sys be a strongly tree-like interaction system with strongly exclusive communication.*

If the following two conditions hold then Sys is free of local deadlocks:

1. $\forall i \forall q_i : q_i$ *complete* $\vee\, q_i^0 \notin BWS(q_i, need(q_i)) \vee \exists \alpha \in Int(q_i)$ *such that* $\forall j \in comp(\alpha) \setminus \{i\} : q_j^0 \notin BWS(PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$

2. $\forall i \in K$ *and* $\forall q_i \in Q_i$ *that are incomplete:*

$$\bigcap_{\alpha \in Int(q_i)} \bigcup_{j \in comp(\alpha) \setminus \{i\}} \mathcal{PA}(q_i, \alpha, j) = \emptyset$$

The conditions can be checked in time polynomial in the size of Sys.

3.6 Examples

This section is devoted to three larger examples. They model real life situations taken from different scenarios. They show that our results can be applied to arbitrarily large systems: The first example is parametrized to allow for an arbitrary number of components. Therefore it yields a class of examples. The third example models a system that can be multiplied and combined to construct arbitrarily large systems.

3.6.1 A Banking System

We model a banking system which is similar to the one described by Baumeister et al. [30]. It consists of a clearing company, a number of banks,

and a number of ATMs for each bank. We only model the withdrawal of money at an ATM, and we do not specify other functionalities the different components may have. A withdrawal should obey the following protocol: An ATM can request money from its bank. With the help of the clearing company the bank then checks the correctness of the PIN that was delivered. Depending on the reply of the clearing company the ATM may disburse money or cancel the transaction. We abstract from the actual values of the PIN and the amounts of money on the accounts since we are only interested in the communication between the components.

We define an interaction system behaving according to this specification. There are $m \geq 1$ banks. Each bank is represented by a component b_i where $1 \leq i \leq m$. For each b_i there are $n_i \geq 1$ ATM components atm^i_j where $1 \leq j \leq n_i$. The clearing company is represented by the component cc. We get $K_{bank} := \{cc, b_i, atm^i_j | 1 \leq i \leq m, 1 \leq j \leq n_i\}$. The transition systems modeling the local behavior of the components are depicted in Figure 3.13. For each i the port set $\mathcal{A}_i$ is understood to coincide with the set of labels of T_i. We will write q^i_j instead of $q_{atm^i_j}$ to denote a local state of atm^i_j.

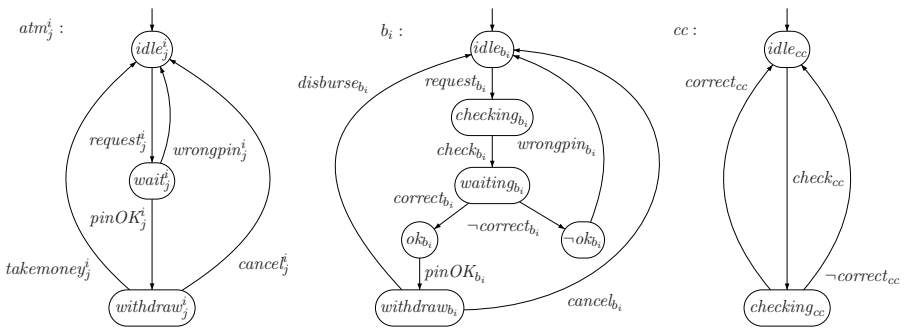

Figure 3.13: The local behavior of the components in K_{bank}

The following interactions describe the cooperations that are allowed between the components. Using $\{request_{b_i}, request^i_j\}$, an ATM may send a request for money to its bank. The bank checks the PIN with the clearing company according to the interaction $\{check_{b_i}, check_{cc}\}$. The clearing company then informs the bank whether the PIN was correct or incorrect using $\{correct_{b_i}, correct_{cc}\}$ respectively $\{\neg correct_{b_i}, \neg correct_{cc}\}$. Be-

cause we abstract from the value of PIN the choice happens nondeterministically. The interactions $\{pinOK_{b_i}, pinOK_j^i\}$ and $\{wrongpin_{b_i}, wrongpin_j^i\}$ are used by the bank to inform the ATM about the result of the check. If necessary the bank disburses money using $\{disburse_{b_i}, takemoney_j^i\}$ or the transaction is canceled using the interaction $\{cancel_{b_i}, cancel_j^i\}$. We get the following subsets of interactions. For each bank b_i we define $Int_i := \{\{check_{b_i}, check_{cc}\}, \{\neg correct_{b_i}, \neg correct_{cc}\}, \{correct_{b_i}, correct_{cc}\}\}$. For each b_i and each atm_j^i the set Int_j^i contains the interactions $\{pinOK_{b_i}, pinOK_j^i\}$, $\{wrongpin_{b_i}, wrongpin_j^i\}$, $\{request_{b_i}, request_j^i\}$, $\{disburse_{b_i}, takemoney_j^i\}$, and $\{cancel_{b_i}, cancel_j^i\}$. Define

$$Int_{bank} := \bigcup_{i=1}^{m} Int_i \ \cup \bigcup_{i=1,j=1}^{m,n_i} Int_j^i$$

and denote the induced interaction system by Sys_{bank}.

In the following we exemplarily compute the notions of this chapter and retrace the results for Sys_{bank}. In general (if there are at least two banks or if there is a bank with at least two ATMs), Sys_{bank} does not have strongly exclusive communication. Each port of cc occurs in different interactions (in one for each bank). Likewise the ports of a bank b_i that are meant for communication with the ATMs also occur in various interactions. We carry out the construction of $\overline{Sys_{bank}}$ defined in Lemma 3.2.1 in detail. Being somewhat imprecise to increase readability, we do not superscript the new ports of the components in $\overline{K_{bank}}$ by the names of the interactions they occur in. Instead we simply superscript each port by the name of the component it is used to communicate with. Further, we do not replace those ports that already occur in one interaction only, i.e., we keep the names of the ports of the ATMs and the names of the ports that a bank uses for communication with the clearing company. The port $check_{cc}$ occurs in all interactions of the form $\{check_{b_i}, check_{cc}\}$. Thus, $\overline{\mathcal{A}_{cc}}$ contains the subset $\{check_{cc}^1, \ldots, check_{cc}^m\}$ of ports replacing $check_{cc}$. The port $check_{cc}^i$ is only used for communication with $check_{b_i}$. In T_{cc} the transition labeled $check_{cc}$ is replaced by the set of parallel edges depicted in Figure 3.14 a). The other ports of cc are treated analogously. Next, consider $request_{b_i} \in \mathcal{A}_{b_i}$. It occurs in $\{request_{b_i}, request_j^i\}$ for $1 \leq j \leq n_i$. As above, a subset $\{request_{b_i}^1, \ldots, request_{b_i}^{n_i}\}$ of new ports

replaces $request_{b_i}$ in $\overline{\mathcal{A}_{b_i}}$. The transitions replacing the transition labeled $request_{b_i}$ are depicted in Figure 3.14 b). G^* for $\overline{Sys_{bank}}$ is depicted in Figure 3.15. It is a tree.

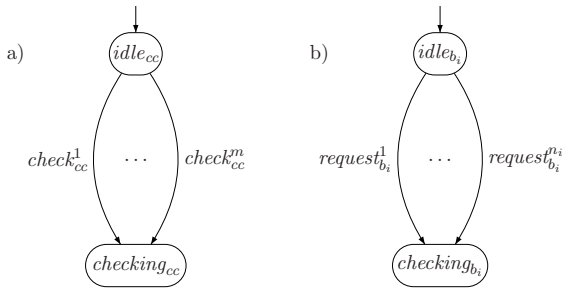

Figure 3.14: The construction of $\overline{Sys_{bank}}$

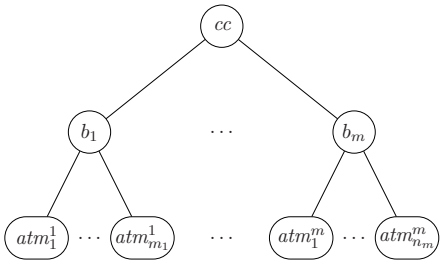

Figure 3.15: G^* of $\overline{Sys_{bank}}$

Corollary 3.4.2 proves that $\overline{Sys_{bank}}$ and consequently Sys_{bank} are deadlock-free. We carry out one of the computations necessary for the application of the corollary. Consider bank b_i, and assume that it has at least two ATMs[2]. Consider ok_{b_i}. We have $need(ok_{b_i}) = \{atm^i_1, \ldots, atm^i_{n_i}\}$. We compute the problematic states of atm^i_j with respect to ok_{b_i}. It is clear that $wait^i_j$ is not problematic with respect to ok_{b_i} because it enables the port $pinOK^i_j$. The other two local states $idle^i_j$ and $withdraw^i_j$ of atm^i_j do not enable this port. In addition $b_i \in need(q^i_j)$ for both these states. We only further have to check whether each of these two states is reachable in com-

[2]If there is only one ATM atm^i_1 the situation is much more simple because $PS_{atm^i_1}(q_{b_i})$ and $PS_{b_i}(q^i_1)$ are empty for all local states q_{b_i} of b_i and q^i_1 of atm^i_1.

bination with ok_{b_i}. This is the case because b_i can proceed independently of atm^i_j by performing ports that are for communication with ATMs $atm^i_{j'}$ other than atm^i_j. We get $PS^0_{atm^i_j}(ok_{b_i}) = \{idle^i_j, withdraw^i_j\}$. In order to see that we also have $PS_{atm^i_j}(ok_{b_i}) = \{idle^i_j, withdraw^i_j\}$ we have to compute the 0-th stage problematic states for the local states q^i_j of atm^i_j, as well. This can be done in the same way as described above. It can be seen that for all q^i_j we have $PS^0_{b_i}(q^i_j) \neq \emptyset$. In fact, for reasons of symmetry we have $q_{b_i} \in PS^0_{b_i}(q^i_j)$ if and only if $q^i_j \in PS^0_{atm^i_j}(q_{b_i})$. We get $PS_{atm^i_j}(ok_{b_i}) = \{idle^i_j, withdraw^i_j\}$. Next, we compute the problematic actions. We first have to evaluate the BWS-operator for the states involved. Note that atm^i_j constitutes a leaf of G^*. Therefore and because there are no singleton interactions it *only* communicates with b_i and according to the statement made in beginning of Example 3.2.1 (p. 46) we have $BWS(q^i_j, need(q^i_j)) = \{q^i_j\}$ for all local states of atm^i_j. All labels on the path

$$checking_{b_i} \overset{check_{b_i}}{\rightarrow} waiting_{b_i} \overset{correct_{b_i}}{\rightarrow} ok_{b_i}$$

are ports that are used for communication with cc, i.e., these ports are not in $\bigcup_{k \in need(ok_{b_i})} \boldsymbol{comm}_{b_i}(k)$. Therefore, we conclude $BWS(ok_{b_i}, need(ok_{b_i})) = \{ok_{b_i}, waiting_{b_i}, checking_{b_i}\}$. Since we are interested in $\mathcal{PA}(ok_{b_i}, atm^i_j)$ we must check whether one of these three states can be reached in combination with $idle^i_j$ or $withdraw^i_j$ in $\overline{Sys_{bank}} \downarrow_{\{b_i, atm^i_j\}}$ by performing a port that is used for communication with a component in $need(ok_{b_i}) = \{atm^i_1, \ldots, atm^i_{n_i}\}$. The only combinations of these states that are relevant are $(checking_{b_i}, idle^i_j)$ and $(checking_{b_i}, waiting^i_j)$. Both pairs of states can be reached by performing $\{request^{j'}_{b_i}\}$ for $j' \in \{1, \ldots, n_i\} \backslash \{j\}$. If an interaction in $\overline{Int_{bank}} \downarrow_{\{b_i, atm^i_j\}}$ involving $request^j_{b_i}$ (i.e., $\{request^j_{b_i}, request^i_j\}$) is performed then $request^i_j$ *must* also be performed. This results in atm^i_j changing to $wait^i_j$. Therefore $request^i_j$ is *not* problematic with respect to ok_{b_i} and atm^i_j. Letting j vary in $need(ok_{b_i}) = \{atm^i_1, \ldots, atm^i_{n_i}\}$ we get analogous results for all other ATMs. This shows:

$$\bigcap_{j \in need(ok_{b_i})} \mathcal{PA}(ok_{b_i}, j) = \emptyset$$

The other states of b_i can be treated similarly. Again the intersections of the sets of problematic states are empty. Considering cc, an analogous

argument shows that for each q_{cc} and each b_i those ports of cc which label a transition entering q_{cc} and which are used for communication with b_i are *not* problematic with respect to q_{cc} and b_i. Thus, the intersection of the sets of problematic actions for q_{cc} is again empty. Finally, for any local state q_j^i of an atm_j^i it can be seen that $\mathcal{PA}(q_j^i, b_i) = \emptyset$. Even though $PS_{b_i}(q_j^i) \neq \emptyset$ combinations of such states can *only* be reached in $\overline{Sys_{bank}} \downarrow_{\{atm_j^i, b_i\}}$ by performing interactions in $\overline{Int_{bank}} \downarrow_{\{atm_j^i, b_i\}}$ not involving atm_j^i. Therefore, there are no problematic actions of atm_j^i with respect to q_j^i and b_i. We conclude that all intersections of sets of problematic actions that have to be considered in the second condition of Corollary 3.4.2 are empty. Therefore, the condition is satisfied. The first condition only has to be checked for local states q_i with $q_i^0 \in BWS(q_i, need(q_i))$. These are $idle_j^i$ for atm_j^i and $idle_{cc}$, as well as $idle_{b_i}$ and $checking_{b_i}$ for the banks. Considering the sets of problematic states for these local states, it is easy to see that the first condition is satisfied for these states. We conclude that $\overline{Sys_{bank}}$ is deadlock-free. In fact, Corollary 3.5.1 shows that $\overline{Sys_{bank}}$ is also free of local deadlocks. According to Lemma 3.2.1, Sys_{bank} has these properties, as well. The set of reachable global states of Sys_{bank} is exponentially large in the number of components if there are several banks and several ATMs for each bank. Therefore, it would not be feasible to check deadlock-freedom for Sys_{bank} directly even if the system only consists of comparatively few components.

This example indicates an interesting feature of the criterion: The approach works particularly well if applied to systems where groups of components that compete for cooperation with $i \in K$ (in the sense that i has states where it can choose between these components) have similar or even the same behavior. As seen above the intersections in the second condition will be empty in this case because whenever we observe i in parallel with j the ports in $\mathit{comm}_i(j)$ do not lead to a problematic state. Such a port has to synchronize with j in $Sys \downarrow_{\{i,j\}}$, and this synchronization should not lead to a pair of states where the ports offered by i and j are not compatible. If it did this could be interpreted as an indicator that the system has not been specified correctly. Sys_{bank} has the above property: All communication partners of cc (the banks) are the same up to nomenclature. All communication partners of b_i (the ATMS) but one — namely cc which never competes with

the ATMs for cooperation — also behave in the same way. However, note that such symmetries are not necessary for the applicability of the criterion.

3.6.2 The River Delta

We consider a system of barrages and locks in a river delta. The system has to control the closing of the barrages to protect the coastline in case of imminent storms. Further, the coordination between the various elements of a lock has to be organized. We do not discuss the application of Proposition 3.4.1 in detail. We primarily use this example to show that there are systems with a natural tree-like architecture where this is not obvious at first glance.

Figure 3.16 shows the stylized coastline of an imaginary country. The

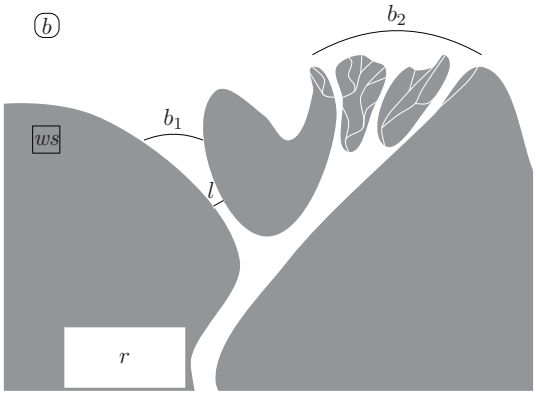

Figure 3.16: The estuary

area comprises the estuary of a river which splits up just before flowing into the sea. One river arm flows through an industrial area and has been developed for shipping traffic while the other further divides into a delta. Both arms are equipped with a barrage. The industrial arm is additionally equipped with a river lock in order to coordinate the arrival and departure of ships. There is a weather station on the coast and a scientific buoy off shore that is supposed to deliver early warning in case of impending storms. Finally, there is a reservoir upstream that can be opened to relax floods in the industrial arm of the river which may be caused by shutting the gates of the barrages. Each barrage has two different functions. First, the

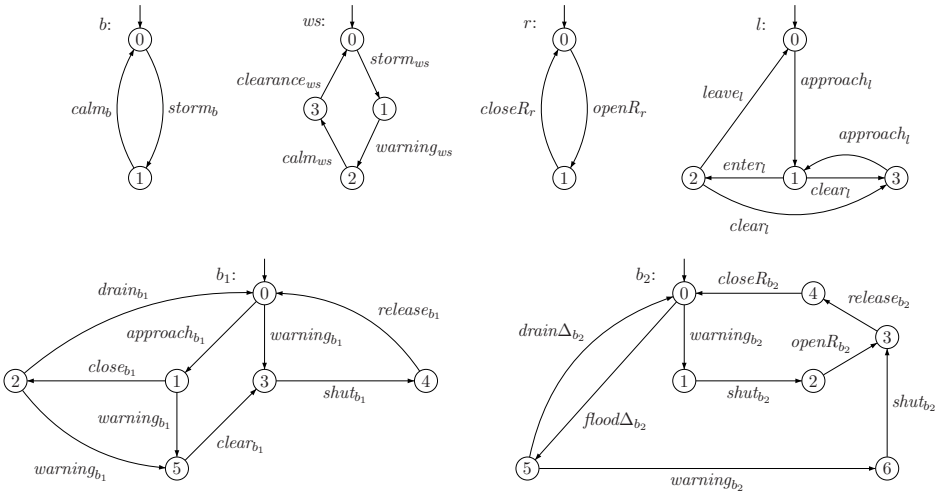

Figure 3.17: The local behavior of the components in K_Δ

barrages are supposed to cooperate in order to protect the coastal area from being flooded in case of surges. Such surges are forecast by the buoy which subsequently informs the weather station. The weather station then has to coordinate the simultaneous closing of the gates. Further the barrages may be closed independently of each other. The first one can work together with the lock in order to allow the passage of ships whereas the other one can be shut in order to permit a (controlled) flooding of the delta (which might be necessary for agricultural reasons, for example). The reservoir has to be opened whenever the second barrage is shut. Closing of the barrages in case of storms always has to be possible. It has to be made sure that the barrages are permanently ready to receive a warning from the weather station.

We model the river delta as an interaction system. Each of the different devices is represented by a component. There are the barrages b_1 and b_2, the river lock l and the reservoir r as well as the weather station ws and the buoy b. We get $K_\Delta = \{b_1, b_2, l, r, ws, b\}$. The transition systems modeling the local behavior of the components are depicted in Figure 3.17. In the figure a state of component $i \in K_\Delta$ labeled x is understood to denote $q_i^x \in Q_i$. For each $i \in K_\Delta$ the port set $\mathcal{A}_i$ coincides with the set of labels of T_i.

We introduce the following interactions: The buoy notifies the weather station of impending or abating storms by means of the interactions $\alpha_1 := \{storm_b, storm_{ws}\}$ respectively $\alpha_2 := \{calm_b, calm_{ws}\}$. The weather station

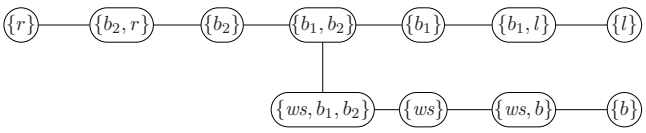

Figure 3.18: The interaction graph of Sys_Δ

then has to inform the barrages of the incoming or retreating storm. For this purpose we introduce $\alpha_3 := \{warning_{ws}, warning_{b_1}, warning_{b_2}\}$ respectively $\alpha_4 := \{clearance_{ws}, release_{b_1}, release_{b_2}\}$. In the first event the two barrages have to be shut simultaneously using $\alpha_5 := \{shut_{b_1}, shut_{b_2}\}$. Furthermore, the second barrage has to prompt the reservoir to be opened or closed. We add interactions $\alpha_6 := \{openR_{b_2}, openR_r\}$ and $\alpha_7 := \{closeR_{b_2}, closeR_r\}$. On the other hand, the barrages have to be able to proceed independently in order to fulfill their respective functionalities. The first one must make sure that the passage of ships is coordinated with the lock. There are interactions indicating the approach $\alpha_8 := \{approach_{b_1}, approach_l\}$, entering $\alpha_9 := \{close_{b_1}, enter_l\}$, and departure $\alpha_{10} := \{drain_{b_1}, leave_l\}$ of a ship. Note that the barrage is ready to obtain a storm-warning in the process of allowing the passage of a ship. In case of such a warning the lock has to be cleared using the interaction $\alpha_{11} := \{clear_{b_1}, clear_l\}$. The second lock has to fulfill the purpose of temporarily flooding the delta. In order to open and close the reservoir in this case there are interactions $\alpha_{12} := \{flood\Delta_{b_2}, openR_r\}$ and $\alpha_{13} := \{drain\Delta_{b_2}, closeR_r\}$. We define $Int_\Delta := \{\alpha_l | 1 \leq l \leq 13\}$, and we denote the induced interaction system by Sys_Δ.

Deadlock-freedom of Sys_Δ is crucial. Figure 3.18 shows that Sys_Δ is tree-like, and Proposition 3.4.1 shows that Sys_Δ is deadlock-free. Of course, further investigations are necessary in order to prove other important properties for this system. For example, it has to be guaranteed that closing of the barrages in case of a storm always has priority over all other functionalities. Priorities can be included into the formalism of interaction systems [72]. Under certain conditions it can be shown that a system with priorities is deadlock-free if the underlying system without priorities is deadlock-free.

3.6.3 The Railway Track

Having seen two examples that can be modeled as tree-like systems, we conclude by discussing a system which does not have an underlying tree-like architecture. However, we will show how a simple construction can be used to obtain an equivalent system that is tree-like. As in the case of $\overline{Sys}$ introduced in Lemma 3.2.1 it is *not* the point to replace the original system. Instead, we prove that the adapted system is deadlock-free and carry the result over to the original system.

We model a simple substructure of a network of train stations and tracks. It consists of two stations, st_1 and st_2, and the track between them. This track is a single track which has to be traveled in both directions. In order to allow trains to avoid collision with oncoming trains the track is equipped with a bifurcation b in the middle. The segment is schematically depicted in Figure 3.19. Assume that all trains that arrive at the bifurcation coming

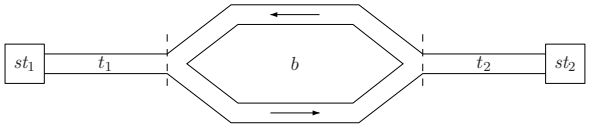

Figure 3.19: Basic segment of the train network

from t_1 take the lower branch of the bifurcation whereas all trains coming from t_2 take the upper branch. Once this substructure is defined, it can be multiplied and stuck together at the stations in order to construct complex railroad systems. This nicely reflects the requirement put upon a component in Section 1.1: It should be reusable and independent of the context it will be used in. Here the context independence is given to a certain degree because the substructure can be used to construct many different networks of tracks.

We model the segment by an interaction system. We have $K_{track} := \{st_1, st_2, t_1, t_2, b\}$. We set $\mathcal{A}_{st_i} := \{from_i^t, to_i^t\}$. These two ports describe the fact that the respective station can receive a train from respectively send a train to the corresponding track. Similarly, each track t_i has ports which allow for passage of a train to and from the corresponding station respectively to and from the bifurcation. We get $\mathcal{A}_{t_i} := \{to_i^{st}, from_i^{st}, to_i^b, from_i^b\}$. The

bifurcation may send a train to or receive a train from either of the tracks. For t_i these actions are described by to^{t_i} respectively $from^{t_i}$. The bifurcation further has the task to control the entering into the tracks of trains coming from the corresponding stations. For each t_i the bifurcation therefore provides a port $enter^{t_i}$. We get $\mathcal{A}'_b := \{to^{t_i}, from^{t_i}, enter^{t_i} | i \in \{1,2\}\}$. The local behavior of the components is given in Figure 3.20. We denote the transition system of b by T'_b.

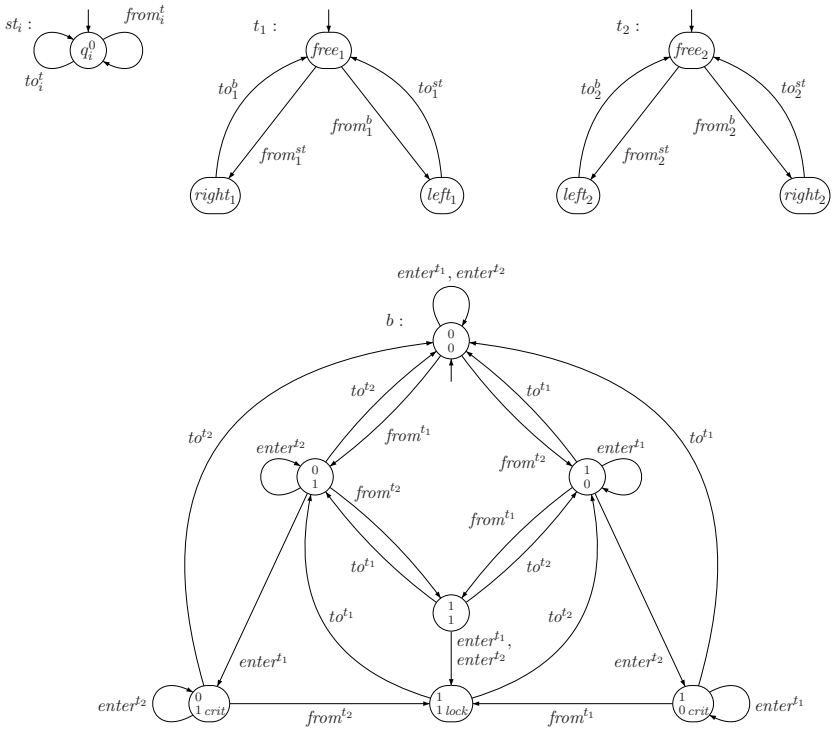

Figure 3.20: The local behavior of the components in K_{track}

The names of the states were chosen to give an intuitive idea of their meaning. For example, the state $right_1$ of t_1 indicates that the track is occupied by a train traveling from st_1 to b. The states of b have been labeled by elements $(x, y) \in \{0,1\}^2$ depicted as $\begin{smallmatrix} x \\ y \end{smallmatrix}$. Here x and y indicate whether the top respectively lower branch of b are occupied. Since both branches may only be traveled in one direction it suffices to keep track of whether they are occupied or not. The transition systems convey the behavior that would be expected of the components. There are only a few subtleties to

be noted. We do not specify the behavior of the stations in detail. Each station may repeatedly and nondeterministically send trains to and receive trains from its corresponding track. This behavior can be concretized if necessary. For the moment we settle for this general description of the stations. The diamond formed by the upper four states in T_b' describes the behavior one would want the bifurcation to have. There are three additional states, though. They restrict the admittance to the tracks of trains coming from the stations. Depending on the state b does not allow trains coming from st_1 or from st_2 or any train at all to enter the track.

We define the following interactions. There is one interaction which allows a train coming from station st_i to enter track t_i. This cooperation is monitored by b. It is given by $\alpha_i^1 := \{to_i^t, from_i^{st}, enter^{t_i}\}$. The interaction $\alpha_i'^2 := \{from_i^t, to_i^{st}\}$ allows a train coming from track t_i to enter the station st_i. Similarly, there are interactions allowing for a train to enter the bifurcation coming from track t_i or to enter track t_i coming from the bifurcation. These are $\alpha_i^3 := \{to_i^b, from^{t_i}\}$ respectively $\alpha_i^4 := \{from_i^b, to^{t_i}\}$. We get $Int'_{track} := \{\alpha_i^1, \alpha_i'^2, \alpha_i^3, \alpha_i^4 | i = 1, 2\}$. We denote this system by Sys'_{track}. The system has strongly exclusive communication. However, Figure 3.21 shows that it is not tree-like. Proposition 3.4.1 cannot be applied.

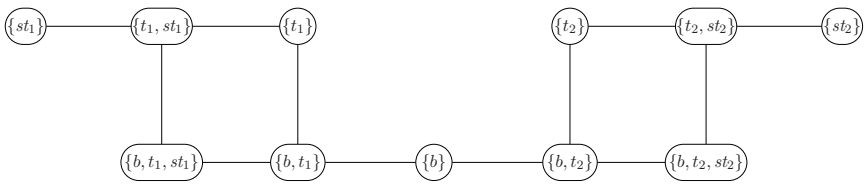

Figure 3.21: The interaction graph of Sys'_{track}

The situation can be remedied by a simple adjustment. The cycle involving t_1 arises because there are interactions involving t_1 and st_1 on the one hand respectively t_1 and b on the other as well as an interaction involving all three of these components. The latter interaction is necessary to allow for the bifurcation to control the admittance of trains coming from st_1 into t_1. Even though the bifurcation does not have to monitor the arrival at st_1 of a train coming from t_1 we may force it to do so *without* changing the global

behavior of Sys'_{track} by carrying out the following construction. We extend $\mathcal{A}'_b$ by $leave^{t_1}$. For each local state q_b in T'_b we add a loop $q_b \overset{leave^{t_1}}{\rightarrow} q_b$. Finally, we replace $\alpha'^2_1 = \{from^t_1, to^{st}_1\}$ by $\alpha^2_1 := \{from^t_1, to^{st}_1, leave^{t_1}\}$. A state q of the resulting interaction system is reachable if and only if it was reachable in the original system. Further, it is a deadlock if and only if it was a deadlock there. We treat t_2 analogously. We denote the new port set and local behavior of b by $\mathcal{A}_b$ respectively T_b and the new set of interactions by Int_{track}. The resulting interaction system is denoted by Sys_{track}. Its interaction graph depicted in Figure 3.22 is a tree. Thus, we may apply Proposition 3.4.1 to obtain results about Sys_{track} and consequently about Sys'_{track}. There are limitations to the applicability of this technique, though. First of all, the interaction graph may contain so many cycles that the construction cannot be performed efficiently. Further, not all cycles can be removed this way. A cycle can only be eliminated if it has a similar structure as the ones encountered in the interaction graph of Sys'_{track}. In particular, we cannot get rid of cycles involving more than one node representing a single component.

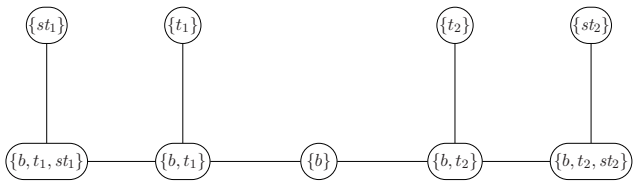

Figure 3.22: The interaction graph of Sys_{track}

We want to show that Sys_{track} is deadlock-free. This is by no means obvious. Consider a situation where the upper branch of b is occupied by a train traveling towards st_1 whereas the lower branch is occupied by a train traveling towards st_2. As long as at least one of the tracks is free there is no problem. However, a situation might arise where each station sends another train to its track before one of the trains in the bifurcation continues its journey. This causes a deadlock because neither of the trains in the bifurcation can leave the bifurcation before one of the trains on the tracks leaves the track and vice versa. It is not clear from the definition of

Sys_{track} that no global state[3] will ever be reached whose entries for b and the t_i coincide with $\begin{smallmatrix}1\\1\end{smallmatrix}$ or $\begin{smallmatrix}1\\1\,lock\end{smallmatrix}$ respectively $right_1$ and $left_2$, and it definitely does not follow from the informal specification of the system given in the beginning of this subsection.

Deadlock-freedom of Sys_{track} in fact follows from Proposition 3.4.1. We omit the details of the necessary computations. We just state a few facts resulting from the computation. Contrary to Sys_{bank} where we had $PS_j^0(q_i) = PS_j(q_i)$ for all relevant combinations of q_i and j, here we profit from iterating the construction of the problematic states. We are able to eliminate a significant number of local states from the sets of problematic states. Similarly to Sys_{bank}, most of the intersections that have to be computed for the second condition of the proposition are empty, but it can be seen that for both stations we have

$$\bigcap_{\alpha\in Int(q_i^0)} \bigcup_{j\in comp(\alpha)\setminus\{st_i\}} \mathcal{PA}(q_i^0,\alpha,j) = \{to_i^t\}$$

whereas for both tracks we have

$$\bigcap_{\alpha\in Int(q_i)} \bigcup_{j\in comp(\alpha)\setminus\{t_i\}} \mathcal{PA}(q_i,\alpha,j) = \{from_i^{st}\}$$

where q_i denotes $right_1$ if we consider t_1 and $left_2$ if we consider t_2. However, this does not cause a problem. The only interaction containing some of these ports is $\alpha_i^1 = \{to_i^t, from_i^{st}, enter^{t_i}\}$. It is clear that α_i^1 satisfies the second condition of Proposition 3.4.1, though, because the port $enter^{t_i}$ makes sure that α_i^1 cannot be used to reach a state where all components in $comp(\alpha_i^1)$ are blocked. We conclude that second condition of the proposition is satisfied. The first condition is satisfied by Sys_{track}, as well. Thus, Sys_{track} is deadlock-free. This result carries over to Sys'_{track} as originally considered.

We point out a few details. We argued informally above that it is not obvious that Sys_{track} is deadlock-free. In particular, a state where the bifurcation is occupied and on each track a train is waiting to enter the bifurcation would cause a deadlock. If in a first naive specification the local transition system of b had only consisted of the diamond of the upper four states without keeping track of trains entering t_1 and t_2 from the stations this deadlock

[3]Note that $D = \{b, t_1, t_2\}$ also constitutes a local deadlock in any such state.

would have indeed been reachable. Retracing the computations for such a system it would become clear that the second condition of Proposition 3.4.1 would not be satisfied for those interactions allowing the global system to move to a state whose entry for b coincides with $\begin{smallmatrix}1\\1\end{smallmatrix}$. In such a state b would not be able to influence trains entering the tracks, and the deadlock might occur. Thus, the need for b to be able to control the entering into the tracks of trains coming from the corresponding stations is directly reflected in the conditions of the proposition. By introducing the local states $\begin{smallmatrix}0\\1\,crit\end{smallmatrix}$, $\begin{smallmatrix}1\\1\,lock\end{smallmatrix}$, and $\begin{smallmatrix}1\\0\,crit\end{smallmatrix}$ we made sure that b is indeed able to exercise this control.

As stated in Definition 3.4.1 for each q_i, $\alpha \in need(q_i)$, and $j \in comp(\alpha)$ we obtain a descending sequence $PS_j^l(q_i, \alpha)$ of sets of problematic states. In particular, we have $PS_j(q_i, \alpha) \subseteq PS_j^l(q_i, \alpha)$ for all $l \in \mathbb{N}$. This means that the proposition could have been formulated with $PS_j^l(q_i, \alpha)$ instead of $PS_j(q_i, \alpha)$ for any index l. This would result in a less general criterion but it would not harm the correctness of the proposition. If the conditions in Proposition 3.4.1 already hold for some l (adapting the definition of the problematic actions by computing the entry actions with respect to the l-th stage sets of problematic states) they certainly also hold for the intersection of all sets of problematic states. If the conditions of Proposition 3.4.1 had been computed for Sys_{track} with $PS_j^0(q_i, \alpha)$ instead of $PS_j(q_i, \alpha)$ they would already have been satisfied. Even though we were able to remove several states from the sets of problematic states by iterating their construction this would not have been necessary. Without going into the details, we use this remark to motivate that strategies could be conceived that help to avoid carrying out all computations to the last detail. In the worst case all iterations have to be followed through but for a concrete implementation of the criterion this observation may help to further improve the runtime of an algorithm based on the proposition.

So far, we have only investigated a substructure as depicted in Figure 3.19. We briefly explain how such substructures have to be assembled in order to create larger railway networks that can *still* be investigated using Proposition 3.4.1. The assumption that all results of this chapter are based on requires the system to be tree-like. Therefore, the building blocks given by Sys'_{track} have to be stuck together in such a way that no cycle arises. This

is not as strong a restriction as one might think at first glance. There are examples of public transport systems that are designed this way. This holds in particular for all such systems that are organized in a centralized manner where all lines radiate from a central station. Because Sys'_{track} is deadlock-free there is no further need to consider the substructures. We only have to make sure that no deadlock is caused by composing the various sections of the railroad network. Taking the current specification of the stations, this is not possible. Depending on the scenario, it might be necessary to further specify the behavior of the stations, though. Changing the specification of the stations (for example, the number of trains a station can accommodate could be restricted), might cause problems and has to be handled with care. The key point is that such a change would not destroy the tree-like structure of Sys'_{track}, and Proposition 3.4.1 would still be applicable.

We conclude by a short remark explaining why it is not possible to model Sys_{track} as a strongly tree-like interaction system. Any interaction in a strongly tree-like system involves two components at most. Thus, if we had tried to model the substructure of tracks by a strongly tree-like interaction system we would have had to realize all cooperations by interactions involving two components. The argument concerning possible deadlocks caused by trains freely entering the tracks shows that there would definitely have to be a cooperation between st_i and b making sure that before sending a train to the track the station checks whether it is allowed to do so. However, there also have to be cooperations between t_i and st_i respectively t_i and b. These interactions would cause a cycle in G^* showing that it is impossible to model Sys_{track} by a strongly tree-like interaction system. Already for this relatively simple system the notion of strongly tree-like interaction systems does not suffice any more.

3.7 Conclusion and Related Work

3.7.1 Conclusion and Discussion

We presented a compositional analysis of deadlock-freedom for the subclass of tree-like interaction systems. We first dealt with reachability. We

defined a backward-analysis for local states and a notion of entry actions which can be computed for subsystems consisting of two components. We presented a result allowing to exclude the possibility that certain global states are reachable only by comparing the relevant sets of entry actions. We introduced two interaction graphs — G and G^* — representing the allowed interactions between the components. These graphs were used to restrict the communication among the components by requiring them to form trees. This architectural constraint was used to introduce a notion of problematic states with respect to a local state q_i describing those local states of components in $need(q_i)$ that do not enable compatible ports. Bringing together the different threads, we used the results about reachability to make sure that no combinations of problematic states are globally reachable that might result in a global deadlock. Accommodating for the two different versions of interaction graphs, we first presented the results for the general case based on G. Then we simplified the results for systems where all interactions involve two components at most. Finally, we modified the results to also account for freedom of local deadlocks.

The results are based on information that we derive from the analysis of subsystems consisting of two interacting components. One might wonder whether it is really necessary to consider a tree-like system in order to apply the techniques presented above or whether they may also help in cases where the interaction graph contains cycles. The following example shows that this is not possible. Consider the system consisting of the components i, j, and k which results from the transition systems depicted in Figure 3.23 and the interaction set $Int := \{\{a_i, a_j\}, \{b_j, b_k\}, \{c_i, c_k\}\}$. The system is not tree-like.

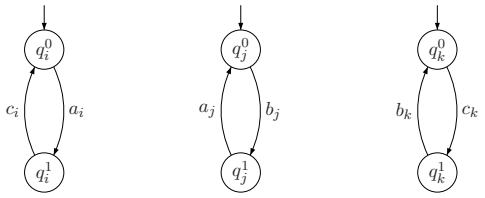

Figure 3.23: The local behavior of the components i, j, and k

Furthermore, its initial state is a deadlock. However, if we were to compute the sets of problematic states all these sets would be empty. For q_i^0 and j,

for example, $q_j^0 \notin PS_j(q_i^0)$ because $i \notin need(q_j^0)$ and $q_j \notin PS_j(q_i^0)$ because (q_i^0, q_j) enables $\{a_i, a_j\}$. A system where all sets of problematic states are empty satisfies the conditions in our criteria. Thus, the ideas of this chapter cannot be used if the communication architecture is cyclic. In particular, it does not suffice to only consider subsystems of size two for such a system.

At this point we would like to advise the reader that it is also possible to take a slightly different point of view towards the results presented in this chapter. So far, we have always taken up the position that the interaction system for which we want to establish deadlock-freedom is given and will not be changed. For such a system we may check whether G is a tree and whether the conditions are satisfied. However, it is also possible to use the results to create systems from scratch that are deadlock-free *by construction*. In Section 2.3 we mentioned that there is a composition operator for interaction systems [92]. Because it is commutative and associative it allows to construct complex systems by composing components respectively systems. The most simple way to do so is by adding one component to an existing system at a time. This approach has the advantage that it can be realized such that the changes to the existing system are always local, i.e., they only affect those components for which interactions involving the newly added component are introduced. Based on this observation we can derive instructions for the construction of deadlock-free systems with a tree-like architecture by making sure that every time we add a new component a) the system's architecture remains tree-like and b) adding the component does not lead to a violation of the conditions formulated for deadlock-freedom in this chapter. Exhibiting such design-guidelines which put requirements on the new component and on the way it is added, it is then possible to derive results for a global system by only performing very few checks every time a component is added. The ultimate goal of this approach is a construction kit for interaction systems which are deadlock-free (respectively correct with regard to some other property) by construction.

We want to stress that the techniques we developed (i.e., the results about reachability respectively the architectural constraints and the consequential definition of problematic states) are orthogonal to each other in the sense that they may be isolated and employed in different contexts. There

are other situations where statements about the reachability of global states are necessary. Basically every property of interaction systems only refers the set of reachable states. Since we detached the notions and conditions referring to reachability of states from the considerations about restricting the architecture and the implications for deadlock-freedom, the results presented in Section 3.2 can be applied in other contexts, as well. Also it should not prove too difficult to adapt the ideas to other automata based formalisms in order to obtain results about reachability there. On the other hand, different characteristic architectures may be conceived which might also allow to extract information about possible deadlocks from subsystems. In a similar vein one can turn to a different property either by maintaining the tree-like architecture or by devising a new one which is more appropriate for the property in question. Following the way in which the notion of problematic states was derived in Remark 3.4.1, one then has to characterize those combinations of local states which might lead to a violation of the property. The orthogonality also manifests itself in the fact that the different aspects of our approach take hold at the different layers of description for an interaction system that were mentioned in Section 2.1. The architectural constraint *only* restricts the interaction model of an interaction system, i.e., the static layer. Only the results about reachability and about the problematic states also refer to the local behavior of the components, i.e., the dynamic layer. The strict separation of communication from behavior enforced for interaction systems in fact helps to derive results about the behavior of a system since it allows to treat issues only concerning one of the layers in an isolated manner without having to tamper with the other.

We presented three real life examples that can be shown to be deadlock-free using the results of this chapter. There are of course deadlock-free systems where our results cannot be used to prove it. This is not surprising, though. The complexity results mentioned in Section 3.4 show that we cannot expect to find a characterization of deadlock-freedom that can be checked efficiently for all systems even if we confine the class of systems by only treating strongly tree-like interaction systems. We want to complete the discussion of the results by taking a closer look at the limitations of the approach and by hinting at some “back doors” that may be resorted to if

the conditions are not satisfied. First, it is clear that there is a large class of interaction systems that our criterion is not even capable of dealing with, precisely because we confine the interaction systems by demanding them to be tree-like. However, also in this subclass of interaction systems there are deadlock-free systems for which our criteria are not able to definitely answer the question whether there are reachable deadlocks. Recall that the criteria only offer a *sufficient* condition. We cannot make any statement about deadlock-freedom if the conditions are violated. There are two main reasons for this phenomenon. First, for components i and j the set of reachable states of $Sys \downarrow_{\{i,j\}}$ is an over-approximation of the projection of $reach(Sys)$ to $\{i, j\}$, i.e., $reach(Sys) \downarrow_{\{i,j\}} \subseteq reach(Sys \downarrow_{\{i,j\}})$. Thus, we may find local states which are problematic with respect to each other even though they can never be globally reached in this combination. Keeping track of the actions that lead to problematic states, as we do, helps to make the over-approximation more precise. Nonetheless, the possibility remains that we recognize conjectured deadlocks in states that are not globally reachable. Secondly, condition 2) of Proposition 3.4.1 makes sure that for every interaction (representing a global step) at least one component participates which cannot — by performing its port in the interaction — reach a state where all possible communication partners do not offer the required ports. If there is an interaction $\tilde{\alpha}$ violating this condition then $\tilde{\alpha}$ could lead to a state q where the components in $comp(\tilde{\alpha})$ constitute a set K' of components for which the possible communication partners block each other even though the system is not in a deadlock. This is because other components that are independent (in q) of the components in K' may be able to proceed. Such a state cannot be distinguished from a real deadlock state by only looking at subsystems of size two because we loose information by projecting Sys to such subsets. It may be possible that the other components proceed to a state that helps to resolve the blockade of the components in K'. On the other hand, it is of course also possible that the components in K' are blocked forever even though the system can still proceed. It may be desirable to also avoid this kind of situation, justifying the relatively sharp requirement which avoids the second situation described above.

The conditions in Proposition 3.4.1 have been devised such that their

violation can be understood as an indicator for where to perform a specific direct check for deadlocks respectively reachability. Such a violation is always directly linked to the components, local states, interactions, and ports that cause the violation. Thus, it may be possible to use the information gathered to pinpoint those global states and interactions that have to be checked directly. In the best case this direct check shows that no reachable deadlock arises. On the other hand, it may become evident that the violation of the conditions is indeed caused by a reachable deadlock. Even in this case the states and actions that fail to comply with the conditions may indicate in what ways the system has been designed faultily and how it has to be changed in order to avoid the deadlock. Only in cases where a direct check proves to be infeasible one might have to resort to different approaches to prove deadlock-freedom. Summarizing, we state that in case of failure of the criterion the information gathered may be used to investigate those states directly, i.e., based on the definition of deadlock-freedom, even though the proposition only states a *sufficient* condition for deadlock-freedom.

3.7.2 Related Work

Originally, the work presented in this chapter emanated from considerations by Bernardo et al. [33] respectively Baumeister et al. [30], the latter providing a more simple version of Sys_{bank} as discussed in Section 3.6.1. Bernardo et al. [33] present an abstract description language for component based systems called PADL. It uses process algebra to model a component system (called an *architectural type*) allowing multiway synchronization by several components over the same action. A notion of acyclic architectural type similar to the one we presented is given. It is also based on a graph which is required to be a tree. A criterion for deadlock-freedom of such architectural types is discussed. It ensures deadlock-freedom by requiring that no component i is restricted in its behavior by the communication with any of its possible partners j for cooperation, i.e., $Sys \downarrow_{\{i,j\}}$ restricted[4] to $\mathcal{A}_i$ should be weakly bisimular to T_i for all interacting com-

[4]Roughly speaking, this restriction is obtained by projecting all transition labels in $T_{Sys\downarrow_{\{i,j\}}}$ to i where every label which only involves j is replaced by τ.

ponents i and j. Consider the set $K := \{i, j, k\}$ of components where the transition systems of the components are given in Figure 3.24. We define $Int := \{\{a_i, a_j\}, \{b_i, b_j\}, \{c_j, c_k\}, \{d_j, d_k\}\}$. The induced interaction system is strongly tree-like. Up to nomenclature it can also be seen as an acyclic architectural type in PADL. Corollary 3.4.2 shows that the system is deadlock-free. However, Bernardo's condition fails since the behavior of j is restricted by the communication with i, which prevents j to ever choose the right branch of T_j. Even though the example is rather small it exhibits one

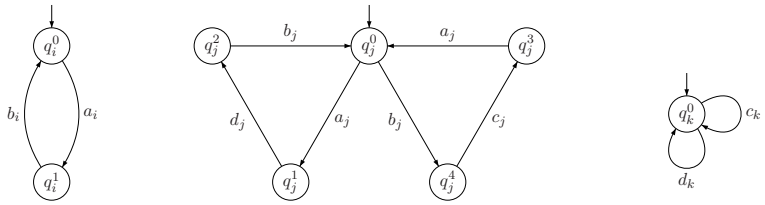

Figure 3.24: Component j is restricted by i

characteristic that will always cause Bernardo's condition to fail: If a component offers a choice between various services but another component only utilizes one of these services then all the other services will be "cut off" by the communication. Therefore the behavior of the service-component will be restricted. The conditions presented in this chapter allow for such a "partial use" of offered services as long as the component making use of the services is well-behaved (in the sense that it will not cause the subsystem consisting of these two components to reach a pair of states that are problematic with respect to each other). This shows that there are deadlock-free tree-like interaction systems where Bernardo's condition fails whereas Proposition 3.4.1 respectively Corollary 3.4.2 are sufficient for deadlock-freedom. Considering the opposite direction, it can be shown that Corollary 3.4.2 covers all deterministic[5] strongly tree-like systems that can be shown to be deadlock-free using Bernardo's condition. It is an open question whether there are tree-like systems that can be proven to be deadlock-free using Bernardo's condition while Proposition 3.4.1 fails, i.e., whether our criterion is more powerful than Bernardo's condition or whether the approaches are not comparable.

[5]In the sense that $q_i \xrightarrow{a_i}_i q_i'$ and $q_i \xrightarrow{a_i'}_i q_i''$ imply $a_i \neq a_i'$.

Systems having an underlying tree-like architecture are also considered by Baumeister et al. [30] and in the follow-up paper [79]. Further restrictions are imposed on the systems, though. The first paper presents a deadlock-freedom result for such a system consisting of three components only. The second paper extends these considerations to *star-like* component architectures of arbitrary size. The results presented in these papers elaborate on an idea similar to the one used by Bernardo et al. [33] also requiring that the communication between two components should not restrict these components' behavior. However, the authors focus on a different issue. The papers investigate the question what part of the behavior of a component has to be disclosed to possible communication partners and investigated with respect to these partners in order to still be able to make statements about deadlock-freedom of the composed system. The underlying idea is that there is no reason why a component i should have to inform j about a branch in its behavior that is altogether reserved for communication with a different component k. This idea is approached by providing each port with its own behavior (called a *port protocol*) which has to comply with the component's behavior in a certain sense. The criterion then only checks whether the protocols of connected ports are compatible. These ideas are of an interesting nature. They reflect the requirement stated in Section 1.1 that a component should encapsulate its behavior. An interesting field of future work on interaction systems unfurls because it seems promising to combine the notion of port protocols with our results for tree-like component architectures. To begin with we would have to define what it means for a port protocol to comply with the component's behavior. In particular, it is necessary to extend interaction systems by some sort of behavioral equivalence.

As noted in Section 3.4.2, the ideas of Brookes and Roscoe [38] show some intriguing parallels to our results with respect to strongly tree-like interaction systems. The authors consider networks of communicating processes in the failures model of CSP restricted to two-way communication. The networks they are dealing with are depicted by so-called communication graphs. The definition of these communication graphs is almost identical to G^*, the only difference being that Brookes and Roscoe [38] allow a node to also stand for a subsystem. Apart from that an edge also rep-

resents direct communication between its nodes. The deadlock-behavior of networks of communicating processes are investigated, and conditions for deadlock-freedom of networks whose communication graphs obey to certain restrictions are presented. The authors do not explicitly focus on tree-like structures. However, their statement which is equivalent to Corollary 3.4.3 is obtained as a corollary by further restricting the graph to be a tree. It is an interesting fact that Brookes and Roscoe [38], despite working into a different direction, obtain results that can be related to our approach.

Brookes and Roscoe [38] as well as Bernardo et al. [33] additionally treat systems whose communication architecture is not given by a tree. It is therefore clear that these works are more general in that regard. It should be noted, though, that this generality comes at the expense of not being able to only consider systems consisting of pairs of interacting components any more. For example, Bernardo et al. consider any subsystem consisting of components forming a cycle in G^*. Aside from the fact that it is not clear that there are reasonably few such cycles, Sys'_{phil_m} as introduced in Chapter 2 shows that such a subsystem may coincide with the original system. The investigation of this "subsystem" does not have any benefit compared to the consideration of the system itself. Nonetheless, extensions in this direction constitute a valuable addition with respect to our comments about the usefulness of a construction kit for component systems. Results as the ones by Brookes and Roscoe respectively Bernardo et al. can be incorporated in order to allow the construction of (sub)systems with a simple cyclic structure. Note that Brookes and Roscoe also interpret the results in a context considering them as part of a design rule guaranteeing deadlock-freedom.

Next, we take a closer look at another result on deadlock-freedom in interaction systems [102]. This approach also computes an over-approximation of the projection of $reach(Sys)$ to certain subsystems where the size of the subsystems can be customized according to a parameter d. The procedure gets more precise for larger d causing cost bounded by a polynomial of degree d. A sufficient condition for deadlock-freedom is checked on the over-approximation. Roughly speaking, this condition ensures that no state constitutes a chain of waiting relations between three components. Such a chain could in the worst case be completed by other components to form

a cycle of waiting relations in a global state and therefore cause a global deadlock. Corollary 3.4.2 is more powerful than the criterion by Majster-Cederbaum et al. [102] applied to strongly tree-like interaction systems. The condition above implies that for no interacting components i and j a pair of states is reachable which are problematic with respect to each other. Thus, for all strongly tree-like systems that can be shown to be deadlock-free using the criterion of Majster-Cederbaum et al. Corollary 3.4.2 also implies deadlock-freedom. On the other hand, it is easy to see that the condition of Majster-Cederbaum et al. fails to prove deadlock-freedom of Sys_{bank}: There are reachable global states where some ATM waits for its bank whereas the bank waits for the clearing company. Such states constitute a chain of waiting relations which causes a violation of the condition checked by Majster-Cederbaum et al.. By taking the tree-like structure into account we obtain a more powerful criterion for strongly tree-like systems compared to the criterion by Majster-Cederbaum et al. applied to such systems, *even though* we only consider subsystems of size two as opposed to size d. Turning to tree-like interaction systems, there are systems that can be proven to be deadlock-free using Proposition 3.4.1 while the condition of Majster-Cederbaum et al. fails but it is an open question whether our proposition is more powerful than this condition applied to such systems. Majster-Cederbaum and Minnameier [100] use a cross-checking technique to improve the over-approximation of the projection of $reach(Sys)$ to subsystems of size d computed by Majster-Cederbaum et al. [102]. There are no results, yet, on how the approach of Majster-Cederbaum and Minnameier [100] applied to systems with a tree-like architecture compares to our results.

Finally, we point out that Abdulla et al. [7] hint at an interesting extension of tree-like systems that may well be incorporated into the approach we presented here. Translated to our setting, this generalization results in interaction systems that are not necessarily tree-like. However, for each state q the "snapshot" of G with respect to q is a tree. Such a snapshot would be obtained by only taking the interactions in $Int(q_i)$ into account when possible edges involving i are considered. A more general setting is obtained because G may contain cycles when all interactions are considered. On the other hand, it seems likely that the ideas presented for tree-like interaction

systems can be transferred to this setting.

3.8 Proofs

3.8.1 Proofs for Section 3.2

Lemma 3.2.2. *Let Sys be an interaction system. Let $K' \subseteq K$ and $q, q' \in Q$. Let $\alpha \in Int$ be an interaction with $comp(\alpha) \cap K' \neq \emptyset$ and $q' \xrightarrow{\alpha} q$.*

1. *There is a transition $q' \downarrow_{K'} \xrightarrow{\alpha \downarrow_{K'}} q \downarrow_{K'}$ in $\tilde{T}_{Sys\downarrow_{K'}}$.*
2. *If q is reachable in Sys then $q \downarrow_{K'}$ is reachable in $Sys \downarrow_{K'}$.*

Proof. We show the two statements separately.

1. Because $comp(\alpha) \cap K' \neq \emptyset$ we know $\alpha \downarrow_{K'} \neq \emptyset$ and therefore $\alpha \downarrow_{K'} \in Int \downarrow_{K'}$. Together with $q' \xrightarrow{\alpha} q$ and the definition of the transition relation of an interaction system this directly implies that there is a transition $q' \downarrow_{K'} \xrightarrow{\alpha \downarrow_{K'}} q \downarrow_{K'}$ in $\tilde{T}_{Sys\downarrow_{K'}}$.
2. The proof of the lemma is a straightforward induction argument: Let σ be a path of length l from q^0 to q in *Sys*.
   $\underline{l = 0:}$ Then $q = q^0$. $q^0 \downarrow_{K'}$ is reachable in $Sys \downarrow_{K'}$.
   $\underline{l \Rightarrow l+1:}$ Let $\sigma = q^0 \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_{l-1}} q^l \xrightarrow{\alpha_l} q$ be a path of length $l+1$. By induction we know that $q^l \downarrow_{K'}$ is reachable in $Sys \downarrow_{K'}$. If $\alpha \downarrow_{K'} = \emptyset$ we do not have to show anything because in this case $q \downarrow_{K'} = q^l \downarrow_{K'}$ is reachable. Otherwise the first statement implies that there is a transition $q^l \downarrow_{K'} \xrightarrow{\alpha_l \downarrow_{K'}} q \downarrow_{K'}$ showing that $q \downarrow_{K'}$ is reachable.

□

Lemma 3.2.3. *Let Sys be an interaction system with strongly exclusive communication. Let $i \in K$, $\tilde{K} \subseteq K \setminus \{i\}$, and q_i such that $q_i^0 \notin BWS(q_i, \tilde{K})$. For each $j \in \tilde{K}$ let $Q'_j \subseteq Q_j$ be a nonempty subset of local states.*

If $\bigcap_{j \in \tilde{K}} \mathcal{EA}(q_i, \tilde{K}, Q'_j, \{i\}) = \emptyset$ then there is no $q \in reach(Sys)$ with $q \downarrow_{(\{i\} \cup \tilde{K})} \in \{q_i\} \times \prod_{j \in \tilde{K}} Q'_j$.

Proof. We make a general observation first: Consider $\alpha \in Int$ with $i(\alpha) \neq \emptyset$ but $\tilde{K} \cap comp(\alpha) = \emptyset$. Then $i(\alpha) \not\subseteq \bigcup_{k \in \tilde{K}} \mathit{comm}_i(k)$. Otherwise there

would be another interaction $\tilde{\alpha}$ with $\tilde{K} \cap comp(\tilde{\alpha}) \neq \emptyset$ and $\alpha \cap \tilde{\alpha} = i(\alpha)$. This is not possible because *Sys* has strongly exclusive communication.

Now assume that there is a global state $q \in reach\,(Sys)$ with $q \downarrow_{(\{i\} \cup \tilde{K})} \in \{q_i\} \times \prod_{j \in \tilde{K}} Q'_j$. There is a path

$$\sigma = q^0 \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_{l-1}} q^l \xrightarrow{\alpha_l} q.$$

There must be an index l' with $i\,(\alpha_{l'}) \neq \emptyset$ and $\tilde{K} \cap comp\,(\alpha_{l'}) \neq \emptyset$. Otherwise for each α_r on σ with $i\,(\alpha_r) \neq \emptyset$ the observation above implies $i\,(\alpha_r) \not\subseteq \bigcup_{k \in \tilde{K}} \boldsymbol{comm}_i\,(k)$. Then the projection of σ to i would yield a path from q_i^0 to q_i in T_i which is only labeled with ports that are not in $\bigcup_{k \in \tilde{K}} \boldsymbol{comm}_i\,(k)$, and $q_i^0 \in BWS(q_i, \tilde{K})$. This is a contradiction to the assumption. Let l_0 be the largest index on σ with $i\,(\alpha_{l_0}) \neq \emptyset$ and $\tilde{K} \cap comp\,(\alpha_{l_0}) \neq \emptyset$. Using the same argument as above we see that $q_i^{l_0+1} \in BWS(q_i, \tilde{K})$.

Choose $j \in \tilde{K}$. For all $l_0 < s \leq l$ the choice of l_0 implies $i\,(\alpha_s) = \emptyset$ if $j\,(\alpha_s) \neq \emptyset$. Therefore $q_j^{l_0+1} \in BWS(q_j, \{i\})$ where again we use the fact that *Sys* has strongly exclusive communication. We have $BWS(q_j, \{i\}) \subseteq BWS(Q'_j, \{i\})$ since $q_j \in Q'_j$. We conclude $(q_i^{l_0+1}, q_j^{l_0+1}) \in BWS(q_i, \tilde{K}) \times BWS(Q'_j, \{i\})$. Lemma 3.2.2 shows that $(q_i^{l_0}, q_j^{l_0}) \in reach(Sys \downarrow_{\{i,j\}})$ and that there is a transition $(q_i^{l_0}, q_j^{l_0}) \xrightarrow{\alpha_{l_0} \downarrow_{\{i,j\}}} (q_i^{l_0+1}, q_j^{l_0+1})$. This means $i\,(\alpha_{l_0}) \subseteq \mathcal{EA}(q_i, \tilde{K}, Q'_j, \{i\})$. Note that indeed $i\,(\alpha_{l_0}) \subseteq \bigcup_{k \in \tilde{K}} \boldsymbol{comm}_i\,(k)$. The other components in $\tilde{K}$ can be treated analogously, and we conclude $i\,(\alpha_{l_0}) \subseteq \bigcap_{j \in \tilde{K}} \mathcal{EA}(q_i, \tilde{K}, Q'_j, \{i\})$. This is a contradiction because the intersection is empty by assumption. □

3.8.2 Proofs for Section 3.3

Lemma 3.3.1. *Let IM be an interaction model.*

If IM is strongly tree-like then $|\alpha| \leq 2$ *for all* $\alpha \in Int$.

Proof. For any $\alpha \in Int$ it is clear that $G^*_\alpha := (comp\,(\alpha), E_\alpha)$ with $E_\alpha := \{e | e \in E \wedge e \subseteq comp\,(\alpha)\} \subseteq E$ constitutes a complete subgraph of G^*. Thus, *IM* always contains cycles if there exists $\alpha \in Int$ with $|comp\,(\alpha)| > 2$. Consequently, if *IM* is strongly tree-like all interactions are binary. □

Lemma 3.3.2. *Let IM be an interaction model.*

1. *If IM is strongly tree-like then it is tree-like.*
2. *If for all $\alpha \in Int$ we have $|\alpha| \leq 2$ and IM is tree-like then IM is strongly tree-like.*

Proof. We prove the statements separately.

1. Let *IM* be strongly tree-like. Assume that G contains a cycle. Because of Lemma 3.3.1 at most two components participate in any interaction α. Since E does not contain any edges connecting two nodes representing components the cycle must be of the form

   $$\{i_0\} - comp(\alpha_0) - \{i_1\} - \ldots - \{i_{m-1}\} - comp(\alpha_{m-1}) - \{i_0\}$$

   where $\alpha_l \in Int$ and $comp(\alpha_l) = \{i_l, i_{l+1}\}$ (the indices are calculated $\bmod m$). We get $\{i_l, i_{l+1}\} \in E^*$ for all l, and G^* contains a cycle which is a contradiction.

   On the other hand, assume that G is not connected. There are nodes K' and K'' in V such that no path connecting these nodes exists. At least one node representing a component is reachable from every node in V_2. Therefore, without loss of generality, we assume $K' = \{i\}$ and $K'' = \{j\}$. In G^* there exists a path connecting i and j because G^* is a tree. For every edge $\{i_l, i_{l+1}\}$ on this path there is some $\alpha_l \in Int$ with $comp(\alpha_l) = \{i_l, i_{l+1}\}$. This means that E contains edges $\{i_l, comp(\alpha_l)\}$ and $\{comp(\alpha_l), i_{l+1}\}$, and the path can be transferred to G, showing that $\{i\}$ and $\{j\}$ are connected. This is a contradiction.

2. Let *IM* tree-like, and let $|\alpha| \leq 2$ for all $\alpha \in Int$. Assume that G^* contains a cycle π. Reasoning the same way as in the second argument above, we see that π can be transferred to G. This is a contradiction.

   Assume that G^* is not connected. Choose components i and j such that no path connecting these components exists. There is a path connecting $\{i\}$ and $\{j\}$ in G. The same way as in the first argument above this path can be transferred to G^*. This is a contradiction.

□

Lemma 3.3.3. *Let IM be an interaction model.*

1. *For all $\alpha \in Int$ there is a node $comp(\alpha) \in V_2$.*

2. *Let $\alpha \in Int$ and $i \in comp(\alpha)$. There is a simple path $\pi_{i,\alpha}$ connecting $\{i\}$ and $comp(\alpha)$ in G. All nodes on $\pi_{i,\alpha}$ are subsets of $comp(\alpha)$. If $|comp(\alpha)| \geq 2$ then all nodes on $\pi_{i,\alpha}$ except $\{i\}$ contain two components at least.*

3. *Let $\alpha \in Int$ and $i, j \in comp(\alpha)$. There is a simple path $\pi^{\alpha}_{i,j}$ connecting $\{i\}$ and $\{j\}$ in G. All nodes on $\pi^{\alpha}_{i,j}$ are subsets of $comp(\alpha)$. All nodes on $\pi_{i,\alpha}$ except $\{i\}$ and $\{j\}$ contain two components at least. The paths $\pi^{\alpha}_{i,j}$, $\pi_{i,\alpha}$, and $\pi_{j,\alpha}$ can be chosen such that there is a node K' on $\pi^{\alpha}_{i,j}$ such that the sub-path of $\pi^{\alpha}_{i,j}$ from $\{i\}$ to K' is a sub-path of $\pi_{i,\alpha}$ and the sub-path of $\pi^{\alpha}_{i,j}$ from K' to $\{j\}$ is a sub-path of $\pi_{j,\alpha}$.*

4. *Let $\alpha, \alpha' \in Int$ with $|comp(\alpha) \cap comp(\alpha')| \geq 2$ and let $i \in comp(\alpha)$ and $j \in comp(\alpha')$. There is a simple path $\pi^{\alpha,\alpha'}_{i,j}$ connecting $\{i\}$ and $\{j\}$ in G. All nodes on $\pi^{\alpha,\alpha'}_{i,j}$ are subsets of $comp(\alpha)$ or $comp(\alpha')$. The first node after $\{i\}$ is contained in $comp(\alpha)$ and the last node before $\{j\}$ is contained in $comp(\alpha')$. All nodes on $\pi^{\alpha,\alpha'}_{i,j}$ except $\{i\}$ and $\{j\}$ contain two components at least.*

Proof. We prove the statements separately.

1. Setting $\alpha' = \alpha$ in Definition 3.3.1 we see that there is a node $comp(\alpha) \cap comp(\alpha) = comp(\alpha) \in V_2$.

2. From the first statement we know $comp(\alpha) \in V$. We also have $\{i\} \in V$. Consider a sequence $\{i\} = K_1, K_2, \ldots K_m = comp(\alpha)$ of subsets of components having the following properties:

   (a) For all l we have $K_l \in V$.

   (b) For all $1 \leq l \leq m-1$ we have $K_l \subseteq K_{l+1}$, and there is no $K' \in V$ with $K_l \subsetneq K' \subsetneq K_{l+1}$.

   It is clear that there is such a sequence. For all $1 \leq l \leq m-1$ there is an edge $\{K_l, K_{l+1}\} \in E$. The path $\pi_{i,\alpha} := K_1 — \ldots — K_m$ has the required properties.

3. Consider $\pi_{i,\alpha}$ and $\pi_{j,\alpha}$ as defined in the previous statement. Let $K' \in V$ be the first node (starting to count from $\{i\}$ respectively $\{j\}$) that occurs on both paths. It is clear that K' exists because both paths end in $comp(\alpha)$. The path $\pi^{\alpha}_{i,j} := \{i\} - \ldots - K' - \ldots - \{j\}$ obtained by connecting the fragments of the two paths ending in K' has the required properties.

4. If $comp(\alpha) = comp(\alpha')$ the statement follows from the previous part of the lemma.

   Otherwise consider the nodes $comp(\alpha)$, $comp(\alpha')$, and $comp(\alpha) \cap comp(\alpha')$ in V. Since $comp(\alpha) \cap comp(\alpha') \subseteq comp(\alpha)$ and $comp(\alpha) \cap comp(\alpha') \subseteq comp(\alpha')$ we may construct simple paths $\pi^{\alpha,\alpha'}_{\alpha}$ connecting $comp(\alpha)$ with $comp(\alpha) \cap comp(\alpha')$ respectively $\pi^{\alpha,\alpha'}_{\alpha'}$ connecting $comp(\alpha')$ with $comp(\alpha) \cap comp(\alpha')$ in the same way as in the proof of the second statement. All nodes on these two paths contain two components at least because $|comp(\alpha) \cap comp(\alpha')| \geq 2$. Piecing together the paths $\pi_{i,\alpha}$ and $\pi^{\alpha,\alpha'}_{\alpha}$ in $comp(\alpha)$, $\pi^{\alpha,\alpha'}_{\alpha}$ and $\pi^{\alpha,\alpha'}_{\alpha'}$ in $comp(\alpha) \cap comp(\alpha')$, and $\pi^{\alpha,\alpha'}_{\alpha'}$ and $\pi_{j,\alpha'}$ in $comp(\alpha')$ we obtain a path connecting $\{i\}$ and $\{j\}$ that has all the required properties with the exception that it may not be simple. It is clear that it contains a simple path $\pi^{\alpha,\alpha'}_{i,j}$ with the required properties.

□

3.8.3 Proofs for Section 3.4.1

The proof of Proposition 3.4.1 consists of two steps. First we state several lemmas and corollaries. Then we use these results and Lemma 3.2.2 to prove the proposition itself.

We formulate the following lemmas in a more general fashion than necessary at the moment. The lemmas applied to the concrete situation we are dealing with are stated as corollaries. The general formulation requires a slight generalization in the notions. We introduce an auxiliary definition. It will only be needed for the proofs of the lemmas.

Definition 3.8.1. *Let Sys be an interaction system. Let $i \in K$ and $q_i \in Q_i$.*

Let $Int'(q_i) \subseteq Int(q_i)$ be a nonempty subset of interactions. We define:

$$need'(q_i) := \bigcup_{\alpha \in Int'(q_i)} comp(\alpha) \backslash \{i\}$$

We need some preliminary notation. For better readability we introduce it before stating the following lemmas. Consider $\alpha \in Int$ with $|comp(\alpha)| \geq 2$ and $j \in comp(\alpha)$. Let

$$\pi_{j,\alpha} = \{j\} \overset{e}{—} \bar{K} — \ldots — comp(\alpha)$$

be a path connecting $\{j\}$ and $comp(\alpha)$ in G as constructed in Lemma 3.3.3. We have $\bar{K} \subseteq comp(\alpha)$ and $|\bar{K}| \geq 2$. Denote by $\bar{G} := (V, E \backslash \{e\})$ the subgraph of G which is obtained by removing the edge e.

Lemma 3.8.1. *Let Sys be an interaction system and let $q \in Q$ be a state. For all $i \in K$ let $Int'(q_i) \subseteq Int(q_i)$ be a nonempty subset of interactions. Let $\alpha \in Int$ and $j \in comp(\alpha)$ such that $need'(q_j) \cap comp(\alpha) = \emptyset$. Let $k \in K$ be a component such that $\{k\}$ is reachable from $\{j\}$ in $\bar{G}$ where $\bar{G}$ is defined as above for α and j. Further, let $\beta' \in Int'(q_k)$ and $\beta \in Int$ with $|comp(\beta') \cap comp(\beta)| \geq 2$.*

For all $l \in comp(\beta)$ the node $\{l\} \in V$ is reachable from $\{j\}$ in $\bar{G}$.

Proof. If $l = k$ it is clear that $\{l\}$ is reachable from $\{j\}$ in $\bar{G}$. Let $l \in comp(\beta) \backslash \{k\}$. Consider a simple path $\pi_{k,l}^{\beta',\beta}$ connecting $\{k\}$ and $\{l\}$ in G as constructed in Lemma 3.3.3. We consider cases:

1. $l = j$: Then it is clear that $\{l\}$ is reachable from $\{j\}$ in $\bar{G}$.

2. $l \neq j$ and $k \neq j$: The edge e which was removed from E connects the two nodes $\{j\}$ and $\bar{K}$. The only two nodes K' occurring on $\pi_{k,l}^{\beta',\beta}$ with $|K'| = 1$ are $\{k\}$ and $\{l\}$ both of which are not equal to $\{j\}$. Therefore e does not occur on $\pi_{k,l}^{\beta',\beta}$, and the path is not affected by the removal of e. Thus, $\{l\}$ is reachable from $\{k\}$ in $\bar{G}$. $\{k\}$ is reachable from $\{j\}$ in $\bar{G}$ by assumption and therefore $\{l\}$ is reachable from $\{j\}$ in $\bar{G}$.

3. $k = j$: In this case $need'(q_j) \cap comp(\alpha) = \emptyset$ implies $comp(\alpha) \cap comp(\beta') = \{j\}$: It is clear that j participates in both interactions.

There cannot be any other component in this intersection because this component would also be contained in $need'(q_j) \cap comp(\alpha)$.

The only edge on $\pi_{k,l}^{\beta',\beta}$ which can possibly coincide with e is the first edge because it is the only edge on $\pi_{k,l}^{\beta',\beta}$ which ends in $\{k\} = \{j\}$. Assume that this edge coincides with e. Then $\bar{K} \subseteq comp(\beta')$ because Lemma 3.3.3 states that at least the first node on $\pi_{k,l}^{\beta',\beta}$ is contained in $comp(\beta')$. We get $\bar{K} \subseteq comp(\alpha) \cap comp(\beta')$. This is a contradiction because $|\bar{K}| \geq 2$ on the one hand but $comp(\alpha) \cap comp(\beta') = \{j\}$ on the other. We conclude that e does not occur on $\pi_{k,l}^{\beta',\beta}$. Then $\pi_{k,l}^{\beta',\beta}$ is not affected by the removal of e, and $\{l\}$ is reachable from $\{j\}$ in $\bar{G}$.

□

The following two lemmas are the only results that directly use the fact that the system is tree-like.

Lemma 3.8.2. *Let Sys be a tree-like interaction system and let $q \in Q$ be a state. For all $i \in K$ let $Int'(q_i) \subseteq Int(q_i)$ be a nonempty subset of interactions such that $need'(q_i) \neq \emptyset$. Let $\tilde{K} \subseteq K$ be a nonempty subset of components such that for all $i \in \tilde{K}$, all $\alpha' \in Int'(q_i)$, and all $\alpha \in Int$ with $|comp(\alpha') \cap comp(\alpha)| \geq 2$ we have $comp(\alpha) \subseteq \tilde{K}$.*

There exists a nonempty set $K' \subseteq \tilde{K}$ such that for all $i \in K'$ and all $\alpha' \in Int'(q_i)$ the following two conditions hold:

1. *$\forall \alpha \in Int$ with $|comp(\alpha') \cap comp(\alpha)| \geq 2$ we have $comp(\alpha) \subseteq K'$*
2. *$\forall j \in comp(\alpha') \setminus \{i\} : need'(q_j) \cap comp(\alpha') \neq \emptyset$*

For $Int' = Int$ the second condition restates the condition that we deduced in Remark 3.4.1 in order to check whether a pair of local states can possibly be involved in a deadlock.

Proof. Setting $\alpha = \alpha'$ in the condition for $\tilde{K}$ we get $comp(\alpha') \subseteq \tilde{K}$ for all $i \in \tilde{K}$ and all $\alpha' \in Int'(q_i)$ with $|comp(\alpha')| \geq 2$. Therefore $need'(q_i) \subseteq \tilde{K}$ for all $i \in \tilde{K}$. Since $need'(q_i) \neq \emptyset$ for all $i \in \tilde{K}$ and $\tilde{K} \neq \emptyset$ this means that $\tilde{K}$ contains at least two elements.

We will prove the lemma by induction on $m = |\tilde{K}|$. According to the previous argument the induction has to start for $m = 2$.

$\underline{m = 2:}$ Then $\tilde{K} = \{i, j\}$. We show that condition 2 above holds for $\tilde{K}$. We have already assessed $need'(q_i) \subseteq \{i, j\}$ and $need'(q_j) \subseteq \{i, j\}$. Together with $need'(q_i) \neq \emptyset$ and $need'(q_j) \neq \emptyset$ this implies $need'(q_i) = \{j\}$ and $need'(q_j) = \{i\}$. Choose $\alpha' \in Int'(q_i)$. We have $comp(\alpha') \subseteq \tilde{K} = \{i, j\}$. The second condition trivially holds if $\alpha' = \{a_i\}$. Therefore assume $comp(\alpha') = \{i, j\}$. We already know $need'(q_j) = \{i\}$. Therefore $need'(q_j) \cap comp(\alpha') = \{i\}$, and the second condition is satisfied. An interaction $\beta' \in Int'(q_j)$ is treated analogously. This shows that condition 2 is satisfied for both components in $\{i, j\}$ and all $\alpha' \in Int'(q_i)$ respectively $\beta' \in Int'(q_j)$. We choose $K' = \tilde{K}$.

$\underline{m \Rightarrow m + 1:}$ Let $|\tilde{K}| = m + 1$. If for all $i \in \tilde{K}$, all $\alpha' \in Int'(q_i)$, and all $j \in comp(\alpha') \setminus \{i\}$ we have $need'(q_j) \cap comp(\alpha') \neq \emptyset$ then $\tilde{K}$ satisfies condition 2. Condition 1 is satisfied by assumption. Choose $K' = \tilde{K}$.

Otherwise there exist $i \in \tilde{K}$, an interaction $\alpha' \in Int'(q_i)$, and a component $j \in comp(\alpha') \setminus \{i\}$ with $need'(q_j) \cap comp(\alpha') = \emptyset$. From $\alpha' \in Int'(q_i)$ we get $j \in need'(q_i)$ and therefore $j \in \tilde{K}$. According to Lemma 3.3.3 consider

$$\pi^{\alpha'}_{j,i} = \{j\} \overset{e}{-} \bar{K} - \ldots - \{i\}$$

the simple path connecting $\{j\}$ and $\{i\}$ in G. All nodes between $\{j\}$ and $\{i\}$ are subsets of $comp(\alpha')$ and contain at least two components. Furthermore, the first edge e of $\pi^{\alpha'}_{j,i}$ is the same edge as the first edge on the path $\pi_{j,\alpha'}$ between $\{j\}$ and $comp(\alpha')$. Therefore $\bar{G} := (V, E \setminus \{e\})$ is the same graph as defined before Lemma 3.8.1. We denote by $\hat{K}$ the set of those $k \in \tilde{K}$ where $\{k\}$ is reachable from $\{j\}$ in $\bar{G}$:

$$\hat{K} := \tilde{K} \cap \big\{k | \{k\} \text{ is reachable from } \{j\} \text{ in } \bar{G}\big\}$$

The following facts hold for $\hat{K}$:

1. $j \in \hat{K}$. This is clear.
2. $i \notin \hat{K}$: Assume $i \in \hat{K}$. Then there would be a path π' connecting $\{j\}$ and $\{i\}$ in $\bar{G}$. This path does not involve the edge e. Therefore $\pi^{\alpha'}_{j,i} \neq \pi'$, and there would be two different paths connecting $\{i\}$ and $\{j\}$ in G. This is not possible because G is a tree.

3. For all $k \in \hat{K}$, all $\beta' \in \mathit{Int}'(q_k)$, and all $\beta \in \mathit{Int}$ such that $|\mathit{comp}(\beta') \cap \mathit{comp}(\beta)| \geq 2$ we have $\mathit{comp}(\beta) \subseteq \hat{K}$: On the one hand, $k \in \hat{K}$ implies that $\{k\}$ is reachable from $\{j\}$ in $\bar{G}$. Lemma 3.8.1 then shows that for all components $l \in \mathit{comp}(\beta)$ the node $\{l\}$ is also reachable from $\{j\}$ in $\bar{G}$. On the other hand, $\mathit{comp}(\beta) \subseteq \tilde{K}$ follows from the assumption about $\tilde{K}$ because $k \in \hat{K}$ also means $k \in \tilde{K}$. We conclude $\mathit{comp}(\beta) \subseteq \hat{K}$.

The first fact above shows $\hat{K} \neq \emptyset$. It is clear that $\hat{K} \subseteq \tilde{K}$. The second fact shows that $\hat{K}$ is a proper subset of $\tilde{K}$. Finally, the third fact states that $\hat{K}$ again satisfies the condition required for $\tilde{K}$ in the lemma. We have already seen that this implies $\mathit{need}'(q_k) \subseteq \hat{K}$ for all $k \in \hat{K}$. Together with $\hat{K} \neq \emptyset$ and $\mathit{need}'(q_k) \neq \emptyset$ for all $k \in \hat{K}$ this means that $\hat{K}$ contains at least two elements.

Summarizing, we assess that $\hat{K}$ satisfies the condition required for $\tilde{K}$ and that $2 \leq |\hat{K}| \leq m$. Applying the induction hypothesis to $\hat{K}$ we see that there is a set $K' \subseteq \hat{K} \subseteq \tilde{K}$ as required.

□

We only used the requirement $\mathit{need}'(q_i) \neq \emptyset$ for those i that are contained in $\tilde{K}$. This means that we could have formulated the lemma based on the less restrictive assumption $\mathit{need}'(q_i) \neq \emptyset$ for all $i \in \tilde{K}$.

Corollary 3.8.1. *Let Sys be a tree-like interaction system. Let $q \in Q$ be a state with $\mathit{need}(q_i) \neq \emptyset$ for all $i \in K$.*

There exists a nonempty subset K' of components such that for all $i \in K'$ and for all $\alpha' \in \mathit{Int}(q_i)$ the following two conditions hold:

1. *$\forall \alpha \in \mathit{Int}$ with $|\mathit{comp}(\alpha') \cap \mathit{comp}(\alpha)| \geq 2$ we have $\mathit{comp}(\alpha) \subseteq K'$*

2. *$\forall j \in \mathit{comp}(\alpha') \setminus \{i\} : \mathit{need}(q_j) \cap \mathit{comp}(\alpha') \neq \emptyset$*

Proof. K satisfies the condition required for $\tilde{K}$ in Lemma 3.8.2 because $\mathit{comp}(\alpha) \subseteq K$ for all interactions α. Applying Lemma 3.8.2 with $\tilde{K} = K$ and $\mathit{Int}'(q_i) = \mathit{Int}(q_i)$ for all i yields the statement of the corollary. □

Returning to the setting described in Lemma 3.8.2 it is clear that K' can always be chosen to be minimal in the sense that no proper subset of K' also has the two properties. We get the following result:

Lemma 3.8.3. *Let Sys be a tree-like interaction system and let $q \in Q$ be a state. For all $i \in K$ let $Int'(q_i) \subseteq Int(q_i)$ be a nonempty subset of interactions such that $need'(q_i) \neq \emptyset$. Let K' be a minimal nonempty set of components satisfying the two conditions stated in Lemma 3.8.2. Let $\alpha \in Int$ be an interaction with $comp(\alpha) \subseteq K'$ and $|comp(\alpha)| \geq 2$. Let $j \in comp(\alpha)$ be a component.*

We have $need'(q_j) \cap comp(\alpha) \neq \emptyset$.

Proof. By way of contradiction assume $need'(q_j) \cap comp(\alpha) = \emptyset$. We will show that in this case K' is not minimal.

According to Lemma 3.3.3 let

$$\pi_{j,\alpha} = \{j\} \overset{e}{—} \bar{K} — \ldots — comp(\alpha)$$

be the path connecting $\{j\}$ and $comp(\alpha)$ in G such that all nodes in between $\{j\}$ and $comp(\alpha)$ are contained in $comp(\alpha)$. Consider $\bar{G} := (V, E \setminus \{e\})$ as defined before Lemma 3.8.1. We denote by $\hat{K}$ the set of components in K' that are reachable from $\{j\}$ in $\bar{G}$:

$$\hat{K} := K' \cap \big\{k | \{k\} \text{ is reachable from } \{j\} \text{ in } \bar{G}\big\}$$

We have $\hat{K} \subseteq K'$. We show that it is a proper subset. By assumption we know $|comp(\alpha)| \geq 2$. Choose $i \in comp(\alpha) \setminus \{j\}$. We claim that i is not contained in $\hat{K}$. Assume this was the case. Then there must be a path connecting $\{j\}$ and $\{i\}$ in $\bar{G}$. This path does not involve the edge e. On the other hand according to Lemma 3.3.3 in G there is also the path $\pi^{\alpha}_{j,i}$ connecting $\{j\}$ and $\{i\}$ whose first edge is e. This means that there are two different paths between $\{j\}$ and $\{i\}$ in G which is not possible because G is a tree. We conclude $i \notin \hat{K}$. Therefore $\hat{K}$ is a proper subset of K'. The following facts hold for $\hat{K}$:

1. $j \in \hat{K}$. This is clear.

2. For $k \in \hat{K}$, $\beta' \in Int'(q_k)$, and $\beta \in Int$ with $|comp(\beta') \cap comp(\beta)| \geq 2$ we have $comp(\beta) \subseteq \hat{K}$. Using Lemma 3.8.1 this is shown exactly the same way as the corresponding fact in the proof of the previous lemma.

These two facts show that $\hat{K}$ is a proper nonempty subset of K' that satisfies the requirements stated for $\tilde{K}$ in Lemma 3.8.2. Lemma 3.8.2 then shows that there exists $\breve{K} \subseteq \hat{K}$ satisfying both conditions stated in the lemma. Because $\breve{K}$ is a proper subset of K' we conclude that K' is not a minimal set satisfying the two conditions. This is a contradiction. We conclude $need'(q_j) \cap comp(\alpha) \neq \emptyset$. □

Corollary 3.8.2. *Let Sys be a tree-like interaction system. Let $q \in Q$ be a state with $need(q_i) \neq \emptyset$ for all $i \in K$. Let K' be a minimal nonempty set of components satisfying the two conditions stated in Corollary 3.8.1. Let $\alpha \in Int$ be an interaction with $comp(\alpha) \subseteq K'$ and $|comp(\alpha)| \geq 2$. Let $j \in comp(\alpha)$ be a component.*

We have $need(q_j) \cap comp(\alpha) \neq \emptyset$.

Proof. Applying Lemma 3.8.3 with $Int'(q_i) = Int(q_i)$ for all i yields the statement of the corollary. □

Lemma 3.8.4. *Let Sys be a tree-like interaction system and let $q \in Q$. For all $i \in K$ let $Int'(q_i) \subseteq Int(q_i)$ be a nonempty subset of interactions such that $need'(q_i) \neq \emptyset$. Let K' be a nonempty set of components satisfying the two conditions stated in Lemma 3.8.2. Let $i \in K'$ and $\alpha \in Int$ with $i(\alpha) \neq \emptyset$.*

If there is $k \notin K'$ with $k(\alpha) \neq \emptyset$ then $comp(\alpha) \cap need'(q_i) = \emptyset$.

Proof. Let $j \in comp(\alpha) \setminus \{i\}$ and assume $j \in need'(q_i)$. There must be an interaction $\alpha' \in Int'(q_i)$ with $j \in comp(\alpha')$. We get $\{i, j\} \subseteq comp(\alpha') \cap comp(\alpha)$ and therefore $|comp(\alpha') \cap comp(\alpha)| \geq 2$. Since $i \in K'$ and $\alpha' \in Int'(q_i)$ the first property of K' implies $comp(\alpha) \subseteq K'$. This is a contradiction because $k \in comp(\alpha)$ but $k \notin K'$.

We conclude $comp(\alpha) \cap need'(q_i) = \emptyset$. □

Corollary 3.8.3. *Let Sys be a tree-like interaction system with strongly exclusive communication. Let $q \in Q$ be a state with $need(q_i) \neq \emptyset$ for all*

$i \in K$. Choose K' as in Corollary 3.8.1. Let $i \in K'$ and $\alpha \in Int$ with $i(\alpha) \neq \emptyset$.

If there is $k \notin K'$ with $k(\alpha) \neq \emptyset$ then $i(\alpha) \not\subseteq \bigcup_{j \in need(q_i)} \mathit{comm}_i(j)$.

Proof. Setting $Int'(q_i) = Int(q_i)$ in Lemma 3.8.4 we see $comp(\alpha) \cap need(q_i) = \emptyset$. Assume $i(\alpha) \subseteq \bigcup_{j \in need(q_i)} \mathit{comm}_i(j)$. Since $comp(\alpha) \cap need(q_i) = \emptyset$ there must be an interaction $\alpha' \neq \alpha$ with $comp(\alpha') \cap need(q_i) \neq \emptyset$ and $i(\alpha') = i(\alpha)$. This is impossible since *Sys* has strongly exclusive communication. □

The previous lemma and corollary only use the first defining property of K'. We could have formulated these results for a set $\tilde{K}$ only satisfying:

$$\forall i \in \tilde{K}\ \forall \alpha' \in Int'(q_i)\ \ \forall \alpha \in Int \text{ with}$$

$$\left|comp(\alpha') \cap comp(\alpha)\right| \geq 2 \text{ we have } comp(\alpha) \subseteq \tilde{K}$$

The following lemma makes a statement about the existence of problematic states. Again, we want to state a general lemma. This necessitates a generalization of the notion of problematic states

Definition 3.8.2. *Let Sys be a tree-like interaction system. Let for all $i \in K$ and all $q_i \in Q_i$ subsets $Int'(q_i) \subseteq Int(q_i)$ be given. For $i \in K$, $q_i \in Q_i$ such that there is no $\alpha \in Int'(q_i)$ with $|\alpha| = 1$, $\alpha \in Int'(q_i)$, and $j \in comp(\alpha) \backslash \{i\}$ we define a descending sequence of subsets of Q_j by:*

$$\begin{aligned} PS'^{0}_{j}(q_i, \alpha) := \{q_j \mid &\bullet\ \nexists \beta \in Int'(q_j) \text{ with } |\beta| = 1 \\ &\bullet\ need'(q_j) \cap comp(\alpha) \neq \emptyset \\ &\bullet\ (q_i, q_j) \text{ is reachable in } Sys\downarrow_{\{i,j\}} \\ &\bullet\ \alpha \notin Int'(q_j) \\ &\bullet\ \nexists \tilde{\alpha} \in Int'(q_i) \cap Int'(q_j) \text{ with } |\tilde{\alpha}| = 2\} \end{aligned}$$

$$\begin{aligned} PS'^{l+1}_{j}(q_i, \alpha) := \{q_j \mid &q_j \in PS'^{l}_{j}(q_i, \alpha) \text{ and } \forall \beta \in Int'(q_j) \exists k \in comp(\beta) \backslash \{j\} \\ &\text{with } PS'^{l}_{k}(q_j, \beta) \neq \emptyset\} \end{aligned}$$

We set:

$$PS'_j(q_i, \alpha) := \bigcap_{l \in \mathbb{N}} PS'^{l}_{j}(q_i, \alpha)$$

It is not surprising that this notion of problematic states coincides with the one presented in Definition 3.4.1 (cf. p. 60) if we choose $Int'(q_i) = Int(q_i)$ for all i and all q_i. This statement is formalized by the following lemma.

Lemma 3.8.5. *Let Sys be a tree-like interaction system. Let $i \in K$, q_i incomplete, $\alpha \in Int(q_i)$, and $j \in comp(\alpha)\backslash\{i\}$.*

If $Int'(q_k) = Int(q_k)$ for all $k \in K$ then

$$PS_j^l(q_i, \alpha) = PS_j'^l(q_i, \alpha)$$

for all $l \in \mathbb{N}$.

Proof. We will show that each of the conditions stated in the definition of the problematic states (Definition 3.4.1) is equivalent to the corresponding condition in Definition 3.8.2. It is clear that q_i is incomplete if and only if there is no $\alpha \in Int'(q_i) = Int(q_i)$ with $|\alpha| = 1$. Thus, the precondition required for q_i in Definition 3.4.1 is equivalent to the precondition required for q_i in Definition 3.8.2. The actual proof uses induction over l.

$\underline{l = 0:}$ Setting $Int'(q_i) = Int(q_i)$ for all i and all q_i in Definition 3.8.2 the conditions stated there are equivalent to the corresponding conditions in Definition 3.4.1: As above we argue that q_j is incomplete if and only if there is no $\beta \in Int'(q_j) = Int(q_j)$ with $|\beta| = 1$. Using the fact that $Int'(q_j) = Int(q_j)$ also implies $need'(q_j) = need(q_j)$, we see that the second condition above is equivalent to the corresponding one in Definition 3.4.1. The condition about reachability of (q_i, q_j) is the same in both definitions. If $Int'(q_j) = Int(q_j)$ then it is clear that $\alpha \notin Int'(q_j)$ if and only if $j(\alpha) \not\subseteq \mathcal{en}(q_j)$ showing that the fourth conditions are equivalent. Finally, the last condition stated above is the same as the last condition in Definition 3.4.1 if $Int'(q_i) = Int(q_i)$ and $Int'(q_j) = Int(q_j)$.

We conclude $PS_j^0(q_i, \alpha) = PS_j'^0(q_i, \alpha)$.

$\underline{l \Rightarrow l+1:}$ We have:

$$PS_j^{l+1}(q_i, \alpha) = \{q_j | q_j \in PS_j^l(q_i, \alpha) \text{ and } \forall \beta \in Int(q_j) \exists k \in comp(\beta)\backslash\{j\} \text{ with } PS_k^l(q_j, \beta) \neq \emptyset\}$$

$$
\begin{aligned}
&= \{q_j | q_j \in PS_j^l(q_i, \alpha) \text{ and } \forall \beta \in Int'(q_j) \exists k \in \\
&\qquad comp(\beta) \setminus \{j\} \text{ with } PS_k^l(q_j, \beta) \neq \emptyset\} \\
&= \{q_j | q_j \in PS_j'^l(q_i, \alpha) \text{ and } \forall \beta \in Int'(q_j) \exists k \in \\
&\qquad comp(\beta) \setminus \{j\} \text{ with } PS_k'^l(q_j, \beta) \neq \emptyset\} \\
&= PS_j'^{l+1}(q_i, \alpha)
\end{aligned}
$$

The second equality uses $Int'(q_j) = Int(q_j)$ whereas the third equality applies the induction hypothesis.

□

Lemma 3.8.6. *Let Sys be a tree-like interaction system and let $q \in Q$ be a reachable state. For all $i \in K$ and all $q_i \in Q_i$ let $Int'(q_i) \subseteq Int(q_i)$ be a nonempty subset of interactions. Let $\tilde{K} \subseteq K$ be a set of components such that for all $i \in \tilde{K}$ and all $\alpha \in Int'(q_i)$ there exists $j \in comp(\alpha) \setminus \{i\} \cap \tilde{K}$ with $\alpha \notin Int'(q_j)$ and $need'(q_j) \cap comp(\alpha) \neq \emptyset$.*

For all $i \in \tilde{K}$ and all $\alpha \in Int'(q_i)$ there exists $j \in comp(\alpha) \setminus \{i\} \cap \tilde{K}$ with $q_j \in PS'_j(q_i, \alpha)$.

Proof. We make the following observation: If $i \in \tilde{K}$ then there is no $\alpha \in Int'(q_i)$ with $|\alpha| = 1$. For such α we have $comp(\alpha) \setminus \{i\} = \emptyset$ and there would not be any j satisfying the condition required for $i \in \tilde{K}$ and $\alpha \in Int'(q_i)$.

Let $i \in \tilde{K}$ and $\alpha_0 \in Int'(q_i)$. Choose $j_0 \in comp(\alpha_0) \setminus \{i\} \cap \tilde{K}$ with $\alpha_0 \notin Int'(q_{j_0})$ and $need'(q_{j_0}) \cap comp(\alpha_0) \neq \emptyset$. We assess:

- $\nexists \beta \in Int'(q_{j_0})$ with $|\beta| = 1$ because $j_0 \in \tilde{K}$.
- $need'(q_{j_0}) \cap comp(\alpha_0) \neq \emptyset$ by assumption.
- (q_i, q_{j_0}) is reachable in $Sys \downarrow_{\{i,j_0\}}$ because of the reachability of q and Lemma 3.2.2.
- $\alpha_0 \notin Int'(q_{j_0})$ by assumption.
- $\nexists \tilde{\alpha} \in Int'(q_i) \cap Int'(q_{j_0})$ with $|\tilde{\alpha}| = 2$ because $i, j_0 \in \tilde{K}$. It would not be possible to find a component in $comp(\tilde{\alpha})$ satisfying the condition required for interactions in $Int'(q_i)$ respectively $Int'(q_{j_0})$.

We conclude $q_{j_0} \in PS'^{0}_{j_0}(q_i, \alpha_0)$. We want to show $q_{j_0} \in PS'_{j_0}(q_i, \alpha_0)$. By way of contradiction assume $q_{j_0} \notin PS'_{j_0}(q_i, \alpha_0)$. There must be an index $l \in \mathbb{N}$ with $q_{j_0} \notin PS'^{l}_{j_0}(q_i, \alpha_0)$. Denote the smallest such index by l_0. We get $q_{j_0} \notin PS'^{l_0}_{j_0}(q_i, \alpha_0)$, but $q_{j_0} \in PS'^{m}_{j_0}(q_i, \alpha_0)$ for all $0 \leq m < l_0$. There must be $\alpha_1 \in Int'(q_{j_0})$ with:

$$(+) \quad \forall k \in comp(\alpha_1) \setminus \{j_0\} \text{ the set } PS'^{l_0-1}_{k}(q_{j_0}, \alpha_1) \text{ is empty.}$$

Because $j_0 \in \tilde{K}$ we know that there exists $j_1 \in \tilde{K} \cap comp(\alpha_1) \setminus \{j_0\}$ with $\alpha_1 \notin Int'(q_{j_1})$ and $need'(q_{j_1}) \cap comp(\alpha_1) \neq \emptyset$. Repeating the steps above we see that $q_{j_1} \in PS'^{0}_{j_1}(q_{j_0}, \alpha_1)$. We denote by l_1 the smallest index such that $q_{j_1} \notin PS'^{l_1}_{j_1}(q_{j_0}, \alpha_1)$. This index exists and it is smaller than l_0 because of $(+)$. Repeating this argument we obtain sequences

- $j_0, j_1, j_2, \ldots$ of components
- $\alpha_0, \alpha_1, \alpha_2, \ldots$ of interactions
- $l_0 > l_1 > l_2 > \ldots \geq 0$ of indices

with $q_{j_r} \in PS'^{0}_{j_r}\left(q_{j_{r-1}}, \alpha_r\right)$ but $q_{j_r} \notin PS'^{l_r}_{j_r}\left(q_{j_{r-1}}, \alpha_r\right)$ (we write $j_{-1} := i$). Because the sequence of the l_r is descending but bounded below by 0, after a finite number of steps there must be an index s with $l_s = 0$. For this index we have $q_{j_s} \in PS'^{0}_{j_s}\left(q_{j_{s-1}}, \alpha_s\right)$ but $q_{j_s} \notin PS'^{l_s}_{j_s}\left(q_{j_{s-1}}, \alpha_s\right) = PS'^{0}_{j_s}\left(q_{j_{s-1}}, \alpha_s\right)$ which is a contradiction.

We conclude $q_{j_0} \in PS'_{j_0}(q_i, \alpha_0)$. □

Corollary 3.8.4. *Let Sys be a tree-like interaction system. Let $q \in Q$ be a reachable deadlock. Let K' be given for q as in Corollary 3.8.1. Let $i \in K'$ and $\alpha \in Int(q_i)$.*

There exists $j \in comp(\alpha) \setminus \{i\} \cap K'$ with $q_j \in PS_j(q_i, \alpha)$.

The corollary requires q to be a deadlock state. In particular this means that no q_i is complete and therefore $need(q_i) \neq \emptyset$. Thus, the conditions for Corollary 3.8.1 are met and we may indeed choose K' for q as above.

Proof. The second defining property of K' implies $need(q_j) \cap comp(\alpha) \neq \emptyset$ for all $j \in comp(\alpha) \setminus \{i\}$. Because q is a deadlock state we know that there

is at least one component $j \in comp(\alpha)\setminus\{i\}$ with $j(\alpha) \not\subseteq \mathcal{en}(q_j)$. The first defining property of K' together with $i \in K'$ and $\alpha \in Int(q_i)$ implies $comp(\alpha) \subseteq K'$. We conclude $j \in K'$.

Setting $Int'(q_k) = Int(q_k)$ for all $k \in K$ and $q_k \in Q_k$ the conditions for Lemma 3.8.6 are met. We conclude that there is $j \in comp(\alpha) \setminus \{i\} \cap K'$ with $q_j \in PS'_j(q_i, \alpha) = PS_j(q_i, \alpha)$, the equality following from Lemma 3.8.5. □

We move on to the proof of Proposition 3.4.1.

Proposition 3.4.1. *Let Sys be a tree-like interaction system with strongly exclusive communication.*

If the following two conditions hold then Sys is deadlock-free:

1. $\forall i \forall q_i : q_i$ *complete* $\vee\, q_i^0 \notin BWS(q_i, need(q_i)) \vee \exists \alpha \in Int(q_i)$ *such that* $\forall j \in comp(\alpha)\setminus\{i\} : q_j^0 \notin BWS(PS_j(q_i,\alpha), \mathfrak{need}(PS_j(q_i,\alpha)))$

2. $\forall \tilde{\alpha} \in Int$ *with* $|comp(\tilde{\alpha})| \geq 2\ \exists i \in comp(\tilde{\alpha})$ *such that for all* $q_i \in Q_i$ *that are incomplete:*

$$i(\tilde{\alpha}) \not\subseteq \bigcap_{\alpha \in Int(q_i)} \bigcup_{j \in comp(\alpha)\setminus\{i\}} \mathcal{PA}(q_i, \alpha, j)$$

The conditions can be checked in time polynomial in the size of Sys.

Proof. Assume that the two conditions in the proposition hold but there is a reachable deadlock. There is a path $\sigma := q^0 \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_{l-1}} q^l \xrightarrow{\alpha_l} q$ such that no interaction is enabled in q. In particular this means that no q_i is complete and therefore $need(q_i) \neq \emptyset$ for all $i \in K$. We choose a minimal set K' for q according to Corollary 3.8.1, and we distinguish two cases:

1. For all $s \leq l$ we have $|\alpha_s| = 1$ or there is $k \notin K'$ with $k(\alpha_s) \neq \emptyset$. If $\alpha_s = \{a_i\}$ for a component $i \in K'$ we have $i(\alpha_s) \not\subseteq \bigcup_{j \in need(q_i)} \mathcal{comm}_i(j)$ because *Sys* has exclusive communication. In the second case Corollary 3.8.3 shows that $i(\alpha_s) \neq \emptyset$ implies $i(\alpha_s) \not\subseteq \bigcup_{j \in need(q_i)} \mathcal{comm}_i(j)$ for all $i \in K'$. Thus, for all $i \in K'$ the projection of σ to i yields a path in T_i starting in q_i^0 and ending in q_i that is only labeled with actions $a_i \in \mathcal{A}_i \setminus \bigcup_{j \in need(q_i)} \mathcal{comm}_i(j)$. We conclude $q_i^0 \in BWS(q_i, need(q_i))$ for all $i \in K'$.

Corollary 3.8.4 implies that for all $i \in K'$ and all $\alpha \in Int(q_i)$ there is $j \in comp(\alpha) \setminus \{i\} \cap K'$ with $q_j \in PS_j(q_i, \alpha)$. Using $j \in K'$, we get $q_j^0 \in BWS(q_j, need(q_j))$. Thus, for all $i \in K'$ and all $\alpha \in Int(q_i)$ there is $j \in comp(\alpha) \setminus \{i\}$ with $q_j^0 \in BWS(PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$. This is a contradiction to the first condition because we have already seen above that q_i is incomplete and $q_i^0 \in BWS(q_i, need(q_i))$.

2. Otherwise let $s_0 \leq l$ be the largest index with $comp(\alpha_{s_0}) \subseteq K'$ and $|\alpha_{s_0}| \geq 2$. For all $s_0 < s \leq l$ we have $|\alpha_s| = 1$ or there exists a component $h \notin K'$ with $h(\alpha_s) \neq \emptyset$. As above, we see that for all $i \in K'$ the projection to i of the segment of σ starting in q^{s_0+1} yields a path that is only labeled with ports $a_i \in \mathcal{A}_i \setminus \bigcup_{j \in need(q_i)} \mathcal{comm}_i(j)$. We get $q_i^{s_0+1} \in BWS(q_i, need(q_i))$ for all $i \in K'$.

   Choose $i \in comp(\alpha_{s_0}) \subseteq K'$ and fix $\alpha' \in Int(q_i)$. According to Corollary 3.8.4 there exists $j \in comp(\alpha') \setminus \{i\} \cap K'$ with $q_j \in PS_j(q_i, \alpha')$. Also fix j. We have $\alpha_{s_0} \downarrow_{\{i,j\}} \in Int \downarrow_{\{i,j\}}$. It is clear that $i(\alpha_{s_0}) \in \alpha_{s_0} \downarrow_{\{i,j\}}$ and that $(q_i^{s_0}, q_j^{s_0})$ is reachable in $Sys \downarrow_{\{i,j\}}$ (the latter follows from Lemma 3.2.2). Using $i, j \in K'$, we get $q_i^{s_0+1} \in BWS(q_i, need(q_i))$ and $q_j^{s_0+1} \in BWS(q_j, need(q_j))$. From $q_j \in PS_j(q_i, \alpha')$ we conclude $BWS(q_j, need(q_j)) \subseteq BWS(PS_j(q_i, \alpha'), \mathfrak{need}(PS_j(q_i, \alpha')))$. Finally, Corollary 3.8.2 shows $need(q_i) \cap comp(\alpha_{s_0}) \neq \emptyset$, and therefore $i(\alpha_{s_0}) \subseteq \bigcup_{k \in need(q_i)} \mathcal{comm}_i(k)$. Altogether this produces:

$$\begin{aligned} i(\alpha_{s_0}) &\subseteq \mathcal{EA}(q_i, need(q_i), PS_j(q_i, \alpha'), \mathfrak{need}(PS_j(q_i, \alpha'))) \\ &= \mathcal{PA}(q_i, \alpha', j) \subseteq \bigcup_{m \in comp(\alpha') \setminus \{i\}} \mathcal{PA}(q_i, \alpha', m) \end{aligned}$$

   Repeating this argument for all $\alpha \in Int(q_i)$, we get

$$i(\alpha_{s_0}) \in \bigcap_{\alpha \in Int(q_i)} \bigcup_{m \in comp(\alpha) \setminus \{i\}} \mathcal{PA}(q_i, \alpha, m).$$

   Analogously we get

$$k(\alpha_{s_0}) \in \bigcap_{\alpha \in Int(q_k)} \bigcup_{m \in comp(\alpha) \setminus \{k\}} \mathcal{PA}(q_k, \alpha, m)$$

   for all other $k \in comp(\alpha_{s_0})$. This is a contradiction to the second condition.

The assumption that q is a reachable deadlock state is wrong, and therefore *Sys* is deadlock-free.

It is clear that the conditions can be checked in time polynomial in the size of *Sys* because all parameters can be computed by analyzing only *Int*, the local transition systems, and the subsystems $Sys \downarrow_{\{i,j\}}$ where i and j are interacting components. □

3.8.4 Proofs for Section 3.4.2

Lemma 3.4.1. *Let Sys be a strongly tree-like interaction system. Let* $i \in K$, $q_i \in Q_i$ *incomplete, and* $j \in need(q_i)$. *Let* $q_j \in Q_j$.

We have $q_j \in PS_j(q_i)$ *if and only if for all* $\alpha \in Int(q_i)$ *with* $j(\alpha) \neq \emptyset$ *we have* $q_j \in PS_j(q_i, \alpha)$.

Proof. We make a general observation first. Let $k \in K$, $q_k \in Q_k$ incomplete, and $\alpha \in Int(q_k)$. We have $|comp(\alpha)| = 2$. There is a component other than k that participates in α because otherwise q_k would be complete. Lemma 3.3.1 shows that there cannot be more than two components in $comp(\alpha)$ because *Sys* is strongly tree-like.

We will now show $q_j \in PS_j^l(q_i)$ if and only if for all $\alpha \in Int(q_i)$ with $j(\alpha) \neq \emptyset$ we have $q_j \in PS_j^l(q_i, \alpha)$ by induction over l.

$\underline{l = 0:}$ We will show both directions.

$\underline{\Rightarrow:}$ Let $q_j \in PS_j^0(q_i)$ and $\alpha \in Int(q_i)$ with $j \in comp(\alpha)$. In order to prove $q_j \in PS_j^0(q_i, \alpha)$ we have to show the following:

- q_j is incomplete: This follows from the definition of $PS_j^0(q_i)$.
- $need(q_j) \cap comp(\alpha) \neq \emptyset$: It is clear that $i \in comp(\alpha)$. Furthermore, we have $i \in need(q_j)$ by definition of $PS_j^0(q_i)$.
- (q_i, q_j) is reachable in $Sys \downarrow_{\{i,j\}}$: This follows from the definition of $PS_j^0(q_i)$.
- $j(\alpha) \not\subseteq \mathcal{en}(q_j)$: Assume $j(\alpha) \subseteq \mathcal{en}(q_j)$. We get $\alpha \in Int(q_i) \cap Int(q_j)$. This is not possible according to the requirement $Int(q_i) \cap Int(q_j) = \emptyset$ stated in the definition of $PS_j^0(q_i)$.

- $\nexists \tilde{\alpha} \in \mathit{Int}(q_i) \cap \mathit{Int}(q_j)$ with $|\tilde{\alpha}| = 2$: This follows from the requirement $\mathit{Int}(q_i) \cap \mathit{Int}(q_j) = \emptyset$ stated in the definition of $PS_j^0(q_i)$.

We conclude $q_j \in PS_j^0(q_i, \alpha)$.

$\Leftarrow$: Assume that for all $\alpha \in \mathit{Int}(q_i)$ with $j(\alpha) \neq \emptyset$ we have $q_j \in PS_j^0(q_i, \alpha)$. Fix $\alpha \in \mathit{Int}(q_i)$ with $j(\alpha) \neq \emptyset$. In order to prove $q_j \in PS_j^0(q_i)$ we have to show the following:

- q_j incomplete: This follows from the definition of $PS_j^0(q_i, \alpha)$.
- $i \in \mathit{need}(q_j)$: From the definition of $PS_j^0(q_i, \alpha)$ we know that $\mathit{need}(q_j) \cap \mathit{comp}(\alpha) \neq \emptyset$. The observation above shows $\mathit{comp}(\alpha) = \{i, j\}$. By definition we have $j \notin \mathit{need}(q_j)$. Thus, the only component that can be contained in $\mathit{need}(q_j) \cap \mathit{comp}(\alpha)$ is i. We conclude $i \in \mathit{need}(q_j)$.
- (q_i, q_j) is reachable in $\mathit{Sys} \downarrow_{\{i,j\}}$: This follows from the definition of $PS_j^0(q_i, \alpha)$.
- $\mathit{Int}(q_i) \cap \mathit{Int}(q_j) = \emptyset$: Assume there is $\alpha \in \mathit{Int}(q_i) \cap \mathit{Int}(q_j)$. The observation above shows $|\mathit{comp}(\alpha)| = 2$. This is a contradiction to the last condition in the definition of $PS_j^0(q_i, \alpha)$.

We conclude $q_j \in PS_j^0(q_i)$.

$l \Rightarrow l+1$: Again we will show both directions.

$\Rightarrow$: Let $q_j \in PS_j^{l+1}(q_i)$ and $\alpha \in \mathit{Int}(q_i)$ with $j \in \mathit{comp}(\alpha)$. In order to prove $q_j \in PS_j^{l+1}(q_i, \alpha)$ we have to show the following:

- $q_j \in PS_j^l(q_i, \alpha)$: By definition of $PS_j^{l+1}(q_i)$ we have $q_j \in PS_j^l(q_i)$. The induction hypothesis implies $q_j \in PS_j^l(q_i, \alpha)$.
- $\forall \beta \in \mathit{Int}(q_j)\, \exists k \in \mathit{comp}(\beta) \setminus \{j\}$ with $PS_k^l(q_j, \beta) \neq \emptyset$: Let $\beta \in \mathit{Int}(q_j)$. Because $q_j \in PS_j^{l+1}(q_i)$ we know that q_j is incomplete. According to the observation above this implies $|\mathit{comp}(\beta)| = 2$. We write $\mathit{comp}(\beta) = \{j, k\}$. We get $k \in \mathit{need}(q_j)$. The definition of $PS_j^{l+1}(q_i)$ implies $PS_k^l(q_j) \neq \emptyset$. The induction hypothesis applied with q_j, k, and β implies $PS_k^l(q_j, \beta) \neq \emptyset$.

We conclude $q_j \in PS_j^{l+1}(q_i, \alpha)$.

$\underline{\Leftarrow}$: Assume that for all $\alpha \in Int(q_i)$ with $j(\alpha) \neq \emptyset$ we have $q_j \in PS_j^{l+1}(q_i, \alpha)$. In order to prove $q_j \in PS_j^{l+1}(q_i)$ we have to show the following:

- $q_j \in PS_j^l(q_i)$: By definition of $PS_j^{l+1}(q_i, \alpha)$ we have $q_j \in PS_j^l(q_i, \alpha)$. The induction hypothesis implies $q_j \in PS_j^l(q_i)$.
- $\forall k \in need(q_j)$ we have $PS_k^l(q_j) \neq \emptyset$: Let $k \in need(q_j)$. There is $\beta \in Int(q_j)$ with $comp(\beta) = \{j, k\}$. Because $q_j \in PS_j^{l+1}(q_i, \alpha)$ there is $k' \in comp(\beta) \setminus \{j\}$ with $PS_{k'}^l(q_j, \beta) \neq \emptyset$. The only possible choice for k' is k. This argument can be repeated for all $\beta \in Int(q_j)$ with $k \in comp(\beta)$. Therefore, the induction hypothesis applied with q_j, k, and any such β implies $PS_k^l(q_j) \neq \emptyset$.

We conclude $q_j \in PS_j^{l+1}(q_i)$.

For all $l \in \mathbb{N}$ we have $q_j \in PS_j^l(q_i)$ if and only if for all $\alpha \in Int(q_i)$ with $j(\alpha) \neq \emptyset$ we have $q_j \in PS_j^l(q_i, \alpha)$. This implies that $q_j \in PS_j(q_i)$ if and only if for all $\alpha \in Int(q_i)$ with $j(\alpha) \neq \emptyset$ we have $q_j \in PS_j(q_i, \alpha)$. □

The following lemma will be needed for the proof of Corollary 3.4.2.

Lemma 3.8.7. *Let Sys be a strongly tree-like interaction system that has strongly exclusive communication. Let $i \in K$ and $q_i \in Q_i$ incomplete.*

1. *Let $\alpha \in Int(q_i)$ with $comp(\alpha) = \{i, j\}$. We have:*

$$BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i))) =$$

$$BWS(PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$$

2. *We have:*

$$\bigcap_{\alpha \in Int(q_i)} \bigcup_{j \in comp(\alpha) \setminus \{i\}} \mathcal{PA}(q_i, \alpha, j) = \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k)$$

Proof. We prove the two statements separately.

1. Lemma 3.4.1 shows $PS_j(q_i) = PS_j(q_i, \alpha)$. Therefore $\mathfrak{need}(PS_j(q_i)) = \mathfrak{need}(PS_j(q_i, \alpha))$, and we conclude $BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i))) = BWS(PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$.

2. Let $\alpha \in Int(q_i)$. The observation made in the beginning of the previous proof shows $|\alpha| = 2$. We write $comp(\alpha) = \{i, j_\alpha\}$. We get:

$$\begin{aligned}\bigcap_{\alpha \in Int(q_i)} \bigcup_{j \in comp(\alpha) \setminus \{i\}} \mathcal{PA}(q_i, \alpha, j) &= \bigcap_{\alpha \in Int(q_i)} \mathcal{PA}(q_i, \alpha, j_\alpha) \\ &= \bigcap_{\alpha \in Int(q_i)} \mathcal{PA}(q_i, j_\alpha) \\ &= \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k)\end{aligned}$$

The second equality follows from Corollary 3.4.1. □

Corollary 3.4.2. *Let Sys be a strongly tree-like interaction system with strongly exclusive communication*

If the following two conditions hold then Sys is deadlock-free:

1. $\forall i \forall q_i : q_i$ *complete* $\vee\ q_i^0 \notin BWS(q_i, need(q_i)) \vee$
   $\exists j \in need(q_i)$ *with* $q_j^0 \notin BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$

2. $\forall \tilde{\alpha} \in Int$ *with* $|comp(\tilde{\alpha})| = 2$ $\exists i \in comp(\tilde{\alpha})$ *such that for all* $q_i \in Q_i$ *that are incomplete:*

$$i(\tilde{\alpha}) \not\subseteq \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k)$$

The conditions can be checked in time polynomial in the size of Sys.

Proof. We show that for strongly tree-like interaction systems the conditions above are equivalent to the corresponding conditions of Proposition 3.4.1:

1. The first two terms in the disjunction are the same as the corresponding terms in Proposition 3.4.1. With regard to the third term we write $comp(\alpha) = \{i, j_\alpha\}$ for $\alpha \in Int(q_i)$ as in the previous proof. We get:

   $\exists \alpha \in Int(q_i) \forall j \in comp(\alpha) \setminus \{i\}$:

$$q_j^0 \notin BWS(PS_j(q_i,\alpha), \mathfrak{need}(PS_j(q_i,\alpha))) \Leftrightarrow$$
$$\exists \alpha \in Int(q_i) : q_{j_\alpha}^0 \notin BWS(PS_{j_\alpha}(q_i,\alpha), \mathfrak{need}(PS_{j_\alpha}(q_i,\alpha))) \Leftrightarrow$$
$$\exists \alpha \in Int(q_i) : q_{j_\alpha}^0 \notin BWS(PS_{j_\alpha}(q_i), \mathfrak{need}(PS_{j_\alpha}(q_i))) \Leftrightarrow$$
$$\exists j \in need(q_i) : q_j^0 \notin BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$$

The second equivalence follows from Lemma 3.8.7.1. The third term in the first condition of the corollary is therefore equivalent to the corresponding term in the first condition of Proposition 3.4.1. Thus, for strongly tree-like interaction systems these conditions are equivalent.

2. From Lemma 3.3.1 we know that all interactions involve two components at most. Thus, the universal quantifier in the second condition of Proposition 3.4.1 referring to all $\tilde{\alpha}$ with $|comp(\tilde{\alpha})| \geq 2$ in fact refers to all $\tilde{\alpha}$ with $|comp(\tilde{\alpha})| = 2$. Lemma 3.8.7.2 directly shows that for strongly tree-like interaction systems the condition stated is equivalent to the corresponding condition in Proposition 3.4.1.

Now assume that the two conditions above are satisfied. Using Lemma 3.3.2 and the equivalences derived above, we conclude that *Sys* is a tree-like interaction system with strongly exclusive communication for which the conditions in Proposition 3.4.1 are satisfied. The proposition shows that *Sys* is deadlock-free. □

As mentioned in Section 3.4.2 we proved a statement which is more general than the corollary's statement. The corollary states that *Sys* is deadlock-free if the two conditions are satisfied. We showed that for strongly tree-like interaction systems the two conditions are equivalent to the corresponding conditions in Proposition 3.4.1. It would have been possible to give a direct proof for Corollary 3.4.2: Using a subset K' satisfying slightly simplified conditions compared to those in Lemma 3.8.2 the proof would have followed the lines of the proof of Proposition 3.4.1. It should be noted, that such a direct argument would have been more simple than the proof given above. On the other hand, such a simple proof would not have guaranteed that there are no strongly tree-like interaction system for which Proposition 3.4.1 proves deadlock-freedom while Corollary 3.4.2 fails to do so. This is why we presented the more involved proof above.

Before proving Corollary 3.4.3 we need some lemmas.

Lemma 3.8.8. *Let Sys be an interaction system with strongly exclusive communication. Let $i \neq j \in K$ and $q_i, q_i' \in Q_i$ and $q_j, q_j' \in Q_j$.*

We have $q_i' \in BWS(q_i, \{j\})$ and $q_j' \in BWS(q_j, \{i\})$ if and only if in $Sys\downarrow_{\{i,j\}}$ there exists a sequence

$$\sigma = \left(q_i', q_j'\right) \xrightarrow{\alpha^0} \ldots \xrightarrow{\alpha^l} (q_i, q_j)$$

with $|\alpha^s| = 1$ for all $0 \leq s \leq l$.

Proof. We will show both directions.

$\underline{\Rightarrow}$: Assume $q_i' \in BWS(q_i, \{j\})$ and $q_j' \in BWS(q_j, \{i\})$. There are paths $q_i' \xrightarrow{a_i^0} \ldots \xrightarrow{a_i^{l_i}} q_i$ in T_i with $a_i^s \notin \mathit{comm}_i(j)$ for all $0 \leq s \leq l_i$ respectively $q_j' \xrightarrow{b_j^0} \ldots \xrightarrow{b_j^{l_j}} q_j$ in T_j with $b_j^s \notin \mathit{comm}_j(i)$ for all $0 \leq s \leq l_j$. Consider a_i^s and $\alpha \in Int$ with $a_i^s \in \alpha$. We have $j \notin comp(\alpha)$ because $a_i^s \notin \mathit{comm}_i(j)$. We get $\alpha\downarrow_{\{i,j\}} = \{a_i^s\} \in Int\downarrow_{\{i,j\}}$. Analogously we have $\{b_j^s\} \in Int\downarrow_{\{i,j\}}$ for all b_j^s. Thus, in $Sys\downarrow_{\{i,j\}}$ there is a sequence

$$\sigma = \left(q_i', q_j'\right) \xrightarrow{\{a_i^0\}} \ldots \xrightarrow{\{a_i^{l_i}\}} \left(q_i, q_j'\right) \xrightarrow{\{b_j^0\}} \ldots \xrightarrow{\{b_j^{l_j}\}} (q_i, q_j)$$

as required in the lemma.

$\underline{\Leftarrow}$: Assume that in $Sys\downarrow_{\{i,j\}}$ there exists a sequence

$$\sigma = \left(q_i', q_j'\right) \xrightarrow{\alpha^0} \ldots \xrightarrow{\alpha^l} (q_i, q_j)$$

with $|\alpha^s| = 1$ for all $0 \leq s \leq l$. Consider an arbitrary α^s. Assume $comp(\alpha^s) = \{i\}$, and write $\alpha^s = \{a_i\}$. There exists $\alpha \in Int$ with $\alpha^s = \alpha\downarrow_{\{i,j\}}$. Clearly, j does not participate in α. Because *Sys* has strongly exclusive communication there is no other interaction $\tilde{\alpha} \in Int$ with $a_i \in \tilde{\alpha}$. We conclude $a_i \notin \mathit{comm}_i(j)$. Projecting σ to i we obtain a path $q_i' \xrightarrow{a_i^0} \ldots \xrightarrow{a_i^{l_i}} q_i$ in T_i that is only labeled with ports that are not in $\mathit{comm}_i(j)$. This implies $q_i' \in BWS(q_i, \{j\})$. The case $comp(\alpha^s) = \{j\}$ is treated analogously, and we infer $q_j' \in BWS(q_j, \{i\})$.

□

Lemma 3.8.9. *Let Sys be a strongly tree-like interaction system such that* $|need(q_i)| \leq 1$ *for all* $i \in K$ *and* $q_i \in Q_i$*. Consider an edge* $\{i,j\} \in E^*$ *and local states* $q_i \in Q_i$ *and* $q_j \in Q_j$*.*

If (q_i, q_j) *is a deadlock state of Sys* $\downarrow_{\{i,j\}}$ *then* $need\,(q_i) = \{j\}$ *and* $need(q_j) = \{i\}$*.*

Proof. Assume that the statement is not true. Because $|need(q_k)| \leq 1$ for all $k \in K$ and all local states q_k we therefore have $j \notin need\,(q_i)$ or $i \notin need(q_j)$. Without loss of generality $j \notin need\,(q_i)$. Choose $\alpha \in Int(q_i)$. Because $j \notin need(q_i)$ we have $j \notin comp\,(\alpha)$. Therefore $\alpha \downarrow_{\{i,j\}} = \{a_i\} \in Int \downarrow_{\{i,j\}}$. This interaction is enabled in (q_i, q_j) which is a contradiction since (q_i, q_j) is a deadlock in $Sys \downarrow_{\{i,j\}}$. We get $need\,(q_i) = \{j\}$ and $need(q_j) = \{i\}$. □

Lemma 3.8.10. *Let Sys be a strongly tree-like interaction system such that* $|need(q_i)| \leq 1$ *for all* $i \in K$ *and* $q_i \in Q_i$*. Consider an edge* $\{i,j\} \in E^*$ *and local states* $q_i \in Q_i$ *and* $q_j \in Q_j$*.*

We have:

$$q_j \in PS_j\,(q_i) \Leftrightarrow (q_i, q_j) \text{ is a reachable deadlock in } Sys \downarrow_{\{i,j\}}$$

Proof. We will show the following two equivalences which together yield the statement of the lemma:

1. $q_j \in PS_j\,(q_i) \Leftrightarrow q_j \in PS_j^0\,(q_i)$
2. $q_j \in PS_j^0\,(q_i) \Leftrightarrow (q_i, q_j)$ is a reachable deadlock in $Sys \downarrow_{\{i,j\}}$

Consider the first equivalence. Because $PS_j\,(q_i) \subseteq PS_j^0\,(q_i)$ it suffices to show that $q_j \in PS_j^0\,(q_i)$ implies $q_j \in PS_j\,(q_i)$. We first show

$$q_j \in PS_j^l\,(q_i) \Rightarrow q_i \in PS_i^l\,(q_j) \tag{3.8.1}$$

for all $l \in \mathbb{N}$ by induction over l.

$\underline{l = 0:}$ Let $q_j \in PS_j^0\,(q_i)$. It is clear that $q_i \in PS_i^0\,(q_j)$ because the conditions stated in Definition 3.4.3 for q_j to be contained in $PS_j^0\,(q_i)$ are symmetric in i and j.

$\underline{l \Rightarrow l+1:}$ Let $q_j \in PS_j^{l+1}(q_i)$. Then we also have $q_j \in PS_j^m(q_i)$ for all $0 \leq m \leq l$. Together with the induction hypothesis this implies $q_i \in PS_i^m(q_j)$ for all $0 \leq m \leq l$. In particular, $q_i \in PS_i^0(q_j)$ implies $j \in need(q_i)$. Because $need(q_i)$ contains at most one element we get $need(q_i) = \{j\}$. We want to prove $q_i \in PS_i^{l+1}(q_j)$. The above implies $q_i \in PS_i^l(q_j)$. We still have to show that for all $k \in need(q_i) = \{j\}$ we have $PS_k^l(q_i) \neq \emptyset$. We know $q_j \in PS_j^l(q_i)$, and we conclude $q_i \in PS_i^{l+1}(q_j)$.

Using condition 3.8.1, we now directly show that for all $l \in \mathbb{N}$ we have:

$$q_j \in PS_j^l(q_i) \Rightarrow q_j \in PS_j^{l+1}(q_i) \tag{3.8.2}$$

Let $q_j \in PS_j^l(q_i)$. In order to conclude $q_j \in PS_j^{l+1}(q_i)$ it suffices to show that for all $k \in need(q_j)$ we have $PS_k^l(q_j) \neq \emptyset$. From $q_j \in PS_j^l(q_i)$ we infer $q_j \in PS_j^0(q_i)$. As above, this implies $need(q_j) = \{i\}$. Thus, we only have to show $PS_i^l(q_j) \neq \emptyset$. This follows from $q_j \in PS_j^l(q_i)$ and condition 3.8.1. We get $q_j \in PS_j^{l+1}(q_i)$.

Now, assume $q_j \in PS_j^0(q_i)$. Then, condition 3.8.2 implies $q_j \in PS_j^l(q_i)$ for all $l \in \mathbb{N}$. Therefore, we have $q_j \in PS_j(q_i)$. This concludes the proof of the first equivalence above.

Next, we show that $q_j \in PS_j^0(q_i)$ if and only if (q_i, q_j) is a reachable deadlock in $Sys\downarrow_{\{i,j\}}$.

$\underline{\Rightarrow:}$ Let $q_j \in PS_j^0(q_i)$. We want to show that (q_i, q_j) is a reachable deadlock in $Sys\downarrow_{\{i,j\}}$. The conditions stated in Definition 3.4.3 for q_j to be contained $PS_j^0(q_i)$ imply that (q_i, q_j) is reachable in $Sys\downarrow_{\{i,j\}}$, that both q_i and q_j are incomplete, and that $j \in need(q_i)$ and $i \in need(q_j)$. Because these sets contain one component at most we get $need(q_i) = \{j\}$ and $need(q_j) = \{i\}$.

Now assume that some $\alpha \in Int\downarrow_{\{i,j\}}$ is enabled in (q_i, q_j). Let $\alpha' \in Int$ be an interaction with $\alpha'\downarrow_{\{i,j\}} = \alpha$. According to Lemma 3.3.1 we have $|\alpha'| \leq 2$. We consider the following cases:

- $comp(\alpha') = \{i\}$: Then q_i would be complete which is not possible. The case $comp(\alpha') = \{j\}$ is treated analogously.

- $comp(\alpha') = \{i, k\}$ with $k \neq j$: We get $k \in need(q_i)$. This is a contradiction to $need(q_i) = \{j\}$.

  The case $comp(\alpha') = \{j, k\}$ with $k \neq i$ is treated analogously.

- $comp(\alpha') = \{i, j\}$: We get $\alpha' = \alpha' \downarrow_{\{i,j\}} = \alpha$. Because α is enabled in (q_i, q_j) this implies $\alpha' \in Int(q_i) \cap Int(q_j)$ which is not possible because $q_j \in PS_j^0(q_i)$.

Therefore, no $\alpha \in Int \downarrow_{\{i,j\}}$ is enabled in (q_i, q_j). We conclude that (q_i, q_j) is a reachable deadlock of $Sys \downarrow_{\{i,j\}}$.

$\Leftarrow$: Let (q_i, q_j) be a reachable deadlock in $Sys \downarrow_{\{i,j\}}$. We want to show $q_j \in PS_j^0(q_i)$. It is clear that q_i and q_j are incomplete because otherwise (q_i, q_j) would not be a deadlock. Lemma 3.8.9 shows $need(q_i) = \{j\}$ and $need(q_j) = \{i\}$. Therefore $PS_j^0(q_i)$ is well defined, and the first two conditions for q_j to be contained in this set are satisfied. By assumption (q_i, q_j) is reachable. There cannot be any interaction $\tilde{\alpha} \in Int(q_i) \cap Int(q_j)$ because $\tilde{\alpha} \downarrow_{\{i,j\}}$ would be enabled in (q_i, q_j) in this case, and (q_i, q_j) would not be a deadlock of $Sys \downarrow_{\{i,j\}}$. We conclude $q_j \in PS_j^0(q_i)$.

Combining the two equivalences, we see that $q_j \in PS_j(q_i)$ if and only if (q_i, q_j) is a reachable deadlock in $Sys \downarrow_{\{i,j\}}$. □

Combining the results above we obtain the following lemma:

Lemma 3.8.11. *Let Sys be a strongly tree-like interaction system that has strongly exclusive communication such that $|need(q_i)| \leq 1$ for all $i \in K$ and $q_i \in Q_i$. Let $q_i, q_i' \in Q_i$ and $q_j' \in Q_j$.*

There exists $q_j \in Q_j$ and a path $\sigma = (q_i', q_j') \xrightarrow{\alpha^0} \ldots \xrightarrow{\alpha^l} (q_i, q_j)$ in $Sys \downarrow_{\{i,j\}}$ such that $|\alpha^s| = 1$ for all $0 \leq s \leq l$ and such that (q_i, q_j) is a reachable deadlock of $Sys \downarrow_{\{i,j\}}$ if and only if $q_i' \in BWS(q_i, need(q_i))$ and $q_j' \in BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$.

Proof. We will show both directions:

$\Leftarrow$: Let $q_i' \in BWS(q_i, need(q_i))$ and $q_j' \in BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$. Thus, there is a local state $q_j \in PS_j(q_i)$ with $q_j' \in BWS(q_j, need(q_j))$.

Lemma 3.8.10 shows that (q_i, q_j) is a reachable deadlock of $Sys \downarrow_{\{i,j\}}$. Lemma 3.8.9 shows $need(q_j) = \{i\}$ and $need(q_i) = \{j\}$. This means that $q_i' \in BWS(q_i, \{j\})$ and $q_j' \in BWS(q_j, \{i\})$. Lemma 3.8.8 yields the existence of a path $\sigma = (q_i', q_j') \xrightarrow{\alpha^0} \ldots \xrightarrow{\alpha^l} (q_i, q_j)$ in $Sys \downarrow_{\{i,j\}}$ such that $|\alpha^s| = 1$ for all $0 \leq s \leq l$.

$\underline{\Rightarrow}$: Consider a path $\sigma = (q_i', q_j') \xrightarrow{\alpha^0} \ldots \xrightarrow{\alpha^l} (q_i, q_j)$ such that $|\alpha^s| = 1$ for all $0 \leq s \leq l$ and (q_i, q_j) is a reachable deadlock state. Lemma 3.8.10 shows that $q_j \in PS_j(q_i)$. Using Lemma 3.8.8, we get $q_i' \in BWS(q_i, \{j\})$ and $q_j' \in BWS(q_j, \{i\})$. Because (q_i, q_j) is a deadlock state, Lemma 3.8.9 implies $need(q_j) = \{i\}$ and $need(q_i) = \{j\}$. We conclude $q_i' \in BWS(q_i, need(q_i))$ on the one hand and $q_j' \in BWS(q_j, need(q_j)) \subseteq BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$ on the other.

□

Corollary 3.4.3. *[38] Let Sys be a strongly tree-like interaction system with strongly exclusive communication and $|need(q_i)| \leq 1$ for all $i \in K$ and for all $q_i \in Q_i$.*

If $Sys \downarrow_{\{i,j\}}$ is deadlock-free for all $\{i,j\} \in E^*$ then Sys is deadlock-free.

Proof. We show that under the additional requirement $|need(q_i)| \leq 1$ for all $i \in K$ and $q_i \in Q_i$ the two conditions in Corollary 3.4.2 hold if and only if $Sys \downarrow_{\{i,j\}}$ is deadlock-free for all $\{i,j\} \in E^*$.

$\underline{\Rightarrow}$: Assume that the conditions of Corollary 3.4.2 are satisfied but there is an edge $\{i,j\} \in E^*$ such that $Sys \downarrow_{\{i,j\}}$ contains a deadlock. Let (q_i, q_j) be a reachable deadlock state in $Sys \downarrow_{\{i,j\}}$. Consider a path

$$(q_i^0, q_j^0) \xrightarrow{\alpha^0} \ldots \xrightarrow{\alpha^l} (q_i, q_j).$$

Lemma 3.8.9 shows $need(q_i) = \{j\}$ and $need(q_j) = \{i\}$. We distinguish two cases:

1. $|\alpha^s| = 1$ for all s: Lemma 3.8.11 shows $q_i^0 \in BWS(q_i, need(q_i))$ and $q_j^0 \in BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$. Then q_i violates the

first condition of Corollary 3.4.2 because it is clear that q_i is incomplete and because the only component in $need(q_i)$ is j.

2. There is s with $|\alpha^s| = 2$: Let s_0 denote the largest such index. We write $\alpha^{s_0} = \{a_i^{s_0}, a_j^{s_0}\}$. For all $s_0 + 1 \leq r \leq l$ we get $|\alpha^r| = 1$. Lemma 3.8.11 shows $q_i^{s_0+1} \in BWS(q_i, need(q_i))$ and $q_j^{s_0+1} \in BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$. In $Sys\downarrow_{\{i,j\}}$ there is a transition $(q_i^{s_0}, q_j^{s_0}) \stackrel{\alpha^{s_0}}{\rightarrow} (q_i^{s_0+1}, q_j^{s_0+1})$ and $(q_i^{s_0}, q_j^{s_0})$ is reachable. It is also clear that $a_i^{s_0} \in \mathcal{comm}_i(j) = \bigcup_{k \in need(q_i)} \mathcal{comm}_i(k)$. Combining these facts we obtain $a_i^{s_0} \in \mathcal{PA}(q_i, j) = \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k)$. Analogously we have $a_j^{s_0} \in \bigcap_{k \in need(q_j)} \mathcal{PA}(q_j, k)$. Then α^{s_0} violates the second condition of Corollary 3.4.2.

We conclude that $Sys\downarrow_{\{i,j\}}$ is deadlock-free for all $\{i, j\} \in E^*$.

$\Leftarrow$: Assume that $Sys\downarrow_{\{i,j\}}$ is deadlock-free for all $\{i, j\} \in E^*$ but one of the conditions of Corollary 3.4.2 is violated:

1. The first condition is violated. There exist $i \in K$ and $q_i \in Q_i$ such that q_i is incomplete, $q_i^0 \in BWS(q_i, need(q_i))$, and for all $j \in need(q_i)$ we have $q_j^0 \in BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$. The last term above quantifies over all $j \in need(q_i)$. Since there can be at most one such j and because q_i is incomplete there is exactly one such j. For j we have $q_j^0 \in BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$. Lemma 3.8.11 shows that there is a path $(q_i^0, q_j^0) \stackrel{\alpha^0}{\rightarrow} \ldots \stackrel{\alpha^l}{\rightarrow} (q_i, q_j)$ in $Sys\downarrow_{\{i,j\}}$ such that (q_i, q_j) is a reachable deadlock. This is a contradiction because $Sys\downarrow_{\{i,j\}}$ is deadlock-free.

2. The second condition is violated. There exists $\tilde{\alpha} \in Int$ with $|\tilde{\alpha}| = 2$ such that $\forall i \in comp(\tilde{\alpha})$ there is $q_i \in Q_i$ incomplete with $i(\tilde{\alpha}) \subseteq \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k)$. Fix $\tilde{\alpha}$, $i \in comp(\tilde{\alpha})$, and q_i incomplete with $i(\tilde{\alpha}) \subseteq \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k)$. Because q_i is incomplete we have $need(q_i) \neq \emptyset$. Therefore $need(q_i) = \{j\}$ for some component j. Writing $i(\tilde{\alpha}) = \{a_i\}$, we get $a_i \in \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k) = \mathcal{PA}(q_i, j)$. Thus, there exists a reachable state (q_i', q_j') in $Sys\downarrow_{\{i,j\}}$ that can be entered by an interaction involving a_i such that $q_i' \in BWS(q_i, need(q_i))$ and $q_j' \in BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$.

Lemma 3.8.11 shows that there is a path $(q_i', q_j') \xrightarrow{\alpha^0} \ldots \xrightarrow{\alpha^l} (q_i, q_j)$ in $Sys \downarrow_{\{i,j\}}$ such that (q_i, q_j) is a reachable deadlock state. This is a contradiction because $Sys \downarrow_{\{i,j\}}$ is deadlock-free.

Now assume that Sys is a strongly tree-like interaction system with $|need(q_i)| \leq 1$ for all $i \in K$ and $q_i \in Q_i$ such that $Sys \downarrow_{\{i,j\}}$ is deadlock-free for all $\{i, j\} \in E^*$. The equivalences derived above show that Sys is a strongly tree-like interaction system for which the conditions of Corollary 3.4.2 are satisfied. Therefore Sys is deadlock-free.

□

The remark made after the proof of Corollary 3.4.2 is true in particular for the proofs above. A direct proof of Corollary 3.4.3 would have been much more simple than the involved proofs of the equivalences above (Lemma 3.8.10). It would not have been as general, though. We would not have been able to see that the two corollaries are equally general for strongly tree-like interaction systems satisfying the additional requirement above.

3.8.5 Proofs for Section 3.5.1

We need two lemmas for the proof of Corollary 3.5.1.

Lemma 3.8.12. *Let Sys be a strongly tree-like interaction system and let D be a local deadlock in q.*

For all $i \in D$ and all $\alpha \in Int(q_i)$ we have $comp(\alpha) \subseteq D$.

Proof. Choose $i \in D$ and $\alpha \in Int(q_i)$. Because D is a local deadlock there exists $j \in comp(\alpha) \cap D$ with $j(\alpha) \not\subseteq \mathcal{en}(q_j)$. Thus, $j \in D$. By assumption we have $i \in D$. It is clear that $i \neq j$. According to Lemma 3.3.1 there cannot be more than two components in $comp(\alpha)$. Therefore $comp(\alpha) \subseteq D$. □

Lemma 3.8.13. *Let Sys be a strongly tree-like interaction system and let D be a local deadlock in q.*

There exists a non empty subset $D' \subseteq D$ such that for all $i \in D'$ and all $\alpha' \in Int(q_i)$ the following two conditions hold:

1. *$\forall \alpha \in Int$ with $|comp(\alpha') \cap comp(\alpha)| \geq 2$ we have $comp(\alpha) \subseteq D'$*

2. $\forall j \in comp(\alpha') \setminus \{i\} : need(q_j) \cap comp(\alpha') \neq \emptyset$

Proof. Let $i \in D$, $\alpha' \in Int(q_i)$, and $\alpha \in Int$ with $|comp(\alpha') \cap comp(\alpha)| \geq 2$. Lemma 3.3.1 states $|comp(\alpha')| \leq 2$ and $|comp(\alpha)| \leq 2$. We conclude $|comp(\alpha') \cap comp(\alpha)| = 2$ and therefore $comp(\alpha) = comp(\alpha')$. Lemma 3.8.12 implies $comp(\alpha') \subseteq D$, and this means $comp(\alpha) \subseteq D$.

Setting $Int'(q_i) := Int(q_i)$ for all $i \in K$, this means that D satisfies the condition required for $\tilde{K}$ in Lemma 3.8.2. It is clear that $need(q_i) \neq \emptyset$ for all $i \in D$ because D is a local deadlock. The lemma shows that there is a subset D' having the desired properties. □

Strictly speaking we are not allowed to apply Lemma 3.8.2 because it requires $need(q_i) \neq \emptyset$ for *all* $i \in K$. We can only guarantee that $need(q_i) \neq \emptyset$ for all $i \in D$. According to the remark made after the proof of Lemma 3.8.2 this suffices to apply the lemma, though, because in the proof of Lemma 3.8.2 the requirement is only needed for the components in $\tilde{K}$.

Corollary 3.5.1. *Let Sys be a strongly tree-like interaction system with strongly exclusive communication.*

If the following two conditions hold then Sys does not contain any local deadlock:

1. $\forall i \; \forall q_i : q_i \text{ complete} \vee q_i^0 \notin BWS(q_i, need(q_i)) \vee$
   $\exists j \in need(q_i) \text{ with } q_j^0 \notin BWS(PS_j(q_i), \mathfrak{need}(PS_j(q_i)))$

2. $\forall \tilde{\alpha} \in Int$ with $|comp(\tilde{\alpha})| = 2 \; \exists i \in comp(\tilde{\alpha})$ such that for all $q_i \in Q_i$ that are incomplete:

$$i(\tilde{\alpha}) \not\subseteq \bigcap_{k \in need(q_i)} \mathcal{PA}(q_i, k)$$

The conditions can be checked in time polynomial in the size of Sys.

Proof. Assume that there exists a reachable local deadlock D in q. Choose a minimal subset $D' \subseteq D$ of components according to Lemma 3.8.13. For all $i \in D'$ all $\alpha \in Int(q_i)$ are blocked by a component $j \in comp(\alpha) \cap D$ because D is a local deadlock. On the other hand, $i \in D'$ and $\alpha \in Int(q_i)$

imply $comp(\alpha) \subseteq D'$ using the first defining property of D'. Thus, for all $i \in D'$ and $\alpha \in Int(q_i)$ the component j is actually contained in D'.

We have seen in the proof of Corollary 3.4.2 that for strongly tree-like systems the two conditions in the corollary above are equivalent to the conditions in Proposition 3.4.1. The proof of Proposition 3.4.1 showed that the existence of a minimal subset D' for q where

1. the two conditions in Corollary 3.8.1 (respectively Lemma 3.8.13) hold
2. for all $i \in D'$ all $\alpha \in Int(q_i)$ are blocked by a component in D'

yields a contradiction to the conditions in Proposition 3.4.1.

We conclude there is no reachable local deadlock D in *Sys*. □

3.8.6 Proofs for Section 3.5.2

The following lemma combines Lemmas 3.8.2 and 3.8.6 and adapts them to the new situation.

Lemma 3.8.14. *Let Sys be a tree-like interaction system. Let D be a minimal local deadlock in q.*

1. *For all $i \in D$, $\alpha \in Int(q_i)$, and $j \in D \cap comp(\alpha) \setminus \{i\}$ we have:*

$$j(\alpha) \not\subseteq \mathit{en}(q_j) \Rightarrow need(q_j) \cap comp(\alpha) \neq \emptyset$$

2. *If q is reachable then for all $i \in D$ and $\alpha \in Int(q_i)$ there exists $j \in D \cap comp(\alpha)$ with $q_j \in PS_j(q_i, \alpha)$.*

Proof. We treat the two statements separately.

1. By way of contradiction assume that the statement is not correct. We will show that in this case D is not minimal.

   Choose $i \in D$, $\alpha \in Int(q_i)$, and $j \in comp(\alpha) \cap D \setminus \{i\}$ with $j(\alpha) \not\subseteq \mathit{en}(q_j)$ and $need(q_j) \cap comp(\alpha) = \emptyset$. Consider the path

$$\pi_{j,\alpha} = \{j\} \stackrel{e}{—} \bar{K} — \ldots — comp(\alpha)$$

according to Lemma 3.3.3 and let $\bar{G} := (V, E \setminus \{e\})$ be defined the same way as before Lemma 3.8.1. We denote by $\hat{D}$ the set of components in D that are reachable from $\{j\}$ in $\bar{G}$:

$$\hat{D} := D \cap \left\{k | \{k\} \text{ is reachable from } \{j\} \text{ in } \bar{G}\right\}$$

Using the same argument as in the proof of Lemma 3.8.3, for all $k' \in comp(\alpha) \setminus \{j\}$ we see that $\{k'\}$ is not reachable from $\{j\}$ in $\bar{G}$. In particular this implies $i \notin \hat{D}$. Therefore, $\hat{D}$ is a proper subset of D.

We will show that $\hat{D}$ is a local deadlock in q. Let $k \in \hat{D}$ and $\bar{\alpha} \in Int(q_k)$. Because $k \in D$ there is a component $l \in D \cap comp(\bar{\alpha})$ with $l(\bar{\alpha}) \not\subseteq \mathcal{en}(q_l)$. Because $k \in \hat{D}$ we know that $\{k\}$ is reachable from $\{j\}$ in $\bar{G}$. Setting $Int'(q_h) = Int(q_h)$ for all $h \in K$, we use Lemma 3.8.1 with $\beta' = \beta = \bar{\alpha}$ and conclude that $\{l\}$ is also reachable from $\{j\}$ in $\bar{G}$. This shows that $l \in \hat{D}$. Thus, $\hat{D}$ is a local deadlock in q such that $\hat{D} \subsetneq D$. This is a contradiction because D was chosen to be minimal.

2. According to the first part of the lemma for all $i \in D$, $\alpha \in Int(q_i)$, and $j \in D \cap comp(\alpha) \setminus \{i\}$ we have $need(q_j) \cap comp(\alpha) \neq \emptyset$ if $j(\alpha) \not\subseteq \mathcal{en}(q_j)$. Because D is a local deadlock, for all $i \in D$ and $\alpha \in Int(q_i)$ at least one such j exists. Since q is reachable we apply Lemma 3.8.6 with $Int'(q_h) = Int(q_h)$ for all $h \in K$ to see that for all $i \in D$ and $\alpha \in Int(q_i)$ there is $j \in D \cap comp(\alpha)$ with $q_j \in PS_j(q_i, \alpha)$.

□

Proposition 3.5.1. *Let Sys be a tree-like interaction system with strongly exclusive communication.*

If the following two conditions hold then Sys is free of local deadlocks:

1. $\forall i \forall q_i : q_i$ *complete* $\vee\, q_i^0 \notin BWS(q_i, need(q_i)) \vee \exists \alpha \in Int(q_i)$ *such that* $\forall j \in comp(\alpha) \setminus \{i\} : q_j^0 \notin BWS(PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$

2. $\forall i \in K$ *and* $\forall q_i \in Q_i$ *that are incomplete:*

$$\bigcap_{\alpha \in Int(q_i)} \bigcup_{j \in comp(\alpha) \setminus \{i\}} \mathcal{PA}(q_i, \alpha, j) = \emptyset$$

The conditions can be checked in time polynomial in the size of Sys.

Proof. Assume that the conditions are satisfied but there exists a local deadlock. There is a path $\sigma = q^0 \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_{l-1}} q^l \xrightarrow{\alpha_l} q$ and a subset $D \subseteq K$ of components such that D is a minimal local deadlock in q. For all $i \in D$ the local state q_i is incomplete. We distinguish two cases:

1. For all $s \leq l$ and all $i \in D$ we have $i(\alpha_s) \not\subseteq \bigcup_{k \in need(q_i)} \boldsymbol{comm}_i(k)$ if $i(\alpha_s) \neq \emptyset$. Because *Sys* has strongly exclusive communication for all $i \in D$ the projection of σ to i yields a path from q_i^0 to q_i that is only labeled with ports in $\mathcal{A}_i \setminus \bigcup_{k \in need(q_i)} \boldsymbol{comm}_i(k)$. For all $i \in D$ we get $q_i^0 \in BWS(q_i, need(q_i))$.

   Lemma 3.8.14 shows that for all $i \in D$ and all $\alpha \in Int(q_i)$ there is a component $j \in D \cap comp(\alpha)$ with $q_j \in PS_j(q_i, \alpha)$. Further $q_j^0 \in BWS(q_j, need(q_j)) \subseteq BWS(PS_j(q_i, \alpha), \mathfrak{need}(PS_j(q_i, \alpha)))$ holds because $j \in D$. This is a contradiction to the first condition above because q_i is incomplete for all $i \in D$.

2. Otherwise let $s_0 \leq l$ be the largest index such that there exists $i \in D \cap comp(\alpha_{s_0})$ with $i(\alpha_{s_0}) \subseteq \bigcup_{k \in need(q_i)} \boldsymbol{comm}_i(k)$. Let i_0 denote such a component. For all $s_0 < s \leq l$ and all $i \in D$ we have $i(\alpha_s) \not\subseteq \bigcup_{k \in need(q_i)} \boldsymbol{comm}_i(k)$ if $i(\alpha_s) \neq \emptyset$. As above for all $i \in D$ this means $q_i^{s_0+1} \in BWS(q_i, need(q_i))$.

   Consider $\alpha \in Int(q_{i_0})$. According to Lemma 3.8.14 there exists $j \in comp(\alpha) \cap D$ with $q_j \in PS_j(q_{i_0}, \alpha)$. Using the same argument as in the proof of Proposition 3.4.1 we get

$$i_0(\alpha_{s_0}) \in \mathcal{PA}(q_{i_0}, \alpha, j) \subseteq \bigcup_{j \in comp(\alpha) \setminus \{i_0\}} \mathcal{PA}(q_{i_0}, \alpha, j),$$

   and repeating the argument for all $\alpha \in Int(q_{i_0})$ we get

$$i_0(\alpha_{s_0}) \in \bigcap_{\alpha \in Int(q_{i_0})} \bigcup_{j \in comp(\alpha) \setminus \{i_0\}} \mathcal{PA}(q_{i_0}, \alpha, j).$$

   This is a contradiction to the second condition above.

The assumption that *Sys* contains a reachable local deadlock is wrong. Therefore the system is free of local deadlocks.

The conditions can be checked in time polynomial in the size of *Sys*, because all parameters can be computed by analyzing *Int*, the T_i, and the subsystems $Sys \downarrow_{\{i,j\}}$ where i and j are interacting components. □

Chapter 4

Progress in Interaction Systems

4.1 Introduction

In this chapter we deal with progress in interaction systems. Since checking progress of a set of components can be reduced to checking progress of a set of ports we will only consider progress of a set $\mathcal{A}_0$ of ports. If one is interested in progress of a subset K' of components the results can be applied with $\bigcup_{i \in K'} \mathcal{A}_i$. We will denote the union $\bigcup_{i \in K'} \mathcal{A}_i$ by $\mathcal{A}_{K'}$. In this chapter we always assume that *Sys* is a *deadlock-free* interaction system.

We first present a characterization of progress of a subset of ports. The section thereafter exhibits a directed graph and a sufficient condition for progress which is based on this graph. We demonstrate how information gathered from the graph can be incorporated into an iteration of the construction of the edges in order to enhance the graph. We explain how to combine the characterization with the graph-criterion if the latter fails to establish progress. The chapter is concluded by an application of the results to the example of the dining philosophers introduced in Chapter 2.

4.2 Characterizing Progress

For a deadlock-free interaction system *Sys* we present a characterization of the subsets $\mathcal{A}_0 \subseteq \mathcal{A}$ that make progress in *Sys*. The complexity result stated in Section 2.2 shows that we cannot expect the conditions of this

characterization to be checkable efficiently in general. The characterization is of theoretical benefit helping to better understand progress in interaction systems. It can be put to practical use in some cases as well, though, because it is fit to be combined with other conditions for progress in cases where these conditions alone do not yield progress of $\mathcal{A}_0$.

In the following for a given set $\mathcal{A}_0$ of ports we need to distinguish the interactions that contain at least one port in $\mathcal{A}_0$ from those that do not.

Definition 4.2.1. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. By $excl\left(\mathcal{A}_0\right) := \left\{\alpha \in Int | \alpha \cap \mathcal{A}_0 = \emptyset\right\}$ we denote the set of interactions that do not contain any port in $\mathcal{A}_0$.*

The interactions in $excl\left(\mathcal{A}_0\right)$ play an important role in the investigation of progress because a run which violates the condition for $\mathcal{A}_0$ to make progress is only labeled with these interactions. This observation will be used in the characterization presented below. The characterization resorts to a labeled transition system where all labels are in $excl\left(\mathcal{A}_0\right)$ (and which is possibly smaller than T_{Sys} itself) and makes a statement about the existence of certain cycles in this transition system.

Definition 4.2.2. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Set:*

$$K_{\overline{\mathcal{A}_0}} := \bigcup_{\alpha \in excl(\mathcal{A}_0)} comp(\alpha)$$

Further define

$$Sys_{\overline{\mathcal{A}_0}} := (K_{\overline{\mathcal{A}_0}}, \{T_i\}_{i \in K_{\overline{\mathcal{A}_0}}}, excl(\mathcal{A}_0))$$

whose induced behavior is given by the labeled transition system $T_{\overline{\mathcal{A}_0}} := (Q_{\overline{\mathcal{A}_0}}, excl(\mathcal{A}_0), \rightarrow)$. Here

1. *$Q_{\overline{\mathcal{A}_0}} := \prod_{i \in K_{\overline{\mathcal{A}_0}}} Q_i$ is the state space of $T_{\overline{\mathcal{A}_0}}$. We denote states by tuples $\overline{q} = (\overline{q}_i)_{i \in K_{\overline{\mathcal{A}_0}}}$.*

2. *$\rightarrow \subseteq Q_{\overline{\mathcal{A}_0}} \times excl\left(\mathcal{A}_0\right) \times Q_{\overline{\mathcal{A}_0}}$ is the labeled transition relation with:*

$$\overline{p} \xrightarrow{\alpha} \bar{q} \Leftrightarrow \forall i \in K_{\overline{\mathcal{A}_0}}\ \overline{p}_i \xrightarrow{a_i} \overline{q}_i \text{ if } i(\alpha) = \{a_i\} \text{ and } \overline{p}_i = \overline{q}_i \text{ otherwise.}$$

$K_{\overline{\mathcal{A}_0}}$ contains those components that participate in at least one interaction not involving any port in $\mathcal{A}_0$. Conversely, the components in $K \backslash K_{\overline{\mathcal{A}_0}}$ are those components which need the ports in $\mathcal{A}_0$ in the sense that whenever an interaction involving a component in $K \backslash K_{\overline{\mathcal{A}_0}}$ is performed then a port in $\mathcal{A}_0$ must also participate. In other words, for all $\alpha \in \mathit{excl}(\mathcal{A}_0)$ and all $k \in K \backslash K_{\overline{\mathcal{A}_0}}$ we have $k(\alpha) = \emptyset$. Note that $i \in K \backslash K_{\overline{\mathcal{A}_0}}$ always holds for a component i if $\mathcal{A}_i \subseteq \mathcal{A}_0$. $K_{\overline{\mathcal{A}_0}}$ is relevant when we are interested in progress of $\mathcal{A}_0$ because if there is a run which does not allow $\mathcal{A}_0$ to participate (and therefore refutes progress of $\mathcal{A}_0$) then every interaction on this run only involves components in $K_{\overline{\mathcal{A}_0}}$. Thus, the behavior of $\mathit{Sys}_{\overline{\mathcal{A}_0}}$ can be understood as the part of *Sys* which may possibly prevent $\mathcal{A}_0$ from making progress. Therefore it suffices to only investigate $T_{\overline{\mathcal{A}_0}}$. This result is stated in the following theorem.

Theorem 4.2.1. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports.*

$\mathcal{A}_0$ makes progress in Sys if and only if $T_{\overline{\mathcal{A}_0}}$ does not contain any cycles visiting a state $\overline{q}$ for which there exists $q' \in \prod_{i \in K \backslash K_{\overline{\mathcal{A}_0}}} Q_i$ such that $(\overline{q}, q')$ is in $\mathit{reach}(\mathit{Sys})$.

$\mathit{Sys}_{\overline{\mathcal{A}_0}}$ does not have an initial state. It is possible to reach a cycle in *Sys* that is only labeled with interactions in $\mathit{excl}(\mathcal{A}_0)$ by first performing some interactions that involve $\mathcal{A}_0$. Therefore, we must consider all cycles in $T_{\overline{\mathcal{A}_0}}$ because the interactions that involve $\mathcal{A}_0$ are not present in $\mathit{Sys}_{\overline{\mathcal{A}_0}}$. This characterization cannot be tested efficiently in general. First of all, it is not clear that going from K to $K_{\overline{\mathcal{A}_0}}$ substantially reduces the size of the system to be investigated. In the worst case every component participates in at least one interaction $\alpha \in \mathit{excl}(\mathcal{A}_0)$. Then the system is not reduced in size at all. But even in cases where $\mathit{Sys}_{\overline{\mathcal{A}_0}}$ results in a remarkably smaller system than *Sys* itself, it may still the case that the condition stated in the theorem does not help to decide progress of a given set $\mathcal{A}_0$ of ports. This is because the condition makes a statement about the reachability of *global* states. This is not surprising, though, because the difficulty of deciding progress also stems from the fact that the definition refers to runs starting in *reachable* states. Clearly, we cannot expect to get rid of this factor in the

complexity of deciding progress by formulating a different characterization.

Nonetheless, the characterization is not solely of theoretical interest. Despite the problems mentioned above progress of $\mathcal{A}_0$ may be deduced in some cases where this is not clear from the original definition. If $\mathit{excl}(\mathcal{A}_0)$ contains few interactions which do not involve many components (in relation to the total number n of components) then $T_{\overline{\mathcal{A}_0}}$ may be relatively small and sparse. In the extreme case every interaction involves a port in $\mathcal{A}_0$, and $K_{\overline{\mathcal{A}_0}}$ is empty. In this case it is clear that $\mathcal{A}_0$ makes progress anyway, though. If $T_{\overline{\mathcal{A}_0}}$ does not contain any cycles then the condition of the theorem is satisfied. But even if there are cycles it may be possible to refute their reachability by other techniques which are concerned with reachability of global states (cf. Section 3.2 or Majster-Cederbaum et al. [100, 102] for other techniques in the context of interaction systems, for example). If one tries to establish progress of $\mathcal{A}_0$ directly from Definition 2.2.2 in such a situation, it may not be clear at all that every run involves a port in $\mathcal{A}_0$ even though $\mathit{excl}(\mathcal{A}_0)$ does not contain many elements.

4.3 Testing Progress

We turn to the question of how progress of a set of ports can be tested in practice. We present a graph-based criterion for progress. The definition of the graph has passed through several stages making the criterion more precise. We will take the following approach here: In the first subsection we give the general definition of the graph which involves a parameter d referring to the size of the subsystems that have to be investigated for the construction of the edges. We then turn to the special case where $d = 1$ which constitutes the graph as it was originally conceived. For larger d we get a more general criterion at the cost of an increase in complexity. The original result using the graph for $d = 1$ can then be formulated as a corollary. This corollary is less powerful, however it allows for a simplification of notions. The second subsection deals with an extension of the graph. Iterating the definition of the edges, we incorporate information about progress gathered from the graph constructed so far into the further construction of the edges. The important point about the criterion is that the construction of the graph

and checking the condition can be performed automatically and efficiently where the degree of efficiency is controlled by d.

4.3.1 A Graph Criterion

The directed graph we use to test progress of $\mathcal{A}_0$ has one node for every component. In addition, there is a special node 0 representing the set $\mathcal{A}_0$ that we check for progress. We want to realize a notion of directed edges where for components i and j the existence of an edge (i, j) indicates that j "needs" i in the sense that no matter how the system behaves j can only participate in a finite number of global transitions before a transition labeled with an interaction involving a port in $\mathcal{A}_i$ *must* be performed[1]. Similarly, we want to have edges $(0, j)$ for those components j that "need" $\mathcal{A}_0$ in the same sense. Agreeing to this meaning of the edges for the moment, it can be seen that all components that are reachable from a given component i can only participate in a finite number of global transitions before i does so as well. The test for progress of $\mathcal{A}_0$ presented in our criterion will come down to a simple reachability analysis in the graph where we check whether all nodes are reachable from 0. The important point concerning the applicability of the criterion is to make sure that for the definition of the edges we only have to resort to information that can be checked *locally* in order to make sure that the graph can be constructed efficiently. The reachability analysis in the graph can be performed efficiently as well, because the graph has $n + 1$ nodes and at most n^2 edges.

We define a graph reflecting the informal requirement on the edges stated above. The construction of the graph only requires the investigation of subsystems of the global system. We first need a notion describing for such a subsystem the fact that interactions from a given subset of interactions occur on every cycle in that subsystem. For this purpose we introduce two different notions of *inevitability* of a set of interactions. These notions are independent of the fact that we are only going to use them for interaction systems. We state them for a general finite labeled transition system and

[1]This requirement can be rephrased saying that j can only participate infinitely often in a run of *Sys* if i does so as well.

then derive the corresponding notions for interaction systems.

Definition 4.3.1. *Let $T = (Q, \mathcal{L}, \rightarrow)$ be a finite labeled transition system where Q is the set of states, $\mathcal{L}$ is the set of labels and $\rightarrow \subseteq Q \times \mathcal{L} \times Q$ is the labeled transition relation. Let $\mathcal{L}', \mathcal{L}'' \subseteq \mathcal{L}$ be subsets of labels.*

1. *$\mathcal{L}'$ is **inevitable with respect to** $\mathcal{L}''$ in T if for every cycle $\pi = q \stackrel{\alpha_0}{\rightarrow} q^1 \stackrel{\alpha_1}{\rightarrow} \ldots \stackrel{\alpha_l}{\rightarrow} q$ in T the following condition holds:*

$$\exists 0 \leq r \leq l \text{ with } \alpha_r \in \mathcal{L}'' \Rightarrow \exists 0 \leq s \leq l \text{ with } \alpha_s \in \mathcal{L}'$$

2. *$\mathcal{L}'$ is **inevitable** in T if for every cycle $\pi = q \stackrel{\alpha_0}{\rightarrow} q^1 \stackrel{\alpha_1}{\rightarrow} \ldots \stackrel{\alpha_l}{\rightarrow} q$ in T the following condition holds:*

$$\exists 0 \leq s \leq l \text{ with } \alpha_s \in \mathcal{L}'$$

Let Sys be an interaction system. Let j be a component and let $Int' \subseteq Int$ be a set of interactions.

1. *Int' is **inevitable with respect to** j in Sys if Int' is inevitable with respect to $Int(j)$ in T_{Sys}.*

2. *Int' is **inevitable** in Sys if Int' is inevitable in T_{Sys}.*

Int' is inevitable in *Sys* if on every cycle in *Sys* at least one interaction in Int' occurs. Because every cycle can be used to construct a run and every run contains at least one cycle (using the fact that *Sys* is finite), this means that on every run of Sys an interaction in Int' occurs. Inevitability of Int' in *Sys* implies that the set $\bigcup_{\alpha \in Int'} \alpha$ of ports that are contained in at least one interaction in Int' makes progress in *Sys*. The converse is not true, though. There are systems such that a set Int' of interactions is not inevitable even though the set of ports contained in at least one interaction in Int' makes progress. Inevitability of Int' with respect to a component j generalizes this notion by requiring an interaction in Int' to occur only on those cycles that j also participates in. Note that inevitability of Int' implies inevitability of Int' with respect to any component j.

Definition 4.3.2. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports. Let $1 \leq d \leq n$ be given. The* ***d-th stage progress graph*** *for Sys and $\mathcal{A}_0$ is defined by:*

$$\mathcal{G}_d^1 := (\mathcal{V}, \mathcal{E}_d^1)$$

Here $\mathcal{V} := K \cup \{0\}$. The set of directed edges is given by

$$\begin{aligned}\mathcal{E}_d^1 := \{(i,j) \,|\, & j \neq i, 0 \text{ and there is } K_d \subseteq K \text{ with } |K_d| = d \text{ and } j \in K_d \text{ such} \\ & \text{that } Int\downarrow_{K_d} \setminus excl(\mathcal{A}_i)\downarrow_{K_d} \text{ is inevitable with respect to } j \text{ in} \\ & Sys\downarrow_{K_d}\}\end{aligned}$$

For the construction of $\mathcal{E}_d^1$ we have to consider inevitability of a certain subset of interactions of $Sys\downarrow_{K_d}$. Namely, for a possible edge (i,j) these are the interactions in $Int\downarrow_{K_d} \setminus excl(\mathcal{A}_i)\downarrow_{K_d}$. We take a closer look at what it means for an interaction to be contained in this set. We defined $excl(\mathcal{A}_i)$ to be the set of interactions of the global system that do not allow any port in $\mathcal{A}_i$ to participate. When we project this set of interactions to K_d we obtain those interactions in $Int\downarrow_{K_d}$ that are contained in *at least* one such global interaction. By removing the interactions in $excl(\mathcal{A}_i)\downarrow_{K_d}$ from $Int\downarrow_{K_d}$ we are left with those interactions α_d in $Int\downarrow_{K_d}$ such that *every* global interaction containing α_d must also involve *at least* one port in $\mathcal{A}_i$. Therefore the interactions in $Int\downarrow_{K_d} \setminus excl(\mathcal{A}_i)\downarrow_{K_d}$ represent global interactions involving $\mathcal{A}_i$. Keeping track of these interactions in the subsystems may help to detect progress of a set of ports. The definition of a subsystem makes sure that every global step can be retraced in the subsystem. By requiring inevitability of $Int\downarrow_{K_d} \setminus excl(\mathcal{A}_i)\downarrow_{K_d}$ with respect to j we see that in this case every run σ of Sys which j participates infinitely often in must also involve $\mathcal{A}_i$ in some step. Otherwise the projection of σ to K_d would result in a cycle that allows j to participate at some point but does not contain any label in $Int\downarrow_{K_d} \setminus excl(\mathcal{A}_i)\downarrow_{K_d}$. In particular, this means that an edge (i,j) indeed conveys the meaning that was motivated above. It also becomes clear that 0 represents the set $\mathcal{A}_0$ of ports since we consider $\mathcal{A}_0$ in the definition when we check the existence of a possible edge $(0,j)$. Note that $Int\downarrow_{K_d} \setminus excl(\mathcal{A}_i)\downarrow_{K_d}$ is not inevitable in $Sys\downarrow_{K_d}$ if there is a cycle that is only labeled with interactions in $excl(\mathcal{A}_i)\downarrow_{K_d}$. Similarly, $Int\downarrow_{K_d} \setminus excl(\mathcal{A}_i)\downarrow_{K_d}$ is not inevitable

with respect to j in $Sys\downarrow_{K_d}$ if there is a cycle that allows j to participate and that is only labeled with interactions in $excl(\mathcal{A}_i)\downarrow_{K_d}$.

Before considering the construction of $\mathcal{G}_d^1$ by means of an example we want to emphasize two issues concerning the choice of d. First, for the special case given by $d = 1$ the definition of the edges can be simplified. For the construction of a d-stage edge (i, j) we have to find a set K_d containing j and $d-1$ other components such that the conditions are satisfied. If $d = 1$ there is only one possible choice. Namely, K_1 must be equal to $\{j\}$. In this case $Sys\downarrow_{\{j\}}$ and $Int\downarrow_{\{j\}}$ coincide with T_j respectively $\mathcal{A}_j$ (up to identification of $\{a_j\}$ and (q_j) with a_j respectively q_j). The following lemma shows that it suffices to check inevitability of $\mathcal{A}_j\backslash excl_j(\mathcal{A}_i)$ in T_j. Here $excl_j(\mathcal{A}_i) \subseteq \mathcal{A}_j$ is the subset of ports that occur in at least one interaction not involving $\mathcal{A}_i$.

Lemma 4.3.1. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports.*

Setting $excl_j(\mathcal{A}_i) := \{a_j | \exists \alpha \in excl(\mathcal{A}_i) : a_j \in \alpha\}$, we have

$$\mathcal{E}_1^1 := \{(i, j) \,| j \neq i, 0 \text{ and } \mathcal{A}_j\backslash excl_j(\mathcal{A}_i) \text{ is inevitable in } T_j\}.$$

Further, we delve into the relation between the progress graphs of different stages. It is more expensive to compute a higher stage graph because we hide fewer components in the subsystems we consider. Therefore, it only makes sense to do so if this increase in complexity yields more specific information. In particular, the existence of an edge $(i, j) \in \mathcal{E}_{d'}^1$ should imply the existence of the same edge in $\mathcal{E}_d^1$ for any $d > d'$. This way we can be sure that once we have decided upon a parameter d (and therefore have chosen the degree of complexity we are willing and/or able to handle) we do not have to concern ourselves with any stage of the progress graph which is smaller than d. The following lemma shows that this statement is true.

Lemma 4.3.2. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d' < d \leq n$ and let $i \in K \cup \{0\}$ and $j \in K$.*

If $(i, j) \in \mathcal{E}_{d'}^1$ then we also have $(i, j) \in \mathcal{E}_d^1$.

Example 4.3.1. Consider $K_1 := \{1, 2, 3, 4\}$. The transition systems of the components are depicted in Figure 4.1. For each $i \in K_1$ the port set $\mathcal{A}_i$ is understood to coincide with the set of labels of T_i. We define Int_1 to contain

the interactions $\{a_1, a_3\}$, $\{a_2, a_3\}$, $\{b_1, b_3\}$, $\{b_2, b_3\}$, $\{c_1, c_2\}$, $\{d_1, d_4\}$, and $\{e_1, e_4\}$. We denote the induced interaction system by Sys_1.

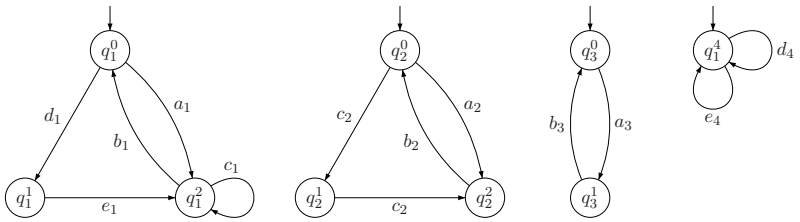

Figure 4.1: The local behavior of the components in K_1

We consider the question whether $\mathcal{A}_0 := \{a_3, d_4\}$ makes progress in Sys_1. We construct the first and second stage progress graph for Sys_1 and $\mathcal{A}_0$. They are given in Figure 4.2. The edges in $\mathcal{E}_1^1 \subseteq \mathcal{E}_2^1$ are depicted as solid lines whereas the edges that are only in $\mathcal{E}_2^1$ are depicted as dashed lines.

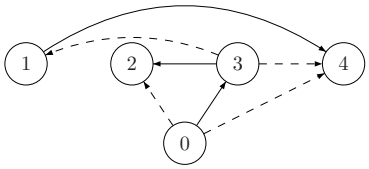

Figure 4.2: The first and second stage progress graphs of Sys_1

We go into a few details concerning the construction. We have $(0, 3) \in \mathcal{E}_1^1$. Intuitively, this can be seen as follows: Whenever an interaction α with $a_3 \in \alpha$ is executed in Sys_1 then $\mathcal{A}_0$ also participates in α (this is clear because a_3 itself is in $\mathcal{A}_0$). However, in T_3 at most one transition can be executed before the transition labeled a_3 must be executed. Together this implies that *if* component 3 participates infinitely often in a run *then* eventually a port in $\mathcal{A}_0$ also participates in that run (in fact, this will be the case for every other step, that 3 participates in, of the run). Considering the formal definition of $\mathcal{E}_1^1$, we first compute $\mathcal{A}_3 \backslash excl_3(\mathcal{A}_0) = \{a_3\}$, i.e., the set of ports of 3 that only occur in global interactions that also involve $\mathcal{A}_0$. This set of ports is inevitable in T_3, and therefore $(0, 3) \in \mathcal{E}_1^1$. Similarly, the existence of the other edges in $\mathcal{E}_1^1$ can be explained. On the other hand $(3, 1) \notin \mathcal{E}_1^1$, for example, because in T_1 there is a cycle that is only labeled with the port

$c_1 \in \mathit{excl}_1(\mathcal{A}_3)$.

On the other hand we have $(3,1) \in \mathcal{E}_2^1$. Choosing $K_2 := \{1,2\}$, we see $\mathit{excl}(\mathcal{A}_3) \downarrow_{K_2} = \{\{c_1, c_2\}, \{e_1\}, \{d_1\}\}$. These are the interactions in $\mathit{Int} \downarrow_{K_d}$ which occur in at least one global interaction which does not involve 3. In $\mathit{Sys}_1 \downarrow_{K_2}$ there are no cycles that are only labeled with these interactions. In particular, the loop in q_1^2 whose label c_1 does not need 3 and which was the reason for $(3,1) \notin \mathcal{E}_1^1$ is forced open by pairing component 1 with component 2. Inevitability of $\mathit{Int} \downarrow_{K_2} \setminus \mathit{excl}(\mathcal{A}_3) \downarrow_{K_2}$ follows (i.e., in $\mathit{Sys} \downarrow_{K_d}$ only a finite number of steps is possible before an interaction which globally *always* requires cooperation with 3 *must* be performed) showing that $(3,1) \in \mathcal{E}_2^1$. For the construction of the edge $(0,4)$ we profit from the fact that an edge (i,j) is defined requiring inevitability of $\mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_i) \downarrow_{K_d}$ *with respect to* j (for a suitable set K_d) rather than mere inevitability of $\mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_i) \downarrow_{K_d}$: For $K_2 = \{1,4\}$ we have $\mathit{excl}(\mathcal{A}_0) \downarrow_{K_2} = \{\{b_1\}, \{c_1\}, \{e_1, e_4\}\}$. In $\mathit{Sys}_1 \downarrow_{K_2}$ there is a cycle which is only labeled with interactions from this set (namely the loop labeled $\{c_1\}$). Nonetheless, $\mathit{Int} \downarrow_{K_2} \setminus \mathit{excl}(\mathcal{A}_0) \downarrow_{K_2}$ is inevitable with respect to 4 because this cycle does not involve 4.

In order to be able to use the graph in practice it is necessary that it can be constructed efficiently. Considering the definition of the edges it is clear that the "most efficient" we can get is $\mathcal{O}(|T_{max}|^d)$ where $|T_{max}|$ is the size of the largest local system because we investigate subsystems of size d. All the same, it is not even obvious at first glance whether checking inevitability of $\mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_i) \downarrow_{K_d}$ with respect to j in a given subsystem of size $\mathcal{O}(|T_{max}|^d)$ can indeed be performed within these bounds. It is not feasible to simply check every existing cycle for the condition stated in the definition because the number of cycles in a directed graph can be more than exponentially large in the number of nodes (cf. Tarjan [133], for example). Nonetheless, a procedure to check the conditions efficiently can be conceived. It will be given in the proof section to this chapter. Momentarily we only explain what the procedure does and argue why it can be implemented in time linear in the product of $|T_{\mathit{Sys}\downarrow_{K_d}}|$ and $|\mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_i) \downarrow_{K_d}|$.

The procedure first removes all edges labeled with an interaction in $\mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_i) \downarrow_{K_d}$ from $T_{\mathit{Sys}\downarrow_{K_d}}$. This can be done in the time bounds

given above. Then it performs a check for cycles in the remaining graph. Checking whether a directed graph contains cycles can be done in time linear in the size of the graph, i.e., in $\mathcal{O}(|T_{max}|^d)$ in our case. If there are no cycles then every cycle of the original system contained at least one edge labeled with an interaction in $Int \downarrow_{K_d} \setminus excl(\mathcal{A}_i) \downarrow_{K_d}$. Otherwise we have to make sure that none of the cycles in the remaining system contain an edge labeled with an interaction involving j. We compute the strongly connected components (SCCs) of the remaining system. For each of these we check whether they contain an edge labeled with an interaction involving j. It can be seen that this is the case for an SCC if and only if in this SCC there is a cycle containing an edge labeled with an interaction involving j. Again, the computation of the SCCs of a directed graph can be performed in time linear in the size of the graph. The same is true for checking whether there is an edge labeled with an interaction involving j within one of these SCCs. Thus, the procedure complies with the required time bounds. The following lemma is the crucial point regarding the applicability of the progress graph.

Lemma 4.3.3. *Let Sys be an interaction system and let* $1 \leq d \leq n$. *Let* $\mathcal{A}_0 \subseteq \mathcal{A}$ *be a set of ports.*

The d-th stage progress graph for Sys and $\mathcal{A}_0$ *can be constructed in time polynomial in the size of Sys. An upper bound for the cost is given by*

$$\mathcal{O}(|Int|^2 n^{d+2} + |Int| n^{d+1} |T_{max}|^d)$$

where $|T_{max}|$ *denotes the size of the largest local transition system.*

The lemma only states a rough upper bound for the cost of constructing the graph. This bound is independent of the underlying implementation used for interaction systems. Depending on the chosen implementation the bound can be made more precise. The important point is the fact that the parameter d is the *variable* factor where incrementing d increases the degree of the polynomial bounding the complexity. The statement of Lemma 4.3.2 shows that this increase in complexity pays off because we obtain a more general graph and therefore also a more general criterion. The size d of the subsystems to be investigated is indeed the lever which can be used to adjust the criterion to a given situation.

Proposition 4.3.1. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d \leq n$ and let $\mathcal{G}_d^1$ be defined as above.*

If all nodes in $\mathcal{V}$ are reachable from 0 in $\mathcal{G}_d^1$ then $\mathcal{A}_0$ makes progress in Sys. The condition can be checked in time polynomial in the size of Sys.

The proposition only requires reachability in $\mathcal{G}_d^1$ starting from 0. This explains why Definition 4.3.2 excludes edges entering 0. Such edges are not needed. The proof of the proposition is based on a simple inductive argument. On every run there is a component j that participates infinitely often. Since there is a path from 0 to j in $\mathcal{G}_d^1$ we may repeat the argument motivating the definition of $\mathcal{E}_d^1$ for every edge on the path to conclude that eventually some port in $\mathcal{A}_0$ must participate in the run.

Example 4.3.1 continued: The proposition together with $\mathcal{G}_2^1$ depicted in Figure 4.2 (cf. p. 139) shows that $\mathcal{A}_0$ makes progress in Sys_1. $\mathcal{G}_1^1$ is not sufficient to show that $\mathcal{A}_0$ makes progress in Sys_1.

Using Lemma 4.3.2 we immediately get the following corollary which shows that increasing the parameter d yields a more powerful criterion.

Corollary 4.3.1. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d' < d \leq n$ and let $\mathcal{G}_{d'}^1$ respectively $\mathcal{G}_d^1$ be defined as above.*

If all components are reachable from 0 in $\mathcal{G}_{d'}^1$ then they are also reachable from 0 in $\mathcal{G}_d^1$.

Wrapping things up, we conclude by taking a closer look at two points worth noting. First, we say a few words about the implications arising from the existence of one or more components i with $\mathcal{A}_i \subseteq \mathcal{A}_0$. Investigating progress of a set $K' \subseteq K$ constitutes a special case of this situation. Having motivated that an edge (i, j) has the meaning that j "needs" i, one would expect that there is an edge $(0, j)$ for any component j with $\mathcal{A}_j \subseteq \mathcal{A}_0$ because in this case whenever j participates with port a_j in an interaction it is clear that $\mathcal{A}_0$ also participates in that interaction. Therefore j indeed "needs" $\mathcal{A}_0$ to proceed. The following lemma formally substantiates this argument.

Lemma 4.3.4. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of*

ports. Let j be a component with $\mathcal{A}_j \subseteq \mathcal{A}_0$.

For all d there is an edge $(0, j) \in \mathcal{E}_d^1$.

Finally, we turn to the role of the node 0. One might argue that at least for those cases where progress of a set K' of components (i.e., $\mathcal{A}_0 = \mathcal{A}_{K'}$) is investigated the additional node 0 which was somehow artificially introduced into $\mathcal{G}_d^1$ is not necessary and could be eliminated. In this case a slightly modified progress graph, that is defined in the same way as in Definition 4.3.2 except that 0 and the corresponding edges are omitted, could be used. In order to show that K' makes progress in *Sys* the criterion would have to be reformulated requiring that for every $j \in K$ there has to be $i \in K'$ such that j is reachable from i in the modified graph. This criterion is sound. Further, it is clear that whenever the condition based on the modified graph shows that K' makes progress then Proposition 4.3.1 using $\mathcal{G}_d^1$ does so as well. This is because $\mathcal{G}_d^1$ contains all edges that exist in the modified graph. Additionally, there may be edges from 0 to certain components. Lemma 4.3.4 shows that there is an edge $(0, i)$ for all $i \in K'$. Then it is clear that all components are reachable from 0 in $\mathcal{G}_d^1$ if all components are reachable from a component $i \in K'$ in the modified graph. The following example shows that the criterion which is based on $\mathcal{G}_d^1$ is in fact more general for a fixed parameter d.

Example 4.3.2. Consider $K_2 := \{5, 6, 7, 8\}$. The transition systems of the components are depicted in Figure 4.3. For $i \in K_2$ the port set $\mathcal{A}_i$ is understood to coincide with the set of labels of T_i. We set $Int_2 := \{\{a_5, a_7\}, \{b_5, b_7\}, \{e_5\}, \{c_6, c_7\}, \{d_6, d_7\}, \{f_6\}, \{g_7, g_8\}, \{h_7, h_8\}\}$, and we call the induced interaction system Sys_2.

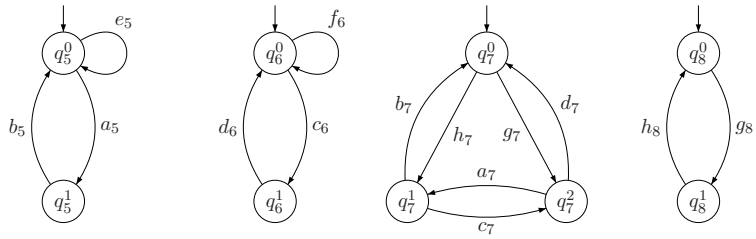

Figure 4.3: The local behavior of the components in K_2

We consider the question whether $\mathcal{A}_0 := \mathcal{A}_5 \cup \mathcal{A}_6$ makes progress in Sys_2.

Figure 4.4 depicts $\mathcal{G}_1^1$.

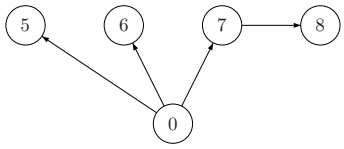

Figure 4.4: The first stage progress graph of Sys_2

For each component $i \neq 8$ there is an edge $(0, i) \in \mathcal{E}_1^1$. Further $(7, 8) \in \mathcal{E}_1^1$. Proposition 4.3.1 shows that $\mathcal{A}_0$ makes progress in Sys_2. Figure 4.4 also shows that it would not be possible to apply the criterion using the modified graph without 0 motivated above in order to show that $\mathcal{A}_0$ makes progress because there are no edges from 5 or 6 to 7 and 8 in $\mathcal{E}_1^1$.

This can be explained as follows: The extra node 0 allows to identify the union of $\mathcal{A}_5$ and $\mathcal{A}_6$ with the single port set $\mathcal{A}_0$. Without doing so, 7 neither needs 5 nor 6 since it may always exclude one of these components by only interacting with the other. However, 7 cannot proceed without any of the ports in the combined port set $\mathcal{A}_0$ being performed eventually.

4.3.2 Enhancing the Graph

We conclude this section by looking at an extension of the progress graph. In the previous subsection we interpreted the fact that a component j is reachable in $\mathcal{G}_d^1$ from i such that j eventually needs component i in order to advance. This way it was possible to formulate Proposition 4.3.1 requiring that all nodes are reachable from 0 in $\mathcal{G}_d^1$. If this condition is not satisfied the information derived can be used to iterate the construction of the edges. Before formalizing the approach, we demonstrate this iteration by means of an example. The components in this example do not do very much but they illustrate the idea we utilize.

Example 4.3.3. Consider $K_3 = \{9, 10, 11\}$. The transition systems of the components are depicted in Figure 4.5. For each i the port set $\mathcal{A}_i$ is understood to coincide with the set of labels of T_i. We define $Int_3 := \{\{a_9, a_{10}\}, \{b_{10}, b_{11}\}, \{c_9, c_{11}\}\}$. The induced interaction system is Sys_3.

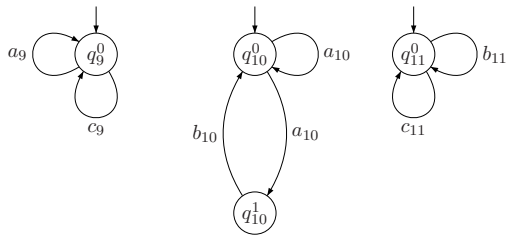

Figure 4.5: The local behavior of the components in K_3

We consider the question whether $\mathcal{A}_0 := \mathcal{A}_9$ makes progress in Sys_3. The first stage progress graph is depicted in Figure 4.6. We cannot use

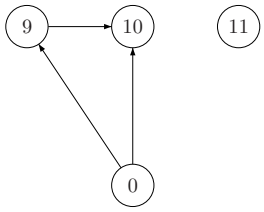

Figure 4.6: The first stage progress graph for Sys_3

Proposition 4.3.1 to infer that $\mathcal{A}_9$ makes progress because 11 is not reachable from 0. However, we know that component 10 can only participate in a finite number of global steps before 9 has to participate because 10 is reachable from 9. Thus, if we are able to show that in T_{11} the set of ports that need cooperation with 9 or 10 (i.e., $\mathcal{A}_{11} \backslash excl_{11}(\mathcal{A}_9 \cup \mathcal{A}_{10})$) is inevitable it would make sense to introduce the edge $(9, 11)$ during the first iteration process because if 11 repeatedly needs 10 we already know that indirectly it will also need 9 to participate at some point. We have $excl(\mathcal{A}_9 \cup \mathcal{A}_{10}) = \emptyset$. This implies $\mathcal{A}_{11} \backslash excl_{11}(\mathcal{A}_9 \cup \mathcal{A}_{10}) = \mathcal{A}_{11}$ which is inevitable in T_{11}. Thus, we add an edge $(9, 11)$ after the first iteration, and 11 is reachable from 0. This shows that $\mathcal{A}_0 = \mathcal{A}_9$ does make progress in Sys_3 after all.

These considerations suggest the following iteration of the construction of the edges: During the m-th phase of the construction we first compute for each node i the set of nodes that are reachable from i according to the edges constructed so far. For the edge (i, j) we then consult to $Int \downarrow_{K_d}$

$\backslash excl(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d}$ in the check for inevitability with respect to j. Here $\mathcal{A}_{r_d^{m-1}(i)}$ denotes the union of the port sets of those nodes that are reachable from i in the progress graph containing the edges constructed during the first $m-1$ phases.

Definition 4.3.3. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports. Let $1 \leq d \leq n$. The **extended d-th stage progress graph** for Sys and $\mathcal{A}_0$ is defined as follows:*

$$\mathcal{G}_d := (\mathcal{V}, \mathcal{E}_d)$$

The set of nodes is $\mathcal{V} := K \cup \{0\}$. The set of edges is $\mathcal{E}_d := \bigcup_{m \geq 1} \mathcal{E}_d^m$. For $m \geq 1$ the sets $\mathcal{E}_d^m$ of directed edges are inductively defined by:

$$\begin{aligned}\mathcal{E}_d^m := \{(i,j) \,|\, & j \neq i, 0 \text{ and } \exists K_d \subseteq K \text{ with } |K_d| = d \text{ and } j \in K_d \text{ such that} \\ & \mathit{Int} \downarrow_{K_d} \backslash excl(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d} \text{ is inevitable with respect to } j \\ & \text{in } \mathit{Sys} \downarrow_{K_d}\}\end{aligned}$$

Here $r_d^{m-1}(i) := \{k | k \text{ is reachable from } i \text{ in } (\mathcal{V}, \bigcup_{l=1}^{m-1} \mathcal{E}_d^l)\}$ denotes the set of nodes that are reachable from i in the graph constructed so far.

$\mathcal{E}_d^1$ also occurs in Definition 4.3.2. Note that $r_d^0(i) = \{i\}$ because for $m = 1$ the graph that has been constructed so far does not contain any edges. Therefore the original definition of $\mathcal{E}_d^1$ coincides with the one given above for $m = 1$. In particular, $\mathcal{G}_d^1$ is a subgraph of the extended progress graph. We get statements analogous to the results stated in the previous subsection.

Lemma 4.3.5. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports.*

1. *For $m \geq 1$ we have*

$$\mathcal{E}_1^m := \left\{(i,j) \,|\, j \neq i, 0 \text{ and } \mathcal{A}_j \backslash excl_j(\mathcal{A}_{r_1^{m-1}(i)}) \text{ is inevitable in } T_j\right\}.$$

2. *Let $1 \leq d' < d \leq n$ and $m \geq 1$ and let $i \in K \cup \{0\}$ and $j \in K$. If $(i,j) \in \mathcal{E}_{d'}^m$ then we also have $(i,j) \in \mathcal{E}_d^m$.*

For the construction of the first stage of the extended progress graph we may again resort to the more simple definition of inevitability. The second part of the lemma states that for a chosen stage d of the extended progress graph we do not have to consider any stage d' where $d' < d$ because $\mathcal{E}_{d'} \subseteq \mathcal{E}_d$.

The extended progress graph of stage d can be constructed efficiently.

Lemma 4.3.6. *Let Sys be an interaction system and let $1 \leq d \leq n$. Let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports.*

The extended d-th stage progress graph for Sys and $\mathcal{A}_0$ can be constructed in time polynomial in the size of Sys. An upper bound for the cost of the construction is given by

$$\mathcal{O}(|Int|^2 n^{d+4} + |Int| n^{d+3} |T_{max}|^d)$$

where $|T_{max}|$ denotes the size of the largest local transition system.

Proposition 4.3.2. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d \leq n$ and let $\mathcal{G}_d$ be defined as above.*

If all nodes in $\mathcal{V}$ are reachable from 0 in $\mathcal{G}_d$ then $\mathcal{A}_0$ makes progress in Sys. The condition can be checked in time polynomial in the size of Sys.

Using the second part of Lemma 4.3.5 we get the following corollary showing that increasing the parameter d yields a more powerful criterion.

Corollary 4.3.2. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d' < d \leq n$ and let $\mathcal{G}_{d'}$ respectively $\mathcal{G}_d$ be defined as above.*

If all components are reachable from 0 in $\mathcal{G}_{d'}$ then they are also reachable from 0 in $\mathcal{G}_d$.

4.3.3 Failure of the Criterion

We conclude by discussing situations where Proposition 4.3.2 fails to provide an answer to the question whether $\mathcal{A}_0$ makes progress. It is clear that this case arises since the criteria we presented only constitute *sufficient* conditions. We have already mentioned that deciding progress is PSPACE-complete. Therefore there will always be situations where the conditions do not hold and we cannot make any statement about progress of the set

of ports in question. One technique which could be used in a first attempt to try and prove that $\mathcal{A}_0$ makes progress if Proposition 4.3.2 does not yield this information even though there is reason to believe that $\mathcal{A}_0$ does make progress has already been provided by the previous subsections: We may increment d in the hope to get a more precise criterion. Of course there are limitations to this approach. We cannot be sure whether we indeed get more information by increasing d, and after all there is still the possibility that the condition is not satisfied simply because $\mathcal{A}_0$ does not make progress. In this case increasing d does not make any difference. The condition of Proposition 4.3.2 will never be satisfied. If nothing else, this technique is limited by our ability to handle subsystems of increasing size.

Thus, we are dealing with the question what to do if Proposition 4.3.2 is not helpful in deciding whether $\mathcal{A}_0$ makes progress. It would of course be possible in this case to try to check the conditions in the definition of progress of $\mathcal{A}_0$ directly or to revert to model-checking. Any such approach has to cope with the problem of the state space explosion, though. Moreover such an approach does not use the information gathered during the construction of the graph in any way whatsoever. Thus, it contravenes the paradigm we stated in Section 1.2.2.

In the following we briefly suggest an approach that may help to answer the question about progress without investigating the complete system even if the conditions for the graph criteria presented above do not hold. In particular, we pay attention to making sure that we use the information gathered in the construction of the graph. We will combine this information with the characterization of progress from Section 4.2.

All steps taken in the motivation and construction of the various progress graphs were aimed at getting a notion of directed edges (i, j) conveying the meaning that j can only participate infinitely often in a run if i does so as well. Extending this statement to all runs (i.e., j can only participate infinitely often in *every* run if i does so as well), we see that an edge (i, j) can also be interpreted in such a way that it has the significance that j can only make progress in *Sys* if i also makes progress in *Sys*. When we deal with a concrete case where progress of a set $\mathcal{A}_0$ has to be checked this means that once we have constructed the progress graph of a certain stage and have

computed the set $r_d(0)$ of nodes that are reachable from 0 we might as well try to decide progress of $\mathcal{A}_{r_d(0)}$. The statement above shows that this set can only make progress if $\mathcal{A}_0$ itself makes progress. This argument results in the following lemma.

Lemma 4.3.7. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d \leq n$ and let $\mathcal{G}_d$ be defined as above.*

$\mathcal{A}_0$ makes progress in Sys if and only if $\mathcal{A}_{r_d(0)}$ makes progress in Sys where we write $r_d(i) := \{j \in \mathcal{V} | j \text{ is reachable from } i \text{ in } \mathcal{G}_d\}$.

Based on these considerations we propose an idea which links the results in this chapter together. We are not going to formally work out the details. Instead, our main objective here is to motivate one way in which the results from the construction and investigation of the graph can be put to further use. The definitive approach has to be devised for each case individually. In particular it will in general require careful consideration about what parameter d is to be chosen in order to try to avoid the case where the graph criterion fails. Having decided upon a parameter d for a given interaction system *Sys* and a subset $\mathcal{A}_0$ of ports, we first compute $\mathcal{G}_d$ for *Sys* and $\mathcal{A}_0$. If Proposition 4.3.2 yields progress of $\mathcal{A}_0$ we are done. Otherwise we go on with checking progress of $r_d(0)$ directly: We compute $Sys_{\overline{\mathcal{A}_{r_d(0)}}}$ as defined in Section 4.2 and check the condition stated in Theorem 4.2.1. The outcome of this check yields a *definitive* answer to the question whether $\mathcal{A}_{r_d(0)}$ makes progress. Together with the lemma above we therefore have also *decided* whether $\mathcal{A}_0$ itself makes progress. In particular, if the condition stated in the theorem is not satisfied we know that $\mathcal{A}_0$ does *not* make progress.

However, this seemingly neat result has to be somewhat put into perspective. It is not clear whether it is possible to check the condition in the allocated time. As mentioned in Section 4.2 checking the conditions of the characterization cannot be performed efficiently in general. The point that we want to stress here is that by checking progress of $\mathcal{A}_{r_d(0)}$ we incorporate the information obtained from the graph. This may result in an effectively smaller system to be investigated. Therefore we follow the objective of the corresponding paradigm stated in Section 1.2.2 instead of starting from scratch again. Nonetheless, the following lemma in a sense shows that even

by following this approach and by using the information gathered we have to expect that — if $\mathcal{A}_0$ indeed makes progress in Sys — the remaining effort to check the conditions in the characterization of progress for $\mathcal{A}_{r_d(0)}$ is *at least* of the complexity that was necessary for constructing the graph.

Lemma 4.3.8. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d \leq n$ and let $\mathcal{G}_d$ be defined as above. Let $K \not\subseteq r_d(0)$.*

If $Sys_{\overline{\mathcal{A}_{r_d(0)}}}$ does not contain any cycles then $|K_{\overline{\mathcal{A}_{r_d(0)}}}| > d$.

The lemma shows that for the case where no further investigation is necessary[2] we must analyze a system whose size is greater than the size of the subsystems that we considered during the construction of $\mathcal{G}_d$.

4.4 Example — The Dining Philosophers

We return to Sys_{phil_m} as introduced in Chapter 2 (cf. pp. 22, 26, and 30). We use Proposition 4.3.2 to formally prove that philosopher p_0 makes progress. An algorithm based on the proposition is in development. Here, we will explain why the results are applicable to Sys_{phil_m}. We discuss an interesting fact regarding the comparison of the progress graph of a certain stage d and its extension.

First, we point out some details with respect to the construction of the progress graph for Sys_{phil_m}. When we actually depict the progress graph we carry out the following simplifications that increase readability. We consider the case $m = 3$, and we will omit the node 0. Since we are interested in the case $\mathcal{A}_0 = \mathcal{A}_{p_0}$ we know that there is an edge $(0, p_0)$ in $\mathcal{E}_d$. It will therefore suffice to show that all components are reachable from p_0.

The first stage progress graph for Sys_{phil_3} is depicted in Figure 4.7. It cannot be used to show that p_0 makes progress in Sys_{phil_3}. The only components that are reachable from p_0 are the other two philosophers. The reason that there is no edge (p_0, c) can be explained as follows. The ports *enter* and *leave* of c also occur in interactions involving philosophers other than

[2] If $Sys_{\overline{\mathcal{A}_{r_d(0)}}}$ contains cycles we still have to check the condition about reachability stated in Theorem 4.2.1 for the states occurring on the cycle.

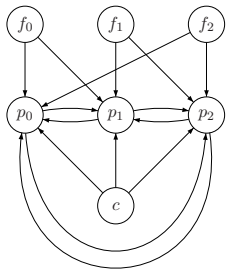

Figure 4.7: $\mathcal{G}_1^1$ for Sys_{phil_3}

p_0. Thus, they are not in $\mathcal{A}_c \backslash excl_c(\mathcal{A}_{p_0})$. Therefore, this set is empty, and it cannot be inevitable in T_c. The same way it can be seen that there are no edges (p_0, f_0) and (p_0, f_1) (it is clear that $(p_0, f_2) \notin \mathcal{E}_1^1$ because these two components do not interact).

In the second stage of the progress graph for all philosophers p_j the edge (p_j, f_j) is added. This can be seen by choosing $K_2 = \{f_j, p_{j-1}\}$, i.e., we pair the fork with the *other* philosopher that uses it. We get $Int \downarrow_{K_2} \backslash excl(\mathcal{A}_{p_j}) \downarrow_{K_2} = \{\{activate_{j-1}\}, \{get_j\}, \{put_j\}\}$. Every cycle in $Sys \downarrow_{K_2}$ involves an edge labeled with one of these interactions. Similarly, it can be seen that for every p_j the edge (p_j, f_{j+1}) is added. Thus, incrementing the considered stage of the graph by one yields the further information that besides all philosophers, also all forks are reachable from p_0. Having said that, also for $\mathcal{E}_2^1$ (if $m > 2$) there are no edges of the form (i, c) for any component $i \in K_{phil_m}$, though. No matter which subset $\{j, c\} \subseteq K_{phil_m}$ consisting of c and some other component j is considered, the set $excl(\mathcal{A}_i) \downarrow_{K_2}$ contains the two interactions $\{leave\}$ and $\{enter\}$. In $Sys \downarrow_{K_2}$ there are cycles only involving these two labels showing that $Int \downarrow_{K_2} \backslash excl(\mathcal{A}_i) \downarrow_{K_2}$ is not inevitable with respect to c in $Sys \downarrow_{K_2}$ and therefore there cannot be any edge of the form (i, c).

This problem causing the nonexistence of an edge (i, c) for any $i \in K_{phil_m}$ persists for any stage d of the graph where $d < m$. This is because for any such d and any set K_d containing c there will be at least one philosopher p_k with $i \neq p_k$ which is not in K_d. This means that again there are interactions $\{leave\}$ and $\{enter\}$ in $excl(\mathcal{A}_i) \downarrow_{K_2}$ and we get the same kind of cycles as above. Only by choosing $d \geq m$, it is possible to obtain edges of the form

$(i, c) \in \mathcal{E}_d^1$, in particular the edge (p_0, c). For example, for $d = m$ we choose $K_m = \{c, p_1, \ldots, p_{m-1}\}$. It can be seen that for this choice of K_m the set $\mathit{Int} \downarrow_{K_m} \setminus \mathit{excl}(\mathcal{A}_{p_0}) \downarrow_{K_m}$ is inevitable with respect to c in $\mathit{Sys} \downarrow_{K_m}$. Therefore, the m-th stage progress graph can be consulted in order to show that philosopher p_0 makes progress in $\mathit{Sys}_{\mathit{phil}_m}$ because in this case at last all components are reachable from p_0.

Of course, this result is not satisfactory because the computational check of the argument above would require the analysis of subsystems whose size is exponential in m, which parametrizes the number of philosophers. Clearly, any procedure exhibiting a complexity which is exponential in an input parameter is not practical. This statement directly leads to the extended progress graph defined in Section 4.3.2. We have seen above that for every philosopher p_j the set $r_1^1(p_j)$ of components that are reachable from p_j in $\mathcal{G}_1^1$ is $\{p_0, \ldots, p_{m-1}\}$. Thus, $\mathit{excl}(\mathcal{A}_{r_1^1(p_j)})$ is empty because every interaction involves at least one philosopher, and for any component k which does not represent a philosopher an edge (p_j, k) is introduced in $\mathcal{E}_1^2$ because we have $\mathcal{A}_k \setminus \mathit{excl}_k(\mathcal{A}_{r_1^1(p_j)}) = \mathcal{A}_k$. $\mathcal{A}_k$ is obviously inevitable in T_k. After only *one* iteration of the construction of the edges of the extended *first* stage progress graph we may conclude that p_0 makes progress in $\mathit{Sys}_{\mathit{phil}_m}$.

The extended first stage progress graph is depicted in Figure 4.8. The edges added after the first iteration are depicted as dashed lines. We only depicted those new edges starting in p_0 that lead to nodes that were not reachable from p_0 in $\mathcal{G}_1^1$. All components are reachable from p_0. According to the remark from the beginning of this section, all components are therefore also reachable from 0.

We conclude this example by a short remark regarding an issue that has not been discussed yet, but is brought up by the explanations above. We have already mentioned that increasing the parameter d yields a more general criterion at the cost of an increase in complexity. Using Lemmas 4.3.3 and 4.3.6, to compare the upper bounds for the complexity of constructing $\mathcal{G}_d^1$ respectively $\mathcal{G}_d$ we see that passing on to the extended progress graph increases the degree of the polynomial that bounds the complexity of the construction by 2 *without* actually considering any subsystem of size greater than d. One might therefore argue that the additional effort for constructing

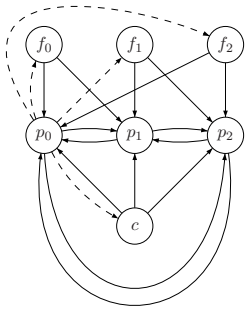

Figure 4.8: $\mathcal{G}_1$ for Sys_{phil_3} after one iteration of the construction of the edges

the extended d-th stage progress graph is not justified because it might be possible to get more precise information from increasing d itself (i.e., one could hide two components less by considering subsystems of size $d+2$) instead of constructing the extended progress graph of stage d. The discussion of Sys_{phil_m} above shows that this statement is *not* tenable. In order to show that p_0 makes progress using $\mathcal{G}_d^1$ we had to pick $d \geq m$. On the other hand, already $\mathcal{G}_1$ is sufficient to show that the philosopher makes progress.

This example demonstrates that it is not necessarily the best solution to always increase the stage of the progress graph when progress of a given set of ports is to be shown. It is not possible to answer in general the question which stage of the (extended) progress graph is best suited (in the sense that it conveys the necessary information at the lowest possible complexity) to test progress in a concrete system. This question has to be answered depending on the considered example on the one hand but also on complexity that we are able to handle during the construction of the graph.

4.5 Conclusion and Related Work

4.5.1 Conclusion and Discussion

We investigated progress of a set $\mathcal{A}_0$ of ports. We started by giving a characterization of the sets $\mathcal{A}_0$ that make progress in a given interaction system. Recalling the PSPACE-completeness results of deciding progress, it is clear that this characterization cannot constitute a condition which is ver-

ifiable efficiently, in general. Nonetheless, we explained that it might come in handy in some situations where checking the conditions of the original definition would be more complex. Next, we presented a graph criterion stating a sufficient condition for progress. The quality of the graph proved to be the fact that it can be constructed *automatically* and *efficiently*. We defined the graph such that it is possible to adjust the exactness of the criterion based on the graph according to a parameter d describing the size of the subsystems that have to be investigated for the construction of the graph. By means of an example we motivated a possible extension of the graph. This extension included information gathered from the graph constructed so far into its further construction. We rounded the results off by suggesting one approach for dealing with situations where the criterion fails and therefore does not allow any statement about progress. We proposed to combine the information obtained from the graph with the characterization of progress. This way it is possible to hope for a definite answer to the question whether $\mathcal{A}_0$ makes progress. Further, we do not discard the information we gained from the graph. We applied our results to Sys_{phil_m} showing that every philosopher makes progress. On the basis of this example, we pointed out that it is not necessarily the best approach to increase the parameter d in the definition of the progress graph. Using the extended progress graph of a lower stage may yield the same information as the progress graph for a large d at a lower cost. However, a general statement about which graph of which stage is best suited for which problem was not found.

4.5.2 Related Work

There are scores of works that investigate progress(-like properties) or more generally liveness properties in concurrent systems. Various formalisms have been used and the issues discussed include adequate temporal logics, fairness, and a variety of proof techniques for these properties. These include proof rules which are compositional with respect to parallel composition [121] and can be extended to form proof lattices for various liveness properties [113], arguments using Büchi-automata [10], translation of liveness checking into safety checking (which has been studied more thoroughly)

[34], and the discussion of implementation relations according to certain preorders [65, 66]. Due to the great variety in the formalisms used and in the approaches taken towards proving liveness properties (and even in the perception of what a liveness property is, as mentioned in Section 2.3) it is difficult to directly relate our results with respect to progress in interaction systems to these works. It should be noted, though, that various of the *liveness manifestos* issued at the *Beyond Safety International Workshop* [1] emphasize that checking liveness in finite systems always involves cycle detection[3] in some way. We also encountered this issue in the characterization of progress (Theorem 4.2.1) as well as in the construction of the progress graph in Section 4.3. As for the construction of the graph, it is worth noting that we shifted this cycle detection to subsystems and therefore reduced its complexity. We mention the closed covers technique presented by Gouda [75] as one example of an approach that also tries to prove a progress property in a system consisting of two communicating finite state machines[4] by formulating conditions that have to hold for cycles in each of the single state machines instead of checking cycles of the composed system. The hope to be able to apply such techniques of shifting cycle detection to subsystems to other formalisms which provide for suitable notions of compositionality and of subsystem does not seem to be illusory at all. In particular, such ideas could be used as a preprocessing to model checking, either making model checking unnecessary or at least rendering it more efficient by combining it with information derived from $\mathcal{G}_d$.

4.6 Proofs

4.6.1 Proofs for Section 4.2

Theorem 4.2.1. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports.*

[3] Interestingly, some manifestos claim that liveness is obsolete because in practice one is only interested in bounded liveness properties which are in fact safety properties. Others argue against this position. A discussion of these issues is out of the scope of this thesis.

[4] The technique is generalizable to an arbitrary number of state machines.

$\mathcal{A}_0$ makes progress in Sys if and only if $T_{\overline{\mathcal{A}_0}}$ does not contain any cycles visiting a state $\overline{q}$ for which there exists $q' \in \prod_{i \in K \setminus K_{\overline{\mathcal{A}_0}}} Q_i$ such that $(\overline{q}, q')$ is in reach(Sys).

Proof. We will show both directions:

$\underline{\Rightarrow}$: Assume that $\mathcal{A}_0$ makes progress in *Sys* but $T_{\overline{\mathcal{A}_0}}$ contains a cycle visiting a state $\overline{q}$ as above. In detail, let

$$\overline{q} \overset{\alpha_0}{\to} \overline{q}^1 \overset{\alpha_1}{\to} \ldots \overset{\alpha_m}{\to} \overline{q}$$

constitute such a cycle in $T_{\overline{\mathcal{A}_0}}$ where $\alpha_l \in \mathit{excl}(\mathcal{A}_0)$ for all $0 \leq l \leq m$. Choose $q' \in \prod_{i \in K \setminus K_{\overline{\mathcal{A}_0}}} Q_i$ such that $(\bar{q}, q')$ is reachable in *Sys*. No component in $K \setminus K_{\overline{\mathcal{A}_0}}$ participates in any α_l for $0 \leq l \leq m$. Therefore

$$(\overline{q}, q') \overset{\alpha_0}{\to} (\overline{q}^1, q') \overset{\alpha_1}{\to} \ldots \overset{\alpha_m}{\to} (\overline{q}, q') \overset{\alpha_0}{\to} (\overline{q}^1, q') \overset{\alpha_1}{\to} \ldots$$

yields a run in *Sys* which starts in a reachable state but is only labeled with interactions in $\mathit{excl}(\mathcal{A}_0)$. This shows that $\mathcal{A}_0$ does not make progress in *Sys* which is a contradiction.

$\underline{\Leftarrow}$: Assume that in $T_{\overline{\mathcal{A}_0}}$ there is no cycle as above. We want to show that $\mathcal{A}_0$ makes progress in *Sys*. Assume that this is not the case. Then there must be a run

$$\sigma = q \overset{\alpha_0}{\to} q^1 \overset{\alpha_1}{\to} \ldots$$

in *Sys* such that q is reachable and no α_l involves any port in $\mathcal{A}_0$. This means that every α_l is in $\mathit{excl}(\mathcal{A}_0)$. By deleting all local states q_i^l with $i \in K \setminus K_{\overline{\mathcal{A}_0}}$ we get an infinite sequence $\sigma_{\overline{\mathcal{A}_0}}$ in $T_{\overline{\mathcal{A}_0}}$ (in particular, the labels are in $\mathit{excl}(\mathcal{A}_0)$) such that all $\overline{q}^l$ on $\sigma_{\overline{\mathcal{A}_0}}$ have the property described in the theorem. Because $T_{\overline{\mathcal{A}_0}}$ is finite it is clear that $\sigma_{\overline{\mathcal{A}_0}}$ contains a cycle. This is a contradiction.

□

4.6.2 Proofs for Section 4.3.2

For better understanding we took the following approach in Section 4.3: We first introduced the more specific results based on $\mathcal{G}_d^1$. Then we used

$\mathcal{G}_d^1$ to motivate its extension and to obtain more general statements. As for the proofs it makes more sense to treat the results from Section 4.3.2 first because the other results can then be formulated as corollaries.

We need the following lemma for the proof of Lemma 4.3.5. It formalizes the argument that followed Definition 4.3.2. Roughly speaking, it states that if $\alpha' \in \mathit{Int}\downarrow_{K'}$ and $\alpha \in \mathit{Int}$ are interactions of $\mathit{Sys}\downarrow_{K'}$ respectively of Sys with $\alpha' \in \mathit{Int}\downarrow_{K'} \setminus \mathit{excl}(\mathcal{A}_0)\downarrow_{K'}$ and $\alpha\downarrow_{K'} = \alpha'$ then $\alpha \cap \mathcal{A}_0 \neq \emptyset$.

Lemma 4.6.1. *Let Sys be an interaction system, let $\mathit{Int}' \subseteq \mathit{Int}$, and let $K' \subseteq K$ be a subset of components. Further, let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports.*

We have:

$$\mathit{Int}'\downarrow_{K'} \setminus \mathit{excl}(\mathcal{A}_0)\downarrow_{K'} =$$

$$\{\alpha' \in \mathit{Int}'\downarrow_{K'} \mid \forall \alpha \in \mathit{Int} : \alpha\downarrow_{K'} = \alpha' \Rightarrow \alpha \cap \mathcal{A}_0 \neq \emptyset\}$$

Proof. By definition $\mathit{excl}(\mathcal{A}_0) = \{\alpha \in \mathit{Int} | \alpha \cap \mathcal{A}_0 = \emptyset\}$ is the set of interactions that do not involve any port in $\mathcal{A}_0$. Projecting this set to K', we obtain $\mathit{excl}(\mathcal{A}_0)\downarrow_{K'}$ the set of interactions α' of $\mathit{Sys}\downarrow_{K'}$ such that there is at least one $\alpha \in \mathit{excl}(\mathcal{A}_0)$ with $\alpha\downarrow_{K'} = \alpha'$. Therefore:

$$\begin{aligned}
\mathit{Int}'\downarrow_{K'} \setminus \mathit{excl}(\mathcal{A}_0)\downarrow_{K'} &= \{\alpha' \in \mathit{Int}'\downarrow_{K'} \mid \neg(\exists \alpha \in \mathit{Int} \text{ with } \alpha \in \mathit{excl}(\mathcal{A}_0) \\
&\qquad \text{and } \alpha\downarrow_{K'} = \alpha')\} \\
&= \{\alpha' \in \mathit{Int}'\downarrow_{K'} \mid \forall \alpha \in \mathit{Int} : \alpha\downarrow_{K'} \neq \alpha' \\
&\qquad \text{or } \alpha \notin \mathit{excl}(\mathcal{A}_0)\} \\
&= \{\alpha' \in \mathit{Int}'\downarrow_{K'} \mid \forall \alpha \in \mathit{Int} : \alpha\downarrow_{K'} = \alpha' \Rightarrow \\
&\qquad \alpha \cap \mathcal{A}_0 \neq \emptyset\}
\end{aligned}$$

□

We move on with the proof of Lemma 4.3.5.

Lemma 4.3.5. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports.*

1. *For $m \geq 1$ we have*

$$\mathcal{E}_1^m := \left\{(i,j) \mid j \neq i, 0 \text{ and } \mathcal{A}_j \setminus \mathit{excl}_j(\mathcal{A}_{r_1^{m-1}(i)}) \text{ is inevitable in } T_j\right\}.$$

2. *Let* $1 \leq d' < d \leq n$ *and* $m \geq 1$ *and let* $i \in K \cup \{0\}$ *and* $j \in K$. *If* $(i,j) \in \mathcal{E}^m_{d'}$ *then we also have* $(i,j) \in \mathcal{E}^m_d$.

Proof. We treat the statements separately.

1. For any subset $Int' \subseteq Int$ of interactions we can identify $Int' \downarrow_{\{j\}} = \{\{a_j\} | a_j \in \mathcal{A}_j \wedge \exists \alpha \in Int' : a_j \in \alpha\}$ with $Int'_j = \{a_j \in \mathcal{A}_j | \exists \alpha \in Int' : a_j \in \alpha\}$. Further, in $T_{Sys\downarrow_{\{j\}}}$ there is a transition $(q_j) \stackrel{\{a_j\}}{\rightarrow} (q_j)$ if and only if in T_j there is a transition $q_j \stackrel{a_j}{\rightarrow} q_j$.

   Consider a cycle in $Sys \downarrow_{\{j\}}$. All labels occurring on this cycle involve j. Thus, the precondition of the implication in the definition of inevitability of $Int \downarrow_{\{j\}} \backslash excl(\mathcal{A}_{r_1^{m-1}(i)}) \downarrow_{\{j\}}$ with respect to j in $Sys \downarrow_{\{j\}}$ is always satisfied. Therefore $Int \downarrow_{\{j\}} \backslash excl(\mathcal{A}_{r_1^{m-1}(i)}) \downarrow_{\{j\}}$ is inevitable with respect to j in $Sys \downarrow_{\{j\}}$ if and only if it is inevitable in $Sys \downarrow_{\{j\}}$.

   Identifying $Int \downarrow_{\{j\}}$ with $\mathcal{A}_j$, $excl(\mathcal{A}_{r_1^{m-1}(i)}) \downarrow_{\{j\}}$ with $excl_j(\mathcal{A}_{r_1^{m-1}(i)})$, and $T_{Sys\downarrow_{\{j\}}}$ with T_j as above, we conclude that $\mathcal{A}_j \backslash excl_j(\mathcal{A}_{r_1^{m-1}}(i))$ is inevitable in T_j if and only if $Int \downarrow_{\{j\}} \backslash excl(\mathcal{A}_{r_1^{m-1}(i)}) \downarrow_{\{j\}}$ is inevitable with respect to j in $Sys \downarrow_{\{j\}}$. The statement of the lemma follows.

2. Fix d' and d with $d' < d$. We use induction over m:

   $\underline{m = 1:}$ We have $m - 1 = 0$. Therefore $r_{d'}^{m-1}(i) = r_d^{m-1}(i) = \{i\}$ for all $i \in \mathcal{V}$ because no edges have been constructed so far.

   Now assume $(i,j) \in \mathcal{E}^1_{d'}$ but $(i,j) \notin \mathcal{E}^1_d$. Choose a subset $K_{d'}$ of d' components with $j \in K_{d'}$ such that $Int \downarrow_{K_{d'}} \backslash excl(\mathcal{A}_i) \downarrow_{K_{d'}}$ is inevitable with respect to j in $Sys \downarrow_{K_{d'}}$. On the other hand, $Int \downarrow_{K_d} \backslash excl(\mathcal{A}_i) \downarrow_{K_d}$ is not inevitable with respect to j in $Sys \downarrow_{K_d}$ for all subsets K_d of d components with $j \in K_d$. Choose K_d with $K_{d'} \subset K_d$. In $Sys \downarrow_{K_d}$ there is a cycle σ_d such that:

   (a) j participates in at least one interaction on σ_d

   (b) no interaction on σ_d is in $Int \downarrow_{K_d} \backslash excl(\mathcal{A}_i) \downarrow_{K_d}$

   Projecting σ_d to $K_{d'}$ we write $\sigma_{d'} := \sigma_d \downarrow_{K_{d'}}$. Since $j \in K_{d'}$ we obtain a cycle $\sigma_{d'}$ in $Sys \downarrow_{K_{d'}}$ that j participates in. Because $Int \downarrow_{K_{d'}} \backslash excl(\mathcal{A}_i) \downarrow_{K_{d'}}$ is inevitable with respect to j in $Sys \downarrow_{K_{d'}}$ there is an interaction $\alpha_{d'} \in Int \downarrow_{K_{d'}} \backslash excl(\mathcal{A}_i) \downarrow_{K_{d'}}$ on $\sigma_{d'}$. Let α_d

be the interaction on σ_d with $\alpha_d \downarrow_{K_{d'}} = \alpha_{d'}$. We have $\alpha_d \notin \mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_i) \downarrow_{K_d}$. According to Lemma 4.6.1 there is an interaction α with $\alpha \downarrow_{K_d} = \alpha_d$ and $\alpha \cap \mathcal{A}_i = \emptyset$. On the other hand, we also have $\alpha \downarrow_{K_{d'}} = \alpha_{d'}$ because $\alpha_d \downarrow_{K_{d'}} = \alpha_{d'}$. Using Lemma 4.6.1 again this shows $\alpha \cap \mathcal{A}_i \neq \emptyset$ because $\alpha_{d'} \in \mathit{Int} \downarrow_{K_{d'}} \setminus \mathit{excl}(\mathcal{A}_i) \downarrow_{K_{d'}}$. This is a contradiction. We conclude $(i,j) \in \mathcal{E}_d^1$.

$\underline{m \Rightarrow m+1:}$ Assume that $(i,j) \in \mathcal{E}_{d'}^{m+1}$ but $(i,j) \notin \mathcal{E}_d^{m+1}$. We proceed as above: Choose $K_{d'}$ with $j \in K_{d'}$ such that $\mathit{Int} \downarrow_{K_{d'}} \setminus \mathit{excl}(\mathcal{A}_{r_{d'}^m(i)}) \downarrow_{K_{d'}}$ is inevitable with respect to j in $\mathit{Sys} \downarrow_{K_{d'}}$ and K_d with $K_{d'} \subset K_d$. Because $(i,j) \notin \mathcal{E}_d^{m+1}$ we know that $\mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_{r_d^m(i)}) \downarrow_{K_d}$ is not inevitable with respect to j in $\mathit{Sys} \downarrow_{K_d}$. Following the same line of argument as above we find $\alpha_{d'} \in \mathit{Int} \downarrow_{K_{d'}} \setminus \mathit{excl}(\mathcal{A}_{r_{d'}^m(i)}) \downarrow_{K_{d'}}$ and $\alpha_d \in \mathit{Int} \downarrow_{K_d}$ such that $\alpha_d \downarrow_{K_{d'}} = \alpha_{d'}$ and $\alpha_d \notin \mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_{r_d^m(i)}) \downarrow_{K_d}$. According to Lemma 4.6.1 there is $\alpha \in \mathit{Int}$ with $\alpha \downarrow_{K_d} = \alpha_d$ and $\alpha \cap \mathcal{A}_{r_d^m(i)} = \emptyset$. From the induction hypothesis we know $\mathcal{E}_{d'}^l \subseteq \mathcal{E}_d^l$ for all $1 \leq l \leq m$. This implies $r_{d'}^m(i) \subseteq r_d^m(i)$ and therefore also $\mathcal{A}_{r_{d'}^m(i)} \subseteq \mathcal{A}_{r_d^m(i)}$. Together with $\alpha \cap \mathcal{A}_{r_d^m(i)} = \emptyset$ we get $\alpha \cap \mathcal{A}_{r_{d'}^m(i)} = \emptyset$. As above, we also have $\alpha \downarrow_{K_{d'}} = \alpha_{d'}$. Lemma 4.6.1 together with $\alpha_{d'} \in \mathit{Int} \downarrow_{K_{d'}} \setminus \mathit{excl}(\mathcal{A}_{r_{d'}^m(i)}) \downarrow_{K_{d'}}$ implies $\alpha \cap \mathcal{A}_{r_{d'}^m(i)} \neq \emptyset$. This is a contradiction. We conclude $(i,j) \in \mathcal{E}_d^{m+1}$.

□

The proof of Lemma 4.3.6 needs some preliminaries. For the construction of $\mathcal{E}_d$ we resort to Algorithm 1 below that has already been discussed in Section 4.3.1. It can be used to check whether a set of interactions is inevitable with respect to a component in a given interaction system. We formulate a more general algorithm that decides inevitability of $\mathcal{L}'$ with respect to $\mathcal{L}''$ in a finite labeled transition system T with label set $\mathcal{L}$.

Lemma 4.6.2. *Algorithm 1 is correct. It can be realized such that it terminates in time linear in the product of the size of T and the size of $\mathcal{L}$.*

Proof. For denotational purposes we introduce $T^{\mathcal{L}\setminus\mathcal{L}'}$ to identify the transition system obtained by removing all edges whose labels are in $\mathcal{L}'$ from T.

Algorithm 1 INEVITABILITY$(T, \mathcal{L}', \mathcal{L}'')$

Input: a finite labeled transition system $T = (Q, \mathcal{L}, \rightarrow)$ and $\mathcal{L}', \mathcal{L}'' \subseteq \mathcal{L}$

Output: **true** if $\mathcal{L}'$ is inevitable with respect to $\mathcal{L}''$ in T, **false** otherwise

1: remove all edges whose label is in $\mathcal{L}'$ from T
2: **if** T does not contain any cycles **then**
3: **return true**
4: **else**
5: compute the strongly connected components of T
6: **if** no strongly connected component contains an edge whose label is in $\mathcal{L}''$ **then**
7: **return true**
8: **return false**

This means that after line 1 the algorithm operates on $T^{\mathcal{L}\setminus\mathcal{L}'}$. We first show that the algorithm is correct.

Assume that the algorithm returns **true**. There are two cases:

1. It returns **true** in line 3. Then $T^{\mathcal{L}\setminus\mathcal{L}'}$ does not contain any cycles. Thus, every cycle in T contained at least one of the edges that were removed. All labels of these edges are in $\mathcal{L}'$ and therefore $\mathcal{L}'$ is inevitable with respect to $\mathcal{L}''$ in T (in fact $\mathcal{L}'$ is even inevitable in T).

2. It returns **true** in line 7. It is easy to see that in $T^{\mathcal{L}\setminus\mathcal{L}'}$ a strongly connected component containing an edge whose label is in $\mathcal{L}''$ exists if and only if a cycle containing such an edge exists in $T^{\mathcal{L}\setminus\mathcal{L}'}$. The algorithm only reaches line 7 if $T^{\mathcal{L}\setminus\mathcal{L}'}$ still contains cycles but no strongly connected components that contain an edge whose label is in $\mathcal{L}''$. The statement above shows that in this case none of the cycles in $T^{\mathcal{L}\setminus\mathcal{L}'}$ contain an edge whose label is in $\mathcal{L}''$. Therefore all the cycles in T that correspond to a cycle in $T^{\mathcal{L}\setminus\mathcal{L}'}$ satisfy the condition in the definition of inevitability with respect to $\mathcal{L}''$ because they do not contain any label in $\mathcal{L}''$. All other cycles of T have been eliminated by removing an edge whose label is in $\mathcal{L}'$. As above we see that these cycles also satisfy the condition for inevitability of $\mathcal{L}'$ with respect to $\mathcal{L}''$.

Thus, $\mathcal{L}'$ is inevitable with respect to $\mathcal{L}''$ in T if the algorithm returns **true**.

Now assume that $\mathcal{L}'$ is inevitable with respect to $\mathcal{L}''$ in T. Again there are two cases:

1. All cycles in T contain at least one edge whose label is in $\mathcal{L}'$. Since all these edges are removed in the first line of the algorithm there cannot be any cycles left in $T^{\mathcal{L}\setminus\mathcal{L}'}$. The algorithm returns **true** in line 3.

2. All cycles in T that do not contain an edge whose label is in $\mathcal{L}'$ do not involve an edge whose label is in $\mathcal{L}''$. Then all cycles that are left in $T^{\mathcal{L}\setminus\mathcal{L}'}$ do not involve an edge whose label is in $\mathcal{L}''$. The fact stated above shows that in this case there also is no strongly connected component containing an edge whose label is in $\mathcal{L}''$. The algorithm returns **true** in line 7.

The algorithm returns **true** if $\mathcal{L}'$ is inevitable with respect to $\mathcal{L}''$ in T. Correctness of Algorithm 1 follows.

It is clear that the algorithm terminates. Removing all edges whose label is in $\mathcal{L}'$ in the first line can be done in $\mathcal{O}(|T|\,|\mathcal{L}'|) = \mathcal{O}(|T||\mathcal{L}|)$. Deciding the existence of a cycle in a directed graph can be done in time linear in sum of the number of nodes and edges using a depth-first-search that keeps track of backward edges, for example. Thus, the cost of the **if**-block in the second line is in $\mathcal{O}(|T^{\mathcal{L}\setminus\mathcal{L}'}|) = \mathcal{O}(|T|)$. The same time bounds can be realized for the computation of the strongly connected components of a directed graph using Tarjan's algorithm [132], for example. Finally, checking whether there is a strongly connected component containing an edge whose label is in $\mathcal{L}''$ can be realized in time $\mathcal{O}(|T^{\mathcal{L}\setminus\mathcal{L}'}||\mathcal{L}''|) = \mathcal{O}(|T||\mathcal{L}|)$. Altogether we see that the algorithm can be realized in $\mathcal{O}(|T|\,|\mathcal{L}|)$. □

Lemma 4.3.6. *Let Sys be an interaction system and let* $1 \leq d \leq n$. *Let* $\mathcal{A}_0 \subseteq \mathcal{A}$ *be a set of ports.*

The extended d-th stage progress graph for Sys and $\mathcal{A}_0$ *can be constructed in time polynomial in the size of Sys. An upper bound for the cost of the construction is given by*

$$\mathcal{O}(|Int|^2 n^{d+4} + |Int| n^{d+3} |T_{max}|^d)$$

where $|T_{max}|$ *denotes the size of the largest local transition system.*

Proof. We do not make any assumptions about the underlying implementation of interaction systems. Depending on the implementation the following computations may be performed more efficiently than the upper bounds that we derive. Since we are only interested in such an upper bound (and in the fact that it is polynomial in the size of *Sys*) this suffices for our purposes.

We first consider the complexity of constructing $\mathcal{E}_d^m$ for $m \geq 1$ fixed. We have to find $r_d^{m-1}(i)$ for every node $i \in \mathcal{V}$, i.e., for every i we have to perform a reachability analysis in the progress graph constructed so far. A reachability analysis in a directed graph $G = (V, E)$ can be performed in $\mathcal{O}(|E| + |V|)$. $|E|$ is in $\mathcal{O}(n^2)$. Therefore the computation of $r_d^{m-1}(i)$ for every i can be done in $\mathcal{O}(n^3)$.

Having found $r_d^{m-1}(i)$, next for all i we compute $\mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)})$ once and store it for later reference. For i fixed we have to check for every interaction $\alpha \in \mathit{Int}$ whether $\alpha \cap \mathcal{A}_{r_d^{m-1}(i)} = \emptyset$. Fix α. Since α contains at most n ports this can be realized in $\mathcal{O}(n|\mathcal{A}_{r_d^{m-1}(i)}|) = \mathcal{O}(n|\mathcal{A}|)$. Thus, the cost for the computation of $\mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)})$ is bounded by $\mathcal{O}(|\mathit{Int}|n|\mathcal{A}|)$. We have these costs for every $i \in \mathcal{V}$ and therefore altogether this step causes cost $\mathcal{O}(n^2|\mathit{Int}||\mathcal{A}|)$.

Now fix $i \in \mathcal{V}$ and $j \in \mathcal{V}\backslash\{i, 0\}$ and consider the cost for checking whether $(i, j) \in \mathcal{E}_d^m$. We have to check whether there is a subset K_d of d components with $j \in K_d$ such that $\mathit{Int} \downarrow_{K_d} \backslash \mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d}$ is inevitable with respect to j in $\mathit{Sys} \downarrow_{K_d}$. There are $\binom{n-1}{d-1}$ possible choices for K_d. Fix such a set K_d. We first have to compute $\mathit{Int} \downarrow_{K_d} \backslash \mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d}$. Projecting an interaction α to K_d can be done in $\mathcal{O}(n)$ because α contains at most n elements. Computing $\mathit{Int} \downarrow_{K_d}$, we get this cost for every interaction α. We further have to check for every α whether $\alpha \downarrow_{K_d}$ is already contained in the set of interactions projected so far. The size of this set is bounded by $|\mathit{Int}|$. Thus, $\mathit{Int} \downarrow_{K_d}$ can be computed in time $\mathcal{O}(|\mathit{Int}|n|\mathit{Int}|) = \mathcal{O}(n|\mathit{Int}|^2)$. The same argument and the fact that $|\mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)})| = \mathcal{O}(|\mathit{Int}|)$ show that $\mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d}$ can also be found in time $\mathcal{O}(n|\mathit{Int}|^2)$. The complexity of computing the complement of $\mathit{Int} \downarrow_{K_d}$ with respect to $\mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d}$ is bounded by $\mathcal{O}(|\mathit{Int} \downarrow_{K_d} ||\mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d} |) = \mathcal{O}(|\mathit{Int}|^2)$. Therefore $\mathit{Int} \downarrow_{K_d} \backslash \mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d}$ can be found in time $\mathcal{O}(n|\mathit{Int}|^2)$. Finally, Lemma 4.6.2 shows that inevitability of $\mathit{Int} \downarrow_{K_d} \backslash \mathit{excl}(\mathcal{A}_{r_d^{m-1}(i)}) \downarrow_{K_d}$ with

respect to j in $Sys \downarrow_{K_d}$ can be decided in time $\mathcal{O}(|T_{Sys\downarrow_{K_d}}||Int \downarrow_{K_d}|) = \mathcal{O}(|T_{max}|^d|Int|)$. Altogether checking whether $(i,j) \in \mathcal{E}_d^m$ can be done in time $\mathcal{O}(\binom{n-1}{d-1}(n|Int|^2 + |T_{max}|^d|Int|)) = \mathcal{O}(n^d|Int|^2 + |Int|n^{d-1}|T_{max}|^d)$.

We have to perform the check above for at most n edges of the form $(0,j)$ where $j \in K$ and for at most $n(n-1)$ edges of the form (i,j) where $i,j \in K$ but $i \neq j$. This means that we get the cost derived above at most n^2 times. Combining all the results above, we see that $\mathcal{E}_d^m$ can be found in:

$$\mathcal{O}(n^3 + n^2|Int||\mathcal{A}| + n^{d+2}|Int|^2 + |Int|n^{d+1}|T_{max}|^d)$$

The first term, i.e., n^3, is accounted for by $n^{d+2}|Int|^2$ because $d \geq 1$. Thus, we may simplify the bound to:

$$\mathcal{O}(n^2|Int||\mathcal{A}| + n^{d+2}|Int|^2 + |Int|n^{d+1}|T_{max}|^d)$$

Next we consider $|\mathcal{A}|$. $\mathcal{A}$ is given by the union of all $\mathcal{A}_i$. Therefore we can write $|\mathcal{A}| = \mathcal{O}(n|\mathcal{A}_{max}|)$ where $\mathcal{A}_{max}$ denotes the largest port set. Further, because every interaction contains at most one port of every component we have $|\mathcal{A}_{max}| = \mathcal{O}(|Int|)$. Combining these bounds, we see $|\mathcal{A}| = \mathcal{O}(n|Int|)$ and therefore $n^2|Int||\mathcal{A}| = \mathcal{O}(n^3|Int|^2)$. Using $d \geq 1$ again, we conclude that $n^2|Int||\mathcal{A}|$ is also accounted for by $n^{d+2}|Int|^2$. We finally get the bound

$$\mathcal{O}(n^{d+2}|Int|^2 + |Int|n^{d+1}|T_{max}|^d)$$

for the construction of $\mathcal{E}_d^m$.

The construction of the edges has to be iterated at most n^2 times because there can be at most n^2 edges in $\mathcal{E}_d$. Thus, the total cost for computing $\mathcal{E}_d$ is bounded $\mathcal{O}(n^2(n^{d+2}|Int|^2 + |Int|n^{d+1}|T_{max}|^d)) = \mathcal{O}(|Int|^2n^{d+4} + |Int|n^{d+3}|T_{max}|^d)$. □

Proposition 4.3.2. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d \leq n$ and let $\mathcal{G}_d$ be defined as above.*

If all nodes in $\mathcal{V}$ are reachable from 0 in $\mathcal{G}_d$ then $\mathcal{A}_0$ makes progress in Sys. The condition can be checked in time polynomial in the size of Sys.

Proof. We need some preliminaries first. Consider an arbitrary run σ. We prove the following two facts by a nested induction over $m \in \mathbb{N}$:

1. $(i,j) \in \mathcal{E}_d^m$ implies that i participates infinitely often in σ if j does so.
2. If j is reachable from i in $(\mathcal{V}, \bigcup_{l=1}^{m} \mathcal{E}_d^l)$ then i participates infinitely often in σ if j does so.

$\underline{m = 1:}$ We will show that both statements hold for $m = 1$.

1. Let $(i,j) \in \mathcal{E}_d^1$ and assume that j participates infinitely often in σ. From the definition of $\mathcal{E}_d^1$ we know that there exists a set K_d of d components with $j \in K_d$ such that $Int \downarrow_{K_d} \setminus excl\big(\mathcal{A}_i\big) \downarrow_{K_d}$ is inevitable with respect to j in $Sys \downarrow_{K_d}$. Because j participates infinitely often in σ the projection $\sigma \downarrow_{K_d}$ of σ to K_d results in a run in $Sys \downarrow_{K_d}$ which j participates an infinite number of times in. There must be a cycle π in $T_{Sys\downarrow_{K_d}}$ that contains an edge labeled with an interaction involving j such that this cycle appears infinitely often as a sub-path of $\sigma \downarrow_{K_d}$. Since $Int \downarrow_{K_d} \setminus excl\big(\mathcal{A}_i\big) \downarrow_{K_d}$ is inevitable with respect to j in $Sys \downarrow_{K_d}$ there must be at least one interaction $\alpha_d \in Int \downarrow_{K_d} \setminus excl\big(\mathcal{A}_i\big) \downarrow_{K_d}$ on π. This interaction occurs infinitely often on $\sigma \downarrow_{K_d}$ and therefore on σ an infinite number of interactions α occur with $\alpha \downarrow_{K_d} = \alpha_d$. Lemma 4.6.1 shows that for all these interactions α we have $\alpha \cap \mathcal{A}_i \neq \emptyset$, proving the first statement for $m = 1$.
2. Let j be reachable from i in $(\mathcal{V}, \mathcal{E}_d^1)$ and assume that j participates infinitely often in σ. We will prove the second statement for $m = 1$ by induction over the length s of a shortest path from i to j in $(\mathcal{V}, \mathcal{E}_d^1)$. If $s = 1$ then the path is simply an edge $(i,j) \in \mathcal{E}_d^1$. The statement then follows from the first statement for $m = 1$. Now assume that the shortest path from i to j in $(\mathcal{V}, \mathcal{E}_d^1)$ has length $s+1$ and consider such a path: $i \to \ldots \to k \to j$. Using the first statement, from $(k,j) \in \mathcal{E}_d^1$ we conclude that k participates infinitely often in σ. It is clear that $i \to \ldots \to k$ is a shortest path from i to k in $(\mathcal{V}, \mathcal{E}_d^1)$. It has length s. Since k participates infinitely often in σ the induction hypothesis about the length of the shortest path shows that i participates infinitely often in σ.

$\underline{m \Rightarrow m+1:}$ Assume that both statements hold for m.

1. Let $(i,j) \in \mathcal{E}_d^{m+1}$ and assume that j participates infinitely often in σ. The same argument (with $\mathcal{A}_{r_d^m(i)}$ replacing $\mathcal{A}_i$) that was used in the proof of the first statement for $m = 1$ is used now to show that on σ there must be an infinite number of interactions α with $\alpha \cap \mathcal{A}_{r_d^m(i)} \neq \emptyset$. Therefore there must be a node $k \in r_d^m(i)$ (i.e., k is reachable from i in $(\mathcal{V}, \bigcup_{l=1}^{m} \mathcal{E}_d^l)$) such that on σ there are infinitely many interactions α with $\alpha \cap \mathcal{A}_k \neq \emptyset$. The induction hypothesis applied to the second statement above shows that i must participate infinitely often in σ.

2. Let j be reachable from i in $(\mathcal{V}, \bigcup_{l=1}^{m+1} \mathcal{E}_d^l)$ and assume that j participates infinitely often in σ. As above we will prove the second statement by induction over the length s of a shortest path from i to j in $(\mathcal{V}, \bigcup_{l=1}^{m+1} \mathcal{E}_d^l)$. If $s = 1$ then the path is simply an edge $(i,j) \in \bigcup_{l=1}^{m+1} \mathcal{E}_d^l$. The statement then follows from the first part if $(i,j) \in \mathcal{E}_d^{m+1}$ respectively from the induction hypothesis for the first part if $(i,j) \in \bigcup_{l=1}^{m} \mathcal{E}_d^l$. Now assume that the shortest path from i to j in $(\mathcal{V}, \bigcup_{l=1}^{m+1} \mathcal{E}_d^l)$ has length $s+1$ and consider such a path: $i \to \ldots \to k \to j$. As above from $(k,j) \in \bigcup_{l=1}^{m+1} \mathcal{E}_d^l$ we conclude that k participates infinitely often in σ. Then $i \to \ldots \to k$ is a shortest path from i to k and it has length s. Since k participates infinitely often in σ the induction hypothesis about the length of a shortest path shows that i participates infinitely often in σ.

Now let j be reachable from i in $\mathcal{G}_d$ and assume that j participates infinitely often in σ. Consider a path connecting i to j in $\mathcal{G}_d$. There exists $m_0 \in \mathbb{N}$ such that all edges along the path are in $\bigcup_{l=1}^{m_0} \mathcal{E}_d^l$. Therefore j is reachable from i in $(\mathcal{V}, \bigcup_{l=1}^{m_0} \mathcal{E}_d^l)$. The second statement above implies that i also participates infinitely often in σ. The proof of the theorem is straightforward now. Indeed, let σ be a run in *Sys*. There is a component j that participates infinitely often in σ. This component is reachable from 0 and thus 0 also participates in σ (i.e., there is an interaction α on σ such that $\mathcal{A}_0 \cap \alpha \neq \emptyset$).

The condition can be checked in time polynomial: $\mathcal{G}_d$ can be constructed in time polynomial according to Lemma 4.3.6. The check of the condition

requires a reachability analysis in $\mathcal{G}_d$ which can be performed in $\mathcal{O}(n^2)$. □

4.6.3 Proofs for Section 4.3.1

The proofs for Section 4.3.1 are easy applications of the results above.

Lemma 4.3.1. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports.*

Setting $excl_j(\mathcal{A}_i) := \{a_j | \exists \alpha \in excl(\mathcal{A}_i) : a_j \in \alpha\}$, we have

$$\mathcal{E}_1^1 := \{(i,j) \,|j \neq i, 0 \text{ and } \mathcal{A}_j \backslash excl_j(\mathcal{A}_i) \text{ is inevitable in } T_j\}.$$

Proof. This directly follows from the first part of Lemma 4.3.5 with $m = 1$ (note that in this case we have $r_1^{m-1}(i) = r_1^0(i) = \{i\}$). □

Lemma 4.3.2. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d' < d \leq n$ and let $i \in K \cup \{0\}$ and $j \in K$.*

If $(i,j) \in \mathcal{E}_{d'}^1$ then we also have $(i,j) \in \mathcal{E}_d^1$.

Proof. This follows from the second part of Lemma 4.3.5 with $m = 1$. □

Lemma 4.3.3. *Let Sys be an interaction system and let $1 \leq d \leq n$. Let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports.*

The d-th stage progress graph for Sys and $\mathcal{A}_0$ can be constructed in time polynomial in the size of Sys. An upper bound for the cost is given by

$$\mathcal{O}(|Int|^2 n^{d+2} + |Int| n^{d+1} |T_{max}|^d)$$

where $|T_{max}|$ denotes the size of the largest local transition system.

Proof. The proof of Lemma 4.3.6 stated that $\mathcal{E}_d^m$ can be found in time $\mathcal{O}(|Int|^2 n^{d+2} + |Int| n^{d+1} |T_{max}|^d)$ for any m. Setting $m = 1$, we see that the construction of $\mathcal{E}_d^1$ can be performed in the time bound above. □

Proposition 4.3.1. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d \leq n$ and let $\mathcal{G}_d^1$ be defined as above.*

If all nodes in $\mathcal{V}$ are reachable from 0 in $\mathcal{G}_d^1$ then $\mathcal{A}_0$ makes progress in Sys. The condition can be checked in time polynomial in the size of Sys.

Proof. $\mathcal{G}_d^1$ is a subgraph of $\mathcal{G}_d$. Thus, if all nodes are reachable from 0 in $\mathcal{G}_d^1$ they are also reachable from 0 in $\mathcal{G}_d$. The statement of the proposition then follows from Proposition 4.3.2.

The fact that the condition can be checked in time polynomial is shown analogously to the corresponding argument in the proof of Proposition 4.3.2 this time using Lemma 4.3.3. □

Lemma 4.3.4. *Let Sys be an interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a set of ports. Let j be a component with $\mathcal{A}_j \subseteq \mathcal{A}_0$.*

For all d there is an edge $(0, j) \in \mathcal{E}_d^1$.

Proof. Fix d and consider an arbitrary subset K_d such that $|K_d| = d$ and $j \in K_d$. Let $\alpha_d \in \mathit{Int}\downarrow_{K_d}$ be an interaction with $j(\alpha_d) \neq \emptyset$. Every $\alpha \in \mathit{Int}$ with $\alpha \downarrow_{K_d} = \alpha_d$ also involves j, and therefore we have $\alpha \notin \mathit{excl}(\mathcal{A}_0)$. Thus, $\alpha_d \notin \mathit{excl}(\mathcal{A}_0)\downarrow_{K_d}$ but $\alpha_d \in \mathit{Int}\downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_0)\downarrow_{K_d}$.

Now consider a cycle σ_d in $T_{Sys\downarrow_{K_d}}$. If there is a transition labeled with an interaction in $\mathit{Int}\downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_0)\downarrow_{K_d}$ on σ_d then the postcondition of the implication in the definition of inevitability with respect to j is satisfied for σ_d. If this is not the case then the above shows that in particular no interaction involving j occurs as a label on σ_d. Therefore in this case the precondition of the implication is satisfied for σ_d. This shows that all cycles in $T_{Sys\downarrow_{K_d}}$ satisfy the condition for inevitability of $\mathit{Int}\downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_0)\downarrow_{K_d}$ with respect to j. Thus, $(0, j) \in \mathcal{G}_d^1$. □

4.6.4 Proofs for Section 4.3.3

Lemma 4.3.7. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d \leq n$ and let $\mathcal{G}_d$ be defined as above.*

$\mathcal{A}_0$ makes progress in Sys if and only if $\mathcal{A}_{r_d(0)}$ makes progress in Sys where we write $r_d(i) := \{j \in \mathcal{V} | j \text{ is reachable from } i \text{ in } \mathcal{G}_d\}$.

Proof. If $\mathcal{A}_0$ makes progress then $\mathcal{A}_{r_d(0)}$ makes progress because $\mathcal{A}_0 \subseteq \mathcal{A}_{r_d(0)}$.

Assume that $\mathcal{A}_{r_d(0)}$ makes progress, and consider a run σ of *Sys*. Since $\mathcal{A}_{r_d(0)}$ participates in σ we can write $\sigma = \sigma'\sigma''$ such that σ' is a path in *Sys* that $\mathcal{A}_{r_d(0)}$ participates in, and σ'' is again a run. Repeating this argument for σ'' we see that $\mathcal{A}_{r_d(0)}$ participates infinitely often in σ. There must be a component $i \in r_d(0)$ that participates infinitely often in σ. The same argument that was used in the proof of Proposition 4.3.2 shows that $\mathcal{A}_0$ participates in σ. Since σ was chosen arbitrarily we see that $\mathcal{A}_0$ makes progress in *Sys*. □

Lemma 4.3.8. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0 \subseteq \mathcal{A}$ be a subset of ports. Let $1 \leq d \leq n$ and let $\mathcal{G}_d$ be defined as above. Let $K \not\subseteq r_d(0)$.*

If $Sys_{\overline{\mathcal{A}_{r_d(0)}}}$ does not contain any cycles then $|K_{\overline{\mathcal{A}_{r_d(0)}}}| > d$.

Proof. We make a general observations first. We have $r_d(0) \cap K_{\overline{\mathcal{A}_{r_d(0)}}} = \emptyset$: Let $i \in r_d(0)$ be a component and let $\alpha \in \mathit{Int}$ be an interaction with $i(\alpha) \neq \emptyset$. Then $\alpha \notin \mathit{excl}(\mathcal{A}_{r_d(0)})$, and therefore $i \notin K_{\overline{\mathcal{A}_{r_d(0)}}} = \bigcup_{\alpha \in \mathit{excl}(\mathcal{A}_{r_d(0)})} \mathit{comp}(\alpha)$.

Now we assume that $|K_{\overline{\mathcal{A}_{r_d(0)}}}| = d' \leq d$, and we show that this leads to a contradiction. We write $K_{d'} := K_{\overline{\mathcal{A}_{r_d(0)}}}$, and we choose an arbitrary subset $K_{d''}$ with $K_{d'} \cap K_{d''} = \emptyset$ and $|K_{d''}| = d - d'$. We write $K_d := K_{d'} \cup K_{d''}$. Thus, $|K_d| = d$.

Choose an arbitrary $j \in K_{d'}$. From the above we know that $j \notin r_d(0)$. In particular, there is no edge $(0, j) \in \mathcal{E}_d$, and this means that there is no l such that $\mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_{r_d^{l-1}(0)}) \downarrow_{K_d}$ is inevitable with respect to j in $\mathit{Sys} \downarrow_{K_d}$. Because there exists an index m_0 with $\mathcal{E}_d = \bigcup_{l=1}^{m_0} \mathcal{E}_d^l$ this also implies that $\mathit{Int} \downarrow_{K_d} \setminus \mathit{excl}(\mathcal{A}_{r_d(0)}) \downarrow_{K_d}$ is not inevitable with respect to j in $\mathit{Sys} \downarrow_{K_d}$. Therefore in $T_{\mathit{Sys}\downarrow_{K_d}}$ there is a cycle

$$(q_{K_{d'}}, q_{K_{d''}}) \overset{\alpha_{0,d}}{\to} (q^1_{K_{d'}}, q^1_{K_{d''}}) \overset{\alpha_{1,d}}{\to} \dots \overset{\alpha_{r,d}}{\to} (q_{K_{d'}}, q_{K_{d''}})$$

such that for all $0 \leq s \leq r$ we have $\alpha_{s,d} \in \mathit{excl}(\mathcal{A}_{r_d(0)}) \downarrow_{K_d}$ but at least one $\alpha_{s,d}$ involves j. Thus, for all s there is an interaction $\alpha_s \in \mathit{excl}(\mathcal{A}_{r_d(0)})$ with $\alpha_s \downarrow_{K_d} = \alpha_{s,d}$. Because $\alpha_s \in \mathit{excl}(\mathcal{A}_{r_d(0)})$ we have $\mathit{comp}(\alpha_s) \subseteq K_{\overline{\mathcal{A}_{r_d(0)}}} = K_{d'}$, and together with $K_{d'} \subseteq K_d$ this implies $\alpha_s \downarrow_{K_d} = \alpha_s$. We conclude $\alpha_{s,d} = \alpha_s$. Because $K_{d'} \cap K_{d''} = \emptyset$ we also get $\mathit{comp}(\alpha_s) \cap K_{d''} = \emptyset$. We can

therefore write the cycle as

$$(q_{K_{d'}}, q_{K_{d''}}) \xrightarrow{\alpha_0} (q^1_{K_{d'}}, q_{K_{d''}}) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_r} (q_{K_{d'}}, q_{K_{d''}})$$

where all α_s are in $excl(\mathcal{A}_{r_d(0)})$. Canceling $q_{K_{d''}}$ from the cycle we obtain

$$q_{K_{d'}} \xrightarrow{\alpha_0} q^1_{K_{d'}} \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_r} q_{K_{d'}}$$

which is a cycle in $Sys_{\overline{\mathcal{A}_{r_d(0)}}}$. This is a contradiction. □

Chapter 5

Robustness of Properties of Interaction Systems

5.1 Motivation

In the long run, the goal for a component model has to be applicability to real life systems. Of course, it is desirable to have theoretical results building the foundation of such a formalism, in particular results which ensure important properties of a system. The previous chapters presented various results of this kind. Despite their importance for a thorough investigation of systems and with regard to applicability of a formalism such results also have a drawback: Modeling a real life situation by means of a formalism usually requires abstraction from a lot of details and assumes a setting which is sometimes not so realistic after all. Of course, such abstraction is unavoidable because modeling a given situation in every detail is not possible, not to mention testing or deciding properties of a setting which takes into account every detail. The results that can be deduced from a theoretical setting are an important factor in the design of the system. Nonetheless, it is also an interesting question how the occurrence of events that are *not* expected during the regular operation of a system can be incorporated into the formalism without having to specify every such event explicitly. In particular, it is desirable or — depending on the application area — even imperative that a system exhibits some kind of fault tolerance with respect to failure of one or more of its components. Indeed, Cleave-

land and Smolka [53] state fault tolerance of concurrent systems as one item of a list of nonfunctional properties that are of particular importance with regard to design and applicability of concurrent systems.

We investigate how a system's behavior is influenced by the breakdown of some of its components or more generally of some of its ports. The fact that a system maintains some kind of basic functionality upon failure of ports can be interpreted such that the system respectively its properties are robust. The investigation of this particular unexpected event, i.e., a partial or complete failure of certain components, deserves special attention because the assumption that parts of a system may break down during the execution is very natural. Depending on what kind of real life system is considered, it may be essential that some basic functionality is maintained because the consequences of a complete failure may be catastrophic. Having said that, we still have to specify what it means for a system to "maintain some basic functionality". In this chapter, we consider the question how properties of an interaction system are influenced by a failure of a set of ports during runtime. Therefore in the following, we will speak (in a sense still to be defined) about robustness of a property of an interaction system rather than robustness of the system itself.

5.2 Defining Robustness of a Property

We formally define what effect failure of a set of ports during the execution of a system has on the system's properties. We will denote the set of ports that may fail by $\mathcal{A}' \subsetneq \mathcal{A}$.

We want to motivate why the definition of robustness of a property has to be approached with care. First of all, we have to decide what effect the failure of a port should have. We take up the position that failure of a port results in a situation were the corresponding local transitions are not enabled any more. This simply means that failure of a port always entails failure of all interactions involving this port. One could choose a different interpretation where from the outside it is not possible to notice that certain ports fail. This could be modeled such that an interaction α involving a defective port a_i would still be enabled in a state q enabling

all ports in α. Then α would result in a state change where component i (and possibly some other components that also participate in α with a defective port) remains in its local state. This approach would result in a much more complicated setting because it implies that new combinations of states would be reachable that are not accounted for in the specification of the system. We cannot expect such a system to behave in an orderly fashion without including particular treatment of these cases in the specification of the system. Instead, we choose the approach above. Considering robustness of deadlock-freedom, for example, we ask whether every reachable state still offers at least one interaction if the ports in $\mathcal{A}'$ fail. But even in this case the definition is not as straightforward as one might think at first glance.

Defining robustness of deadlock-freedom with respect to absence of $\mathcal{A}'$, one might be tempted to consider the "interaction system" obtained by deleting those local transitions from the local behaviors that are labeled with a port $a_i \in \mathcal{A}'$ and by deleting those interactions from *Int* that contain at least one such port. Then one might argue that deadlock-freedom is robust with respect to absence of $\mathcal{A}'$ if the resulting system is deadlock-free. This approach is problematic in various aspects. First, it is not formally correct. In general, we do not obtain an interaction system by simply removing transitions from the local transition systems and by deleting interactions. This may result in a system for which the local transition systems are terminating or for which there exist ports that are not contained in any interaction. These are technical details, and the definition of interaction systems could be adapted to a new situation allowing for such subtleties. Nonetheless, the main reason persists why this definition does not convey the meaning of robustness that we intend: By removing transitions and interactions *before* investigating the resulting system, we obtain a system whose set of reachable states may be a *proper* subset of the set of reachable states of the original system. Thus, even if we are able to show that every reachable state of the modified system enables an interaction not involving $\mathcal{A}'$ we cannot be sure that in the original system deadlock-freedom is robust with respect to failure of $\mathcal{A}'$. There may be a deadlock in a state q of *Sys* that is caused by failure of $\mathcal{A}'$ but cannot be detected by looking at the modified system simply because q is not reachable in that system. We need a definition which models

the fact that the ports in $\mathcal{A}'$ may fail at any point during the execution of the system and which makes sure that even in case of such a failure in every reachable state still at least one interaction α with $\alpha \cap \mathcal{A}' = \emptyset$ is enabled.

Definition 5.2.1. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}' \subsetneq \mathcal{A}$ be a nonempty subset of ports. Deadlock-freedom is* ***robust with respect to absence or failure of*** *$\mathcal{A}'$ in Sys if for every $q \in reach(Sys)$ there exists $\alpha \in excl(\mathcal{A}')$ that is enabled in q.*

We require *Sys* to be deadlock-free. This is necessary because a state which does not enable *any* interaction will certainly not satisfy the condition in the definition. Before turning to the effect that a failure of $\mathcal{A}'$ has on progress we want to point out an interesting relation between progress of $\mathcal{A}'$ in *Sys* and robustness of deadlock-freedom with respect to absence of $\mathcal{A}'$. If *Sys* is deadlock-free and $\mathcal{A}'$ makes progress in *Sys* then deadlock-freedom is not robust with respect to absence of $\mathcal{A}'$. If this was the case it would be possible to construct a run not letting any port in $\mathcal{A}'$ participate, and this is not possible. The converse does not hold.

Having defined what it means for deadlock-freedom to be robust with respect to absence of $\mathcal{A}'$, we investigate the effect that a failure of these ports has on the question whether a different set $\mathcal{A}_0$ of ports makes progress in *Sys*. We first have to adapt the notion of a run.

Definition 5.2.2. *Let Sys be a deadlock-free interaction system and $\mathcal{A}' \subsetneq \mathcal{A}$ be a subset of ports. A* ***run without*** *$\mathcal{A}'$ is an infinite sequence $\sigma = q \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} q^2 \xrightarrow{\alpha_2} \ldots$ in T_{Sys} with $q \in reach(Sys)$ and $\alpha_l \in excl(\mathcal{A}')$ for all $l \in \mathbb{N}$.*

We define what it means for a subset $\mathcal{A}_0$ to make progress without $\mathcal{A}'$.

Definition 5.2.3. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}', \mathcal{A}_0 \subsetneq \mathcal{A}$ be disjoint subsets of ports. Let deadlock-freedom in Sys be robust with respect to absence of $\mathcal{A}'$. $\mathcal{A}_0$* ***makes progress without (participation of)*** *$\mathcal{A}'$ if for every run $\sigma = q \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} \ldots$ without $\mathcal{A}'$ there is an index l with $\alpha_l \cap \mathcal{A}_0 \neq \emptyset$.*

Recalling the definition of robustness of deadlock-freedom, we see that the definition of progress without $\mathcal{A}'$ is slightly different. We do not require

$\mathcal{A}_0$ to make progress in *Sys*. Further, we do not speak of *robustness of progress* but of *progress without* $\mathcal{A}'$. The reason can easily be explained. If $\mathcal{A}_0$ makes progress in *Sys* then $\mathcal{A}_0$ participates in *every* run of *Sys*. This is true in particular for the runs without $\mathcal{A}'$. Therefore, we would not obtain an interesting property if we defined robustness of progress analogously to the way we defined robustness of deadlock-freedom. This property would directly follow from the fact that $\mathcal{A}_0$ makes progress and from robustness of deadlock-freedom. On the other hand, it is interesting to investigate whether $\mathcal{A}_0$ makes progress without $\mathcal{A}'$ because it is possible that certain runs in which $\mathcal{A}_0$ does not participate are no longer present when the ports in $\mathcal{A}'$ are not available any more. We require *Sys* to be a deadlock-free system where deadlock-freedom is robust with respect to absence of $\mathcal{A}'$. This is necessary to exclude the case that $\mathcal{A}_0$ makes progress in a system that does not have any run without $\mathcal{A}'$[1].

Before considering the question how to investigate the effect that a failure of $\mathcal{A}'$ has on a property, we need a few more definitions which adjust existing notions to the new situation.

Definition 5.2.4. *Let Sys be an interaction system and let* $\mathcal{A}' \subsetneq \mathcal{A}$ *be a nonempty subset of ports. Let* $i \in K$ *and* $q_i \in Q_i$. *By* $Int_{\mathcal{A}'}(q_i) := \{\alpha \in Int(q_i) | i(\alpha) \not\subseteq \mathcal{A}'\}$ *we denote the set of those interactions containing a port in* $\boldsymbol{en}(q_i)\backslash\mathcal{A}'$. *We say that* q_i *is* ***complete with respect to*** $\mathcal{A}'$ *if there exists an interaction* $\alpha = \{a_i\} \in Int_{\mathcal{A}'}(q_i)$. *The set of potential communication partners of component* i *in* q_i *after failure of* $\mathcal{A}'$ *is defined by* $need_{\mathcal{A}'}(q_i) := \bigcup_{\alpha \in Int_{\mathcal{A}'}(q_i)} comp(\alpha)\backslash\{i\}$.

$Int_{\mathcal{A}'}(q_i)$ describes the interactions that i still wants to perform in q_i if the ports in $\mathcal{A}'$ are not available any more. There can be an interaction $\alpha \in Int_{\mathcal{A}'}(q_i)$ with $\alpha \cap \mathcal{A}' \neq \emptyset$. In a global state q the system will not be able to perform such an interaction α if $\mathcal{A}'$ fails but from the point of view of i it does not make any difference whether this is the case because there is a component $j \in comp(\alpha)\backslash\{i\}$ with $j(\alpha) \subseteq \mathcal{A}'$ or with $j(\alpha) \not\subseteq \boldsymbol{en}(q_j)$.

[1] This requirement is analogous to the requirement that *Sys* should be deadlock-free made in Definition 2.2.2.

5.3 Testing Robustness of Properties

Having formally defined robustness of deadlock-freedom with respect to absence of $\mathcal{A}'$ and progress without $\mathcal{A}'$, we turn to the investigation of these properties. In particular, we are interested in the question how they can be tested efficiently. The need for such tests is supported by the fact that the complexity results for deciding the various properties carry over to deciding the corresponding property under absence of $\mathcal{A}'$. These results have not been published, yet, but a reduction from deciding deadlock-freedom to deciding robustness of deadlock-freedom, for example, can easily be accomplished as follows: For every component i we extend $\mathcal{A}_i$ by a new port a_i and for all q_i we add a loop labeled a_i. Adding the new interaction $\{a_1, \ldots, a_n\}$ to Int, we obtain a deadlock-free system for which deadlock-freedom is robust with respect to failure of $\mathcal{A}' := \{a_1, \ldots, a_n\}$ if and only if the original system was deadlock-free.

Robustness of a property is closely related to the property itself. Therefore an approach suggests itself which adapts existing conditions and ideas to the new situation. This way we hope to be able to profit from previous results. We follow this line of reasoning in the following subsections. We explain in detail how the sufficient conditions presented in Chapters 3 and 4 for deadlock-freedom of a tree-like interaction system respectively for progress can be adapted to treat these properties in case of failure of $\mathcal{A}'$.

5.3.1 Robustness of Deadlock-Freedom for Tree-Like Systems

The adaptation of the ideas presented in Chapter 3 is particularly convenient because the conditions for deadlock-freedom of tree-like interaction systems were formulated making sure that the different issues concerning deadlock-freedom, i.e., reachability of q and the fact that no interaction is enabled in q, were strictly separated. Thus, we may approach the question where to adapt the conditions formulated for each specific issue without having to tamper with the notions and conditions referring to the other. In particular, the strength of the approach chosen towards reachability in Section 3.2 becomes apparent. There, we did not really elaborate on how to suitably choose the parametrization of $\mathcal{EA}$. We only stated a simple

lemma (Lemma 3.2.3) indicating one possibility to draw conclusions about the reachability of a global state by choosing the parameters adequately. Now we recognize how the sets of components used to parametrized BWS and $\mathcal{EA}$ can be chosen to find out whether a certain combination of states is reachable: In Section 3.4.1 we defined the problematic actions of a component i with respect to a local state q_i using the notion of entry actions parametrized with $need(q_i)$. The reason for this choice of parameter was the fact that — at least in the view of q_i — the components in $need(q_i)$ had to be paired with i in order to find the sets of problematic states. This idea carries over to the current situation. Similarly, to the argument in Section 3.4.1 we want to identify those local states which may cause a deadlock under failure of $\mathcal{A}'$ when reached in combination with q_i. Only this time we consider the components in $need_{\mathcal{A}'}(q_i)$. In order to exclude the possibility that critical combinations of such states are globally reachable we compute the sets of entry actions where the parametrization has been adapted to the situation, i.e., we use $need_{\mathcal{A}'}(q_i)$ to parametrize BWS and $\mathcal{EA}$.

In Chapter 3 the motivation behind the definition of problematic states of j with respect to q_i and $\alpha \in Int(q_i)$ was to pinpoint those local states of j that might cause the system to deadlock if reached in combination with q_i (and possibly some other states). In Remark 3.4.1 (cf. pp. 55ff.) we informally derived this notion from considering a deadlock in a tree-like system. We saw that q_j did not necessarily have to be problematic with respect to q_i and α even if $j(\alpha) \not\subseteq \mathcal{en}(q_j)$. We only identified those q_j as problematic where $j(\alpha) \not\subseteq \mathcal{en}(q_j)$ *and* $need(q_j) \cap comp(\alpha) \neq \emptyset$. Considering robustness of deadlock-freedom with respect to failure of $\mathcal{A}'$, we check the analogous condition $need_{\mathcal{A}'}(q_j) \cap comp(\alpha) \neq \emptyset$ in order to identify those states of j which are problematic with respect to q_i and $\alpha \in Int_{\mathcal{A}'}(q_i)$ because also in this case cyclic waiting between the components may ultimately cause a deadlock. Only this time the sequences constituting the cycle of waiting relations we derived in Remark 3.4.1 must further satisfy the condition $i_s(\alpha_s) \not\subseteq \mathcal{A}'$. Otherwise i_s would not be ready to perform α_s anyway. This becomes manifest in the fact that in the condition mentioned above we replace $need(q_i)$ and $Int(q_i)$ by $need_{\mathcal{A}'}(q_i)$ respectively $Int_{\mathcal{A}'}(q_i)$. Note that $\alpha \in Int_{\mathcal{A}'}(q_i)$ not only can be blocked by $j \in comp(\alpha)$ because $j(\alpha) \not\subseteq \mathcal{en}(q_j)$ but also

because $j(\alpha) \subseteq \mathcal{A}'$. Thus, in order to find out whether q_j is problematic with respect to q_i and α we have to check both conditions. It is easy to see that this is equivalent to checking the more concise condition $\alpha \notin Int_{\mathcal{A}'}(q_j)$. The condition about the availability of interactions of size one or two also has to be adapted to the situation: Instead of checking completeness of a local state we check for *completeness with respect to* $\mathcal{A}'$. For the same reason we check whether an interaction $\alpha \in excl(\mathcal{A}')$ (instead of $\alpha \in Int$) with $comp(\alpha) = \{i, j\}$ is enabled in (q_i, q_j). This amounts to checking whether there is $\alpha \in Int_{\mathcal{A}'}(q_i) \cap Int_{\mathcal{A}'}(q_j)$ with $comp(\alpha) = \{i, j\}$. The requirement about reachability of (q_i, q_j) does not have to be changed.

Definition 5.3.1. *Let Sys be a tree-like interaction system and let $\mathcal{A}' \subsetneq \mathcal{A}$ be a nonempty subset of ports. For $i \in K$, $q_i \in Q_i$ incomplete with respect to $\mathcal{A}'$, $\alpha \in Int_{\mathcal{A}'}(q_i)$, and $j \in comp(\alpha) \setminus \{i\}$ we inductively define a descending sequence of subsets of Q_j by:*

$$\begin{aligned} PS^0_{j,\mathcal{A}'}(q_i, \alpha) := \{q_j \mid & \bullet\ q_j \text{ incomplete with respect to } \mathcal{A}' \\ & \bullet\ need_{\mathcal{A}'}(q_j) \cap comp(\alpha) \neq \emptyset \\ & \bullet\ (q_i, q_j) \text{ is reachable in } Sys\downarrow_{\{i,j\}} \\ & \bullet\ \alpha \notin Int_{\mathcal{A}'}(q_j) \\ & \bullet\ \nexists \tilde{\alpha} \in Int_{\mathcal{A}'}(q_i) \cap Int_{\mathcal{A}'}(q_j) \text{ with } |\tilde{\alpha}| = 2\} \\ PS^{l+1}_{j,\mathcal{A}'}(q_i, \alpha) := \{q_j \mid & q_j \in PS^l_{j,\mathcal{A}'}(q_i, \alpha) \text{ and } \forall \beta \in Int_{\mathcal{A}'}(q_j) \exists k \in \\ & comp(\beta) \setminus \{j\} \text{ with } PS^l_{k,\mathcal{A}'}(q_j, \beta) \neq \emptyset\} \end{aligned}$$

The set of states of j that are ***problematic with respect to*** *q_i, α,* ***and*** *$\mathcal{A}'$ is given by:*

$$PS_{j,\mathcal{A}'}(q_i, \alpha) := \bigcap_{l \in \mathbb{N}} PS^l_{j,\mathcal{A}'}(q_i, \alpha)$$

The condition in the inductive definition of the further sets of problematic states has also been adapted correspondingly by requiring the condition only for $\beta \in Int_{\mathcal{A}'}(q_i)$. For better readability we will simply speak of problematic states if there is no danger of confusion. In particular, in this chapter we will not explicitly mention $\mathcal{A}'$ because we only consider robustness of deadlock-freedom rather than deadlock-freedom itself and therefore it is clear that we always speak about problematic states with respect to $\mathcal{A}'$.

Using the adapted notion of problematic states and keeping in mind the introductory remark to this section about the choice of the parameters for the entry actions, it is now straightforward to define a notion of problematic actions for the current setting.

Definition 5.3.2. *Let Sys be a tree-like interaction system with strongly exclusive communication and let $\mathcal{A}' \subsetneq \mathcal{A}$ be a nonempty subset of ports. Let $i \in K$, $q_i \in Q_i$ incomplete with respect to $\mathcal{A}'$, $\alpha \in Int_{\mathcal{A}'}(q_i)$, and $j \in comp(\alpha) \setminus \{i\}$. We define*

$$\mathcal{PA}\left(q_i, \alpha, j, \mathcal{A}'\right) :=$$

$$\mathcal{EA}\left(q_i, need_{\mathcal{A}'}\left(q_i\right), PS_{j,\mathcal{A}'}\left(q_i, \alpha\right), \mathfrak{need}_{\mathcal{A}'}(PS_{j,\mathcal{A}'}\left(q_i, \alpha\right))\right)$$

the set of ***problematic actions of*** *i* ***with respect to*** *q_i, α, j,* ***and*** *$\mathcal{A}'$. For $Q'_j \subseteq Q_j$ we write: $\mathfrak{need}_{\mathcal{A}'}(Q'_j) := \{need_{\mathcal{A}'}(q_j)\}_{q_j \in Q'_j}$.*

The definitions above only treat the case where $\alpha \in Int_{\mathcal{A}'}(q_i)$, i.e., we explicitly exclude situations where for q_i and $\alpha \in Int(q_i)$ we have $i(\alpha) \subseteq \mathcal{A}'$. At the moment we can ignore such interactions because for now we only consider deadlocks resulting from a cycle of waiting relations between components. As mentioned above a component i can only participate in such a cycle with interaction α if it is ready to perform α. In case of failure of $\mathcal{A}'$ this is not the case if $i(\alpha) \subseteq \mathcal{A}'$. On the other hand, the failure of $\mathcal{A}'$ may also cause deadlocks which are *not* generated by cyclic waiting. It is possible that no interaction α with $\alpha \cap \mathcal{A}' = \emptyset$ is enabled in q any more simply because all interactions that were enabled in q involved at least one port in $\mathcal{A}'$. This is only one possibility to get a deadlock which is not caused by cyclic waiting between the components. An alternative variant for such a deadlock in q would only require the existence of a single component j with $Int_{\mathcal{A}'}(q_j) = \emptyset$. In this situation it might be possible that all other components (directly or indirectly) need j in order to proceed such that no $\alpha \in excl(\mathcal{A}')$ would be enabled. However, there would be no cycle of waiting relations and q_j would not be problematic with respect to any local state of any other component precisely because $need_{\mathcal{A}'}(q_j) = \emptyset$ follows from $Int_{\mathcal{A}'}(q_j) = \emptyset$. Therefore the condition in Definition 5.3.1 referring to $need_{\mathcal{A}'}(q_j)$ would never be satisfied. We conclude that the notion of problematic states is not suitable to deal with

local states q_j where $Int_{\mathcal{A}'}(q_j) = \emptyset$. According to our assumption $\boldsymbol{en}(q_j) \neq \emptyset$ for all local states q_j, this is equivalent to $\boldsymbol{en}(q_j) \subseteq \mathcal{A}'$. We have to deal with such states directly by a priori excluding the possibility of their occurrence.

This leads to the changes that have to be carried out with regard to the first condition of Proposition 3.4.1. First of all, we have to change the condition about completeness of a local state q_i to completeness with respect to $\mathcal{A}'$. The second part is only concerned with reachability of q_i from q_i^0. We simply adapt the parametrization of BWS by replacing $need(q_i)$ by $need_{\mathcal{A}'}(q_i)$. We get the condition $q_i^0 \notin BWS(q_i, need_{\mathcal{A}'}(q_i))$. The last condition is adapted such that it makes sure that there is at least one $\alpha \in Int_{\mathcal{A}'}(q_i)$ that cannot be blocked if q_i does not satisfy the first two conditions:

$$\exists \alpha \in Int_{\mathcal{A}'}(q_i) \forall j \in comp(\alpha) \backslash \{i\} :$$

$$q_j^0 \notin BWS\left(PS_{j,\mathcal{A}'}\left(q_i, \alpha\right), \mathfrak{need}_{\mathcal{A}'}(PS_{j,\mathcal{A}'}\left(q_i, \alpha\right))\right)$$

The combination of the three conditions mentioned above makes sure that there cannot be a local state q_i with $\boldsymbol{en}(q_i) \subseteq \mathcal{A}'$, without requiring this fact directly: If there was such a q_i it would definitely be incomplete with respect to $\mathcal{A}'$. Further, $\boldsymbol{en}(q_i) \subseteq \mathcal{A}'$ implies $Int_{\mathcal{A}'}(q_i) = \emptyset$ and therefore $need_{\mathcal{A}'}(q_j) = \emptyset$. Since q_i is reachable from q_i^0 in T_i we get $q_i^0 \in BWS(q_i, need_{\mathcal{A}'}(q_i))$. Finally, $Int_{\mathcal{A}'}(q_i) = \emptyset$ also shows that the third condition stated above is not satisfied for q_i. Thus, a state q_i with $\boldsymbol{en}(q_i) \subseteq \mathcal{A}'$ violates the first condition of the proposition stated below. Therefore this situation is taken care of in the criterion. Gathering the details we obtain the following result adjusting Proposition 3.4.1 to the situation where $\mathcal{A}'$ may fail.

Proposition 5.3.1. *Let Sys be a deadlock-free tree-like interaction system with strongly exclusive communication. Let $\mathcal{A}' \subsetneq \mathcal{A}$ be a nonempty subset of ports.*

If the following two conditions hold then deadlock-freedom is robust with respect to failure of $\mathcal{A}'$ in Sys.

1. $\forall i \forall q_i : q_i$ *complete with respect to* $\mathcal{A}' \vee q_i^0 \notin BWS(q_i, need_{\mathcal{A}'}(q_i)) \vee$
   $\exists \alpha \in Int_{\mathcal{A}'}\left(q_i\right)$ *such that* $\forall j \in comp\left(\alpha\right) \setminus \{i\} :$
   $q_j^0 \notin BWS\left(PS_{j,\mathcal{A}'}\left(q_i, \alpha\right), \mathfrak{need}_{\mathcal{A}'}(PS_{j,\mathcal{A}'}\left(q_i, \alpha\right))\right)$

2. $\forall \tilde{\alpha} \in Int$ *with* $|comp(\tilde{\alpha})| \geq 2$ $\exists i \in comp(\tilde{\alpha})$ *such that for all* $q_i \in Q_i$ *that are incomplete with respect to* $\mathcal{A}'$*:*

$$i(\tilde{\alpha}) \not\subseteq \bigcap_{\alpha \in Int_{\mathcal{A}'}(q_i)} \bigcup_{j \in comp(\alpha)} \mathcal{PA}(q_i, \alpha, j, \mathcal{A}')$$

The conditions can be checked in time polynomial in the size of Sys.

In order to get a better understanding of the implications caused by the changes in the notions and the criterion we consider the application of the results to Sys_{track} and Sys_{bank} introduced in Chapter 3. We point out some peculiarities that have to be taken into account when considering robustness of deadlock-freedom and applying the criterion.

Example 5.3.1. Consider Sys_{track} as defined in Section 3.6.3 (cf. pp. 80ff.). The remark made before Proposition 5.3.1 shows that the first condition and consequently the whole criterion is not satisfied if we consider failure of $\mathcal{A}'$ such that there is $i \in Sys_{track}$ and $q_i \in Q_i$ with $\mathcal{en}(q_i) \subseteq \mathcal{A}'$. In particular, this holds for any choice of $\mathcal{A}'$ where $\mathcal{A}' = \mathcal{en}(q_i)$. Setting $\mathcal{A}' = \{to_1^b\} = \mathcal{en}(right_1)$, for example, we obtain such a situation. The criterion is not satisfied. Of course initially this does not mean anything for the existence of a deadlock in Sys_{track} if to_1^b fails. It can be seen, though, that such a failure means that the first track is not able to allow a train waiting in the track to move on to the bifurcation any more (i.e., t_1 cannot leave the state $right_1$ because the only port that was enabled has failed). In this situation the system can reach a state q with $q_b = {}^{1}_{0}crit$ and $q_{t_2} = left_2$. Such a state does not enable any $\alpha \in excl(\{to_1^b\})$, and there is a deadlock. For this deadlock there does *not* exist a cycle of waiting relations between the components. Instead, a situation arises where t_1 does not wait for any component because $Int_{\mathcal{A}'}(right_1) = \emptyset$ and therefore also $need_{\mathcal{A}'}(right_1) = \emptyset$. All other components directly (st_1 and b) or indirectly (t_2 and st_2) wait for t_1. In fact, there will always be reachable deadlocks in Sys_{track} if a set of ports $\mathcal{A}' = \mathcal{en}(q_i)$ for some q_i fails. In general, $\mathcal{A}' = \mathcal{en}(q_i)$ for some q_i and the consequential violation of the first condition do not necessarily mean that deadlock-freedom is not robust. If we consider robustness of deadlock-freedom of Sys_{bank} (cf. pp. 71ff.) with respect to $\mathcal{A}' = \{request_{b_1}\}$ for an

instance of the system with at least two banks, for example, *cc* still operates correctly with all other banks. Deadlock-freedom is robust after all. The criterion cannot recognize this situation because it might be the case that all other components wait for b_1 but no cycle of waiting relations arises. This is indeed the case if b_1 is the only bank of the system.

Returning to Sys_{track}, we consider failure of $\mathcal{A}' = \{from_1^{st}\}$. The only interaction involving this port is $\alpha_1^1 = \{to_1^t, from_1^{st}, enter^{t_1}\}$. We obtain $Int_{\mathcal{A}'}(free_1) = \{\alpha_1^4\}$ and $need_{\mathcal{A}'}(free_1) = \{b\}$. All ports of t_1 are used for communication with b. Thus, $BWS(free_1, need_{\mathcal{A}'}(free_1)) = \{free_1\}$. No other local state is influenced by the failure of $\mathcal{A}'$. Therefore, the results obtained for BWS in Section 3.6.3 still hold. Computing the sets $PS_{j,\mathcal{A}'}(q_i, \alpha)$, it can be seen that for all combinations of q_i, $\alpha \in Int_{\mathcal{A}'}(q_i)$, and $j \in comp(\alpha)\backslash\{i\}$ that have to be investigated we have $PS_{j,\mathcal{A}'}(q_i, \alpha) = PS_j(q_i, \alpha)$. Compared to the computation of $PS_j(q_i, \alpha)$, in this case the computation of the sets of problematic states requires one more iteration, though, before they become stationary and the computation can be stopped. We do not have to compute $PS_{j,\mathcal{A}'}(free_1, \alpha_1^1)$ for any $j \in \{st_1, t_1\}$. In fact, this expression is not even defined because $\alpha_1^1 \notin Int_{\mathcal{A}'}(free_1)$. Since all sets of problematic states with respect to $\mathcal{A}'$ coincide with the corresponding regular sets of problematic states we also get $\mathcal{PA}(q_i, \alpha, j, \mathcal{A}') = \mathcal{PA}(q_i, \alpha, j)$ for all combinations of local states, interactions, and components. The only choice of q_i for which we may possibly have

$$\bigcap_{\alpha \in Int_{\mathcal{A}'}(q_i)} \bigcup_{j \in comp(\alpha)} \mathcal{PA}\left(q_i, \alpha, j, \mathcal{A}'\right) \neq \bigcap_{\alpha \in Int(q_i)} \bigcup_{j \in comp(\alpha)} \mathcal{PA}\left(q_i, \alpha, j\right)$$

is $q_i = free_1$ because it is the only state with $Int_{\mathcal{A}'}(q_i) \neq Int(q_i)$. The sets of problematic actions with respect to failure of $\mathcal{A}'$ coincide with the regular sets of problematic actions in all cases. In particular, $\mathcal{PA}\left(free_1, \alpha_1^4, b\right) = \mathcal{PA}\left(free_1, \alpha_1^4, b, \mathcal{A}'\right)$. Checking the details, we get $\mathcal{PA}\left(free_1, \alpha_1^4, b\right) = \emptyset$. Thus, both intersections above are empty and therefore coincide after all. Combining these arguments, we see that the second condition of Proposition 5.3.1 holds. The first condition only has to be checked for the local initial states of the components. It can be seen that it is also satisfied. Therefore, in Sys_{track} deadlock-freedom is robust with respect to failure of $\mathcal{A}' = \{from_1^{st}\}$.

At this point it is advisable to step back and to classify this result by looking at the bigger picture. We have shown that in Sys_{track} deadlock-freedom is robust with respect to failure of $\{from_1^{st}\}$. However, it is also clear that the functionality of the railway segment has been significantly impaired. Failure of $\{from_1^{st}\}$ results in a track that can henceforth only be traveled by trains coming from st_2, i.e., the segment has become a one-way track. Using this observation, we want to motivate that robustness of deadlock-freedom should be handled with care. We only guarantee that the system will not deadlock. We do not make any statement about the remaining functionality of the system. It is not surprising that the functionality of a system is influenced by the failure of some of its ports. Using our results, we can check whether this malfunction also involves deadlocks or whether the system may at least still proceed without the defective ports.

We conclude these arguments by pointing out a detail that should always be taken into account when applying the criterion. Since Proposition 5.3.1 requires Sys to have *strongly exclusive communication* we have to clarify how failure of a subset $\mathcal{A}'$ of ports in Sys is adequately represented by failure of a subset of ports in $\overline{Sys}$. A port a_i in Sys is modeled by a subset $\{a_i^\alpha | \alpha \in Int \text{ with } a_i \in \alpha\}$ of ports in $\overline{Sys}$. Therefore, failure of a_i has to be modeled by failure of all the ports representing a_i in $\overline{Sys}$. It is intuitively clear that this approach is reasonable, and one might ask why further consideration should be put into the use of $\overline{Sys}$ when we consider robustness of deadlock-freedom. The point is that we are only allowed to pass on to $\overline{Sys}$ *after* we have decided on $\mathcal{A}'$. The problem that may arise otherwise can be motivated by means of Sys_{bank}. Considering $\overline{Sys_{bank}}$ one might be tempted to expect that deadlock-freedom is robust with respect to failure of $\{request_{b_i}^1\}$ if the bank b_i has several ATMs. In this case the bank would not allow its first ATM to request money any more but because there is at least one more ATM the bank can still cooperate with the other ATMs and the system is deadlock-free. It might therefore come as a surprise that the conditions of Proposition 5.3.1 are *not* satisfied for $\overline{Sys_{bank}}$ and $\mathcal{A}' = \{request_{b_i}^1\}$. For $j \neq 1$ the interaction $\{request_{b_i}^j, request_j^i\}$ does not satisfy the second condition of the proposition. We encounter the following problem: Even though it is a meaningful question to ask whether deadlock-freedom is robust with

respect to failure of $\{request^1_{b_i}\}$ in $\overline{Sys_{bank}}$ this question is posed disregarding the fact that $request^1_{b_i}$ was introduced as only one port in a set of ports representing the port $request_{b_i}$ in Sys_{bank}. From this point of view it does not make sense to consider failure of only one of these ports in $\overline{Sys_{bank}}$ because it is not clear how this failure should be interpreted by the failure of a corresponding port in Sys_{bank}. We conclude that $\overline{Sys}$ may still be resorted to when we consider robustness of deadlock-freedom for a system without strongly exclusive communication. However, we have to choose the set $\mathcal{A}'$ *before* the transition to $\overline{Sys}$ and we have to model failure of $\mathcal{A}'$ in Sys by failure of $\overline{\mathcal{A}'} := \{a_i^\alpha | a_i \in \mathcal{A}' \wedge \alpha \in Int \wedge a_i \in \alpha\}$.

5.3.2 Progress without Participation of a Set of Ports

We turn to progress without $\mathcal{A}'$. The ideas developed in Chapter 4 in order to establish progress can also be used to treat progress without $\mathcal{A}'$. However, we will not follow the same approach that was taken in Chapter 4 where we first presented the progress graph of stage d in order to highlight the ideas used and to motivate the construction of the extended progress graph based on these ideas. Instead, we directly adapt the extended progress graph and state the most general result. The results also hold for an adapted progress graph where the construction of the edges is not iterated.

Again, we follow the guideline that the criterion should only be adjusted to the new situation as opposed to devising a completely new condition. Thus, our goal is a progress graph where an edge (i, j) conveys the meaning that for any run *without* $\mathcal{A}'$ component j can only participate a finite number of times before an interaction involving a port in $\mathcal{A}_i$ has to be performed. The idea originally used for the construction of $\mathcal{E}_d$ required inevitability of the set of interactions that globally "need" $\mathcal{A}_i$ with respect to j in at least one subsystem of size d, i.e., every cycle in the subsystem had to contain at least one of these interactions if j also participated in the cycle. It is easy to adjust this condition: We maintain the condition about inevitability but we only require it for those cycles in the subsystem that could possibly have resulted from a run without $\mathcal{A}'$. In the following we formalize these ideas. First, we need some preliminaries.

Definition 5.3.3. *Let $T = (Q, \mathcal{L}, \rightarrow)$ be a finite labeled transition system where Q is the set of states, $\mathcal{L}$ is the set of labels, and $\rightarrow \subseteq Q \times \mathcal{L} \times Q$ is the labeled transition relation. Let $\mathcal{L}' \subseteq \mathcal{L}$ be a subset of labels. We define the* ***behavior of*** *T* ***restricted to*** *$\mathcal{L}'$ by the labeled transition system $T^{\mathcal{L}'} := (Q, \mathcal{L}', \rightarrow)$ where $\rightarrow \subseteq Q \times \mathcal{L}' \times Q$ is the labeled transition relation of T restricted to those transitions whose label is in $\mathcal{L}'$.*

The behavior of T restricted to $\mathcal{L}'$ is obtained by deleting all transitions whose label is not in $\mathcal{L}'$. If T has a designated starting state q^0 there may be nodes that are not reachable from q^0 in $T^{\mathcal{L}'}$.

Definition 5.3.4. *Let Sys be an interaction system and let $\mathcal{A}_0, \mathcal{A}' \subsetneq \mathcal{A}$ be disjoint subsets of ports. Let $1 \leq d \leq n$ be given. The* ***extended d-th stage progress graph for Sys and $\mathcal{A}_0$ with respect to failure of $\mathcal{A}'$*** *is defined as follows:*

$$\mathcal{G}_{d,\mathcal{A}'} := (\mathcal{V}, \mathcal{E}_{d,\mathcal{A}'})$$

The set of nodes is $\mathcal{V} := \{i | i \in K \wedge \mathcal{A}_i \not\subseteq \mathcal{A}'\} \cup \{0\}$. The set of directed edges is $\mathcal{E}_{d,\mathcal{A}'} := \bigcup_{m \geq 1} \mathcal{E}^m_{d,\mathcal{A}'}$. For $m \geq 1$ the sets $\mathcal{E}^m_{d,\mathcal{A}'}$ are inductively defined by:

$$\begin{aligned}\mathcal{E}^m_{d,\mathcal{A}'} := \{(i,j) \,|\, & j \neq i, 0 \text{ and } \exists K_d \subseteq K \text{ with } |K_d| = d \text{ and } j \in K_d \text{ such that} \\ & excl(\mathcal{A}') \downarrow_{K_d} \setminus excl(\mathcal{A}_{r^{m-1}_{d,\mathcal{A}'}(i)}) \downarrow_{K_d} \text{ is inevitable with respect} \\ & \text{to } excl(\mathcal{A}', j) \downarrow_{K_d} \text{ in } T^{excl(\mathcal{A}')\downarrow_{K_d}}_{Sys\downarrow_{K_d}}\}\end{aligned}$$

We write $excl(\mathcal{A}', j) := excl\left(\mathcal{A}'\right)(j)$ for the set of interactions $\alpha \in excl(\mathcal{A}')$ with $j(\alpha) \neq \emptyset$ and $r^m_{d,\mathcal{A}'}(i) := \{k | k \text{ is reachable from } i \text{ in } (\mathcal{V}, \bigcup_{l=1}^m \mathcal{E}^l_{d,\mathcal{A}'})\}$.

We get statements analogous to the results stated in Chapter 4.

Lemma 5.3.1. *Let Sys be an interaction system and let $\mathcal{A}_0, \mathcal{A}' \subsetneq \mathcal{A}$ be disjoint subsets of ports.*

1. *For $m \geq 1$ we have*

$$\begin{aligned}\mathcal{E}^m_{1,\mathcal{A}'} = \{(i,j) \,|\, & j \neq i, 0 \text{ and } excl_j(\mathcal{A}') \setminus excl_j(\mathcal{A}_{r^{m-1}_{1,\mathcal{A}'}(i)}) \\ & \text{is inevitable in } T_j^{excl_j(\mathcal{A}')}\}.\end{aligned}$$

2. *Let* $1 \leq d' < d \leq n$ *and* $m \geq 1$ *be given and let* $i \in \mathcal{V}$ *and* $j \in \mathcal{V} \setminus \{i, 0\}$. *If there is an edge* $(i, j) \in \mathcal{E}^m_{d',\mathcal{A}'}$, *then we also have* $(i, j) \in \mathcal{E}^m_{d,\mathcal{A}'}$.

For the construction of the first stage of the extended progress graph with respect to failure of $\mathcal{A}'$ we may again resort to the more simple definition of inevitability. The second part of the lemma states that for a chosen stage d of the extended progress graph with respect to failure of $\mathcal{A}'$ we do not have to consider any stage d' where $d' < d$ because $\mathcal{E}_{d',\mathcal{A}'} \subseteq \mathcal{E}_{d,\mathcal{A}'}$.

We adapt Lemma 4.3.6 and Proposition 4.3.2 to the new situation.

Lemma 5.3.2. *Let Sys be an interaction system, and let* $1 \leq d \leq n$ *be given. Let* $\mathcal{A}_0, \mathcal{A}' \subsetneq \mathcal{A}$ *be disjoint subsets of ports.*

$\mathcal{G}_{d,\mathcal{A}'}$ *can be constructed in time polynomial in the size of Sys. An upper bound for the cost is given by*

$$O(|Int|^2 n^{d+4} + |Int| n^{d+3} |T_{max}|^d)$$

where $|T_{max}|$ *denotes the size of the largest local transition system.*

Proposition 5.3.2. *Let Sys be a deadlock-free interaction system and let* $\mathcal{A}_0, \mathcal{A}' \subsetneq \mathcal{A}$ *be disjoint subsets of ports. Let deadlock-freedom be robust with respect to failure of* $\mathcal{A}'$ *in Sys. Let* $1 \leq d \leq n$ *be given and let* $\mathcal{G}_{d,\mathcal{A}'}$ *be defined as above.*

If all nodes in $\mathcal{V}$ *are reachable from* 0 *in* $\mathcal{G}_{d,\mathcal{A}}$ *then* $\mathcal{A}_0$ *makes progress without* $\mathcal{A}'$ *in Sys. The condition can be checked in time polynomial in the size of Sys.*

We end this section by discussing an example. It is merely a toy example which could easily be verified by hand. We mainly want to illustrate the construction of the graph and the criterion.

Example 5.3.2. Example: We model a small server/user system consisting of a user u who may choose between three different services. The first one is offered by server s_1, the second one is offered by server s_2, and the third service requires both servers simultaneously. The system also includes two maintenance components m_1 and m_2 whose job it is to check the servers after they have provided service to u.

We define $K_{s/u} := \{u, s_1, s_2, m_1, m_2\}$. The transition systems of the components are depicted in Figure 5.1. For each i the port set $\mathcal{A}_i$ is understood to coincide with the set of labels of T_i. We define $Int_{s/u}$ to be the set of interactions containing $\{internal\}$, $\{req_{1,2}, service_1, service_2\}$, and $\{req_i, service_i\}$ respectively $\{maint_i, m_j^i\}$ for any choice of $i, j \in \{1, 2\}$. We call the induced interaction system $Sys_{s/u}$.

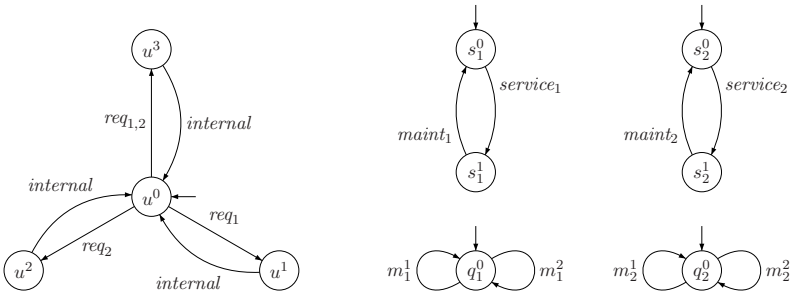

Figure 5.1: The local behavior of the components in $K_{s/u}$

$Sys_{s/u}$ is deadlock-free, and deadlock-freedom is robust with respect to failure of $\mathcal{A}_{m_2}$. We consider the question whether $\mathcal{A}_0 := \mathcal{A}_{m_1}$ makes progress without $\mathcal{A}' := \mathcal{A}_{m_2}$ in $Sys_{s/u}$, and we retrace the construction of $\mathcal{G}_{1,\mathcal{A}_{m_2}}$. Figure 5.2 depicts part of the graph. The edges in $\mathcal{E}^1_{1,\mathcal{A}'}$ are represented as solid lines. We are not able to conclude that $\mathcal{A}_{m_1}$ makes progress without $\mathcal{A}_{m_2}$ because u is not reachable from 0. The dashed edge from 0 to u is added during the first iteration step of the construction of the edges. It is not in $\mathcal{E}^1_{1,\mathcal{A}'}$ because $excl_u(\mathcal{A}')\backslash excl_u(\mathcal{A}_0) = \emptyset$. When we consider $\mathcal{E}^2_{1,\mathcal{A}'}$ we can take into account the fact that at this point all other components are reachable from 0. The set $excl_u(\mathcal{A}')\backslash excl_u(\mathcal{A}_{r^1_{1,\mathcal{A}'}(0)}) = \{req_1, req_2, req_{1,2}\}$ is inevitable in T_u, and therefore $(0, u) \in \mathcal{E}^2_{1,\mathcal{A}'}$. Progress of m_1 without m_2 follows from Proposition 5.3.2.

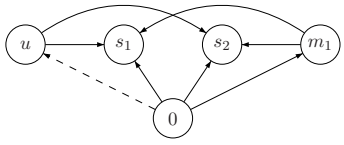

Figure 5.2: (Part of) $\mathcal{G}_{1,\mathcal{A}_{m_2}}$ for $Sys_{s/u}$

This property can be understood such that each service component still undergoes maintenance regularly even if the second maintenance component fails. The example could be extended to allow for arbitrarily many users such that an analogous statement about progress of $\mathcal{A}_{m_1}$ without $\mathcal{A}_{m_2}$ holds.

5.4 Conclusion and Related Work

5.4.1 Conclusion and Discussion

We investigated the effect that failure of a set of ports has on the properties of a system. We defined *robustness* of deadlock-freedom with respect to failure of $\mathcal{A}'$ and progress of $\mathcal{A}_0$ *without* $\mathcal{A}'$. Since the presence of a property under the assumption that the ports in $\mathcal{A}'$ fail is closely related to the presence of the property itself in the original system, we chose the approach to consider the criteria introduced before and to adapt the ideas to a situation where $\mathcal{A}'$ fails. We carried out the details of this approach for deadlock-freedom of tree-like interaction systems and for progress of $\mathcal{A}_0$ without $\mathcal{A}'$. By means of the examples presented in Chapter 3 we pointed out some details that have to be taken into account when dealing with robustness of deadlock-freedom. Furthermore, we presented a small example illustrating the criterion for progress without $\mathcal{A}'$.

We would like to draw attention to one issue that has been omitted so far. We considered existing criteria and adapted the underlying ideas to a situation where $\mathcal{A}'$ fails. This is reasonable because of the similarities of the definition of a property compared to the definition of the corresponding property under the assumption that the ports in $\mathcal{A}'$ fail. However, so far we have not incorporated any previous knowledge we might have into our criteria. Recall that we had to require the original system to be deadlock-free in order to be even able to define robustness of deadlock-freedom. Thus, at least as far as robustness of deadlock-freedom is concerned we indeed have previous knowledge that could be incorporated into the investigation. We know that *Sys* is deadlock-free. We have not used this knowledge in our approach. We informally explain one idea indicating how this could be realized: The notion of problematic states is fundamental to our considerations

regarding deadlock-freedom of tree-like interaction systems. A deadlock in q always entails the existence of a cycle of waiting relations as mentioned in Remark 3.4.1. The converse is not true in general. There may be cycles of waiting relations among components in q without q being a deadlock. In Chapter 3 we were not able to distinguish the fact that two local states where problematic with respect to each other because of such an "unproblematic" cycle from the fact that this was because of a cycle which is really involved in a global deadlock in q. Therefore, we had to assume the worst case meaning that any problematic state had to be taken into account. We had to accept the possibility that some of these states might never cause deadlocks. The situation is different now. Roughly speaking, we know that the problematic states that result from a cycle of waiting conditions in q such that no component i on this cycle is influenced by the failure of $\mathcal{A}'$ (i.e., $Int_{\mathcal{A}'}(q_i) = Int(q_i)$ for all these components) are not problematic after all. For example, consider a case where $\mathcal{A}'$ only consists of one port a_k. There could be a set K' of components and local states q_i for $i \in K'$ that on the one hand are completely independent of any state q_k that is influenced by the failure of a_k, i.e., these local states never need cooperation with q_k, but where the corresponding sets of problematic states among the components in K' are of such nature that the conditions of Proposition 5.3.1 are violated. Since we know that the original system is deadlock-free, we know that this violation of the conditions cannot be the *cause* of a deadlock after failure of $\mathcal{A}'$. This does not necessarily mean that in a global state consisting of a combination of these problematic states an interaction only involving components in K' is enabled. Indeed, such a state could be a global deadlock precisely because a_k failed. What we do know is the fact that for K' these local states must already have been problematic with respect to each other *before* failure of a_k because $Int_{\mathcal{A}'}(q_i) = Int(q_i)$. Since we know that *Sys* is deadlock-free, we know that for every reachable global state whose entries for K' coincide with a combination of these problematic states at least one interaction α was enabled. Either we have $a_k \notin \alpha$ for at least one such α showing that α is enabled in q even if a_k is not available any more. On the other hand, if all interactions α that were enabled in q involve a_k we have seen that one of the conditions in Proposition 5.3.1 must be violated.

The important point is that this violation will manifest itself in components that are *not* in K' because of the assumption that the components in K' are independent of any local state q_k that enables a_k. We conclude that in this case we may ignore any violation of Proposition 5.3.1 which is caused by components that are not influenced by a failure of the ports in $\mathcal{A}'$.

Having discussed this idea, we also have to acknowledge that even though it is promising with regard to improving the preciseness of the criterion it is not as simple and practical as it might seem at first glance. Even though one might have an intuitive idea of what it means for i respectively q_i *not* to be influenced by the failure of $\mathcal{A}'$ it proves to be more complicated to formally grasp this notion. First, it is not obvious how to define it because a local state q_i might be influenced by the failure of a_j not allowing j to proceed any more, not only because q_i directly waits for j but also because it waits for j because of a chain of waiting relations. Second, this argument already hints at a problem that arises from the fact that we only want to consider subsystems consisting of pairs of interacting components. It is not clear how to find those states *in practice* that are directly or indirectly influenced by the failure of $\mathcal{A}'$, while still only considering subsystems of size two. A thorough treatment of this idea and similar enhancements would go beyond the scope of this chapter. We omit it with the conclusion that such detailed refinements are possible once the criterion has reached a stage where it is implemented and used in practice.

We conclude the discussion of the results by a short remark about those cases where the criteria presented above fail to show that the property in question is present if the ports in $\mathcal{A}'$ are defective. These cases will arise because of the complexity results motivated in the beginning of Section 5.3. It is clear that the criteria have the same weaknesses as the counterparts they have been derived from. Note that ideas as the one sketched above may help to circumvent some of these shortcomings because in case of failure of the criterion we may still be able to derive information about the property from previous knowledge about the system. This is a significant difference compared to the treatment of the corresponding properties in the original system where we do not have such previous knowledge. On the other hand, the criteria of course also have the same strengths as the corresponding

criteria they are derived from. Again a failure of Proposition 5.3.1 can be interpreted as an indication where to perform direct checks in the same way that was motivated in the discussion to Chapter 3. Similarly, ways out can be found if Proposition 5.3.2 fails to show that $\mathcal{A}_0$ makes progress without $\mathcal{A}'$. If the assumption that $\mathcal{A}_0$ indeed makes progress without $\mathcal{A}'$ seems likely but the condition in Proposition 5.3.2 is not satisfied it is possible to increment the parameter d used in the construction of $\mathcal{G}_{d,\mathcal{A}'}$ in order to increase the precision of the criterion. Alternatively, the information derived from $\mathcal{G}_{d,\mathcal{A}'}$ could be combined with other conditions conceived for progress without $\mathcal{A}'$ in a similar manner to the suggestions given in Section 4.3.3 with regard to joining Theorem 4.2.1 and Proposition 4.3.1.

5.4.2 Robustness of Other Properties

Other properties can be considered with respect to failure of $\mathcal{A}'$, as well. It is possible to define robustness of freedom of local deadlocks analogously to Definition 5.2.1. The result concerning freedom of local deadlocks of tree-like interaction systems presented in Section 3.5.2 can be adapted to account for this property in the same way that we deduced Proposition 5.3.1 from Proposition 3.4.1. Furthermore, it is possible to carry out the same simplifications in the notions and criteria that were obtained from considering strongly tree-like systems in Sections 3.4.2 respectively 3.5.1.

We further refer to Majster-Cederbaum and Martens [98]. Section 5.3.2 extends some of the results presented there. The paper also includes further results about robustness of deadlock-freedom. In addition, an approach to treat liveness[2] without $\mathcal{A}'$ and ideas on how to adjust the results of Majster-Cederbaum et al. [102] in order to be able to treat robustness of freedom of local deadlocks are sketched.

5.4.3 Related Work

Having been an active field of research for a long time (cf. Lee and Anderson [95] or Jalote [86], for example), there are vast numbers of works

[2]We have not considered liveness in this thesis.

that investigate dependability in distributed systems in all kinds of contexts. Being a rather general term[3] it comprises such concepts as reliability and availability [95], for example. Reliability is concerned with investigating how probable it is that a system does not conform to its specification because of faults in the design or in one of the parts of the system. The above description, though not being formally precise, clearly implies that such approaches must involve concepts taken from probability theory in order to describe the probability that a system behaves correctly with regard to its specification. Thus, such approaches cannot really be compared to the approach taken in this chapter. We mention that there are various works considering reliability of component systems in the sense above. Based on Markov Chain theory, Cheung [47] or Reussner et al. [124], for example, offer frameworks which can be used to estimate reliability of component systems. We refer to Dimov and Punnekkat [61] for a review of further existing approaches and directions of current research in reliability of component systems.

The results presented in this chapter can rather be interpreted in the context of availability or more precisely fault tolerance whose maximization is striven for in order to make sure that for "most of the time" a system is delivering its service respectively at least a limited form of it. There are various definitions of fault tolerance. For example, Arora and Kulkarni [17] define three types of fault tolerance which differ in the kind of properties they preserve (i.e., only safety properties or only liveness properties or both). The authors introduce so called *detectors* and *correctors* and show that each of these components of a program is necessary and sufficient for that program to exhibit a certain type of fault tolerance. On the other hand, there are definitions that characterize fault tolerance in four phases: error detection, damage confinement/assessment, error recovery, and fault treatment/continued system service [86, 95]. The realization of these phases is tackled differently in various approaches, but the listing above shows that such approaches at best aim at fully restoring a faulty system. There are approaches to incorporate fault tolerance in this sense into component based systems (cf. Troubitsyna [135] or recently Hamid et al. [76, 77]), but it is

[3]Lee and Anderson [95] define dependability as that property of a (computing) system which allows reliance to be justifiably placed on the service it delivers.

also clear that our notion of robustness of a property is different.

In our perspective robustness can be seen as one type of *graceful degradation* or *survivability*. In many cases it is not possible to afford the cost of implementing the phases above. Graceful degradation offers a compromise which allows for a weaker degree of fault tolerance. In this context, Herlihy and Wing [81] use *relaxation lattices* to specify different constraints describing the different requirements put upon the system behavior in the presence of anomalies in the environment the system is deployed in. Under the expected circumstances the system has to conform to all constraints but they can be weakened (or "gracefully relaxed") to describe what deviation from the specification is acceptable if unexpected faults occur. Survivability constitutes a variant of graceful degradation. There are various (informal) definitions of what it means for a system to be survivable[4]. Deutsch and Willis [57], for example, describe survivability as the ability of a system to maintain the availability of essential functions even though some part of the system is down (i.e., not working correctly or even not working at all). Even though they are not equivalent, all these characterizations require a system to be able to bridge phases where its service is impaired in a not intended way by still offering some (possibly degraded) service instead of completely shutting down. Robustness of deadlock-freedom as described in this chapter is a *necessary* condition for a system to be survivable because the system definitely will not provide *any* service at all under failure of $\mathcal{A}'$ if it reaches a state which does not offer any interaction in $\mathit{excl}(\mathcal{A}')$. There are works which investigate survivability in the context of component systems. Shelton and Koopman [129], for example, achieve survivability by offering alternative services if a component fails. Saridakis [125, 126], on the other hand, does so by shutting down components that are affected by a failure and by propagating the effect of this shutdown to the other components in the hope that it will not affect all components. However, these results are too different in order to relate them to our approach. It might be interesting, though, to see whether it is possible to define a notion of robustness of deadlock-freedom corresponding to the one we presented in

[4]We refer to Knight et al. [89] for one attempt to formalize the notion of survivability.

this chapter for the component models we discussed in Section 3.7.2. Then, one could try to adapt the conditions that were presented in those models to be suitable to also handle robustness. As noted in the previous subsection Majster-Cederbaum and Martens [98] present an informal explanation how to adapt the results by Majster-Cederbaum et al. [102] to be able to also treat robustness of freedom of local deadlocks. On the other hand, it is not self-explanatory how to incorporate failure of actions into the equivalences used to formulate the other criteria discussed in Section 3.7.2. We will not further delve into these considerations here.

We conclude by pointing out that even though we investigated a special form of fault tolerance (in fact, even a special form of survivability for that matter) in the example discussing $Sys_{s/u}$ we encountered a characteristic which is a common factor to all approaches dealing with fault tolerance: redundancy. Redundancy refers to those parts of a system that are not needed for the correct functioning of the system if no fault tolerance is to be supported [86]. In the example the redundancy expressed itself in the fact that there were *two* maintenance components m_1 and m_2 even though one would suffice under normal conditions. It might not seem very sophisticated to mask possible failures by replicating components. However, it is not realistic to expect a system to be fault tolerant (respectively a property to be robust) without *beforehand* providing some mechanism which is able to absorb the effects of a failure during runtime.

5.5 Proofs

5.5.1 Proofs for Section 5.3.1

At last it becomes obvious why we — seemingly — took a detour in the proof section of Chapter 3 by proving lemmas that were more general than necessary. We are now able to give the statements needed for the proof of Proposition 5.3.1 as simple corollaries to the lemmas derived in Section 3.8.

Corollary 5.5.1. *Let Sys be a tree-like interaction system and let $\mathcal{A}' \subsetneq \mathcal{A}$ be a nonempty subset of ports. Let $q \in Q$ be a state with $need_{\mathcal{A}'}(q_i) \neq \emptyset$ for all $i \in K$.*

There exists a nonempty subset K' of components such that for all $i \in K'$ and for all $\alpha' \in Int_{\mathcal{A}'}(q_i)$ the following two conditions hold:

1. $\forall \alpha \in Int$ *with* $|comp(\alpha') \cap comp(\alpha)| \geq 2$ *we have* $comp(\alpha) \subseteq K'$
2. $\forall j \in comp(\alpha') \setminus \{i\} : need_{\mathcal{A}'}(q_j) \cap comp(\alpha') \neq \emptyset$

Proof. The condition required for $\tilde{K}$ in Lemma 3.8.2 is satisfied for K since $comp(\alpha) \subseteq K$ for all $\alpha \in Int$. Applying the lemma with $\tilde{K} = K$ and $Int'(q_i) = Int_{\mathcal{A}'}(q_i)$ for all q_i, yields the statement of the corollary. □

Corollary 5.5.2. *Let Sys be a tree-like interaction system and let $\mathcal{A}' \subseteq \mathcal{A}$ be a nonempty subset of ports. Let $q \in Q$ a state with $need_{\mathcal{A}'}(q_i) \neq \emptyset$ for all $i \in K$. Choose K' to be a minimal set of components satisfying the two conditions stated in Corollary 5.5.1. Let $\alpha \in Int$ be an interaction with $comp(\alpha) \subseteq K'$ and $|comp(\alpha)| \geq 2$ and let $j \in comp(\alpha)$.*

We have $need_{\mathcal{A}'}(q_j) \cap comp(\alpha) \neq \emptyset$.

Proof. Applying Lemma 3.8.3 with $Int'(q_i) = Int_{\mathcal{A}'}(q_i)$ for all i and all q_i, yields the statement of the corollary. □

Corollary 5.5.3. *Let Sys be a tree-like interaction system with strongly exclusive communication and let $\mathcal{A}' \subsetneq \mathcal{A}$ be a nonempty subset of ports. Let $q \in Q$ be a state with $need_{\mathcal{A}'}(q_i) \neq \emptyset$ for all $i \in K$. Choose K' as in Corollary 5.5.1. Let $i \in K'$ and $\alpha \in Int$ with $i(\alpha) \neq \emptyset$.*

If there is $k \notin K'$ with $k(\alpha) \neq \emptyset$ then $i(\alpha) \not\subseteq \bigcup_{j \in need_{\mathcal{A}'}(q_i)} \mathit{comm}_i(j)$.

Proof. Assume $i(\alpha) \subseteq \bigcup_{j \in need_{\mathcal{A}'}(q_i)} \mathit{comm}_i(j)$. Applying Lemma 3.8.4 with $Int'(q_k) = Int_{\mathcal{A}'}(q_k)$ for all $k \in K$ shows $comp(\alpha) \cap need_{\mathcal{A}'}(q_i) = \emptyset$. Thus, there must be $\alpha' \neq \alpha$ with $comp(\alpha') \cap need_{\mathcal{A}'}(q_i) \neq \emptyset$ and $i(\alpha') = i(\alpha)$. This is not possible because *Sys* has strongly exclusive communication. □

The following lemma takes up the generalized notion of problematic state that was introduced in Definition 3.8.2 (cf. p. 107) and shows that it coincides with $PS_{j,\mathcal{A}'}(q_i, \alpha)$ if we set $Int'(q_k) = Int_{\mathcal{A}'}(q_k)$ for all $k \in K$.

Lemma 5.5.1. *Let Sys be a tree-like interaction system and let $\mathcal{A}' \subsetneq \mathcal{A}$ be a subset of ports. Let $i \in K$, q_i incomplete with respect to $\mathcal{A}'$, $\alpha \in Int_{\mathcal{A}'}(q_i)$, and $j \in comp(\alpha) \setminus \{i\}$.*

Setting $Int'(q_k) = Int_{\mathcal{A}'}(q_k)$ *for all* $k \in K$*, we have*

$$PS^{l}_{j,\mathcal{A}'}(q_i, \alpha) = PS'^{l}_{j}(q_i, \alpha)$$

for all $l \in \mathbb{N}$*.*

Proof. We will show that each of the conditions stated in the definition of problematic states with respect to failure of $\mathcal{A}'$ (Definition 5.3.1, p. 178) is equivalent to the corresponding condition stated in Definition 3.8.2. It is clear that q_i is incomplete with respect to $\mathcal{A}'$ if and only if there is no $\alpha \in Int_{\mathcal{A}'}(q_i) = Int'(q_i)$ with $|\alpha| = 1$. Therefore the precondition stated for q_i in Definition 5.3.1 is equivalent to the precondition in Definition 3.8.2.

The proof uses induction over l.

$\underline{l = 0:}$ Setting $Int'(q_i) = Int_{\mathcal{A}'}(q_i)$ for all i and all q_i in Definition 3.8.2, we see that each condition is equivalent to the corresponding one in Definition 5.3.1 where as above we argue that q_j is incomplete with respect to $\mathcal{A}'$ if and only if there is no $\beta \in Int'(q_j) = Int_{\mathcal{A}'}(q_j)$ with $|\beta| = 1$ and we use the fact that $Int'(q_j) = Int_{\mathcal{A}'}(q_j)$ for all q_j also implies $need'(q_j) = need_{\mathcal{A}'}(q_j)$. Note that the condition about reachability of (q_i, q_j) the same in both definitions. We conclude $PS^{0}_{j,\mathcal{A}'}(q_i, \alpha) = PS'^{0}_{j}(q_i, \alpha)$.

$\underline{l \Rightarrow l+1:}$ We have:

$$\begin{aligned}
PS^{l+1}_{j,\mathcal{A}'}(q_i, \alpha) &= \{q_j | q_j \in PS^{l}_{j,\mathcal{A}'}(q_i, \alpha) \text{ and } \forall \beta \in Int_{\mathcal{A}'}(q_j) \\
&\qquad \exists k \in comp(\beta) \setminus \{j\} \text{ with } PS^{l}_{k,\mathcal{A}'}(q_j, \beta) \neq \emptyset\} \\
&= \{q_j | q_j \in PS^{l}_{j,\mathcal{A}'}(q_i, \alpha) \text{ and } \forall \beta \in Int'(q_j) \\
&\qquad \exists k \in comp(\beta) \setminus \{j\} \text{ with } PS^{l}_{k,\mathcal{A}'}(q_j, \beta) \neq \emptyset\} \\
&= \{q_j | q_j \in PS'^{l}_{j}(q_i, \alpha) \text{ and } \forall \beta \in Int'(q_j) \\
&\qquad \exists k \in comp(\beta) \setminus \{j\} \text{ with } PS'^{l}_{k}(q_j, \beta) \neq \emptyset\} \\
&= PS'^{l+1}_{j}(q_i, \alpha)
\end{aligned}$$

The third equality applies the induction hypothesis.

□

Corollary 5.5.4. *Let Sys be a tree-like interaction system and let $\mathcal{A}' \subsetneq \mathcal{A}$ be a subset of ports. Let $q \in reach(Sys)$ be a state with $need_{\mathcal{A}'}(q_i) \neq \emptyset$ for all i such that no $\tilde{\alpha} \in excl(\mathcal{A}')$ is enabled in q and let K' be given for q as in Corollary 5.5.1. Let $i \in K'$ and $\alpha \in Int_{\mathcal{A}'}(q_i)$.*

There exists $j \in comp(\alpha) \setminus \{i\}$ with $q_j \in PS_{j,\mathcal{A}'}(q_i, \alpha)$.

Proof. The second defining property of K' implies $need_{\mathcal{A}'}(q_j) \cap comp(\alpha) \neq \emptyset$ for all $j \in comp(\alpha) \setminus \{i\}$. Because no $\tilde{\alpha} \in excl(\mathcal{A}')$ is enabled in q we know that there is at least one $j \in comp(\alpha)$ with $\alpha \notin Int_{\mathcal{A}'}(q_j)$. The first defining property of K' together with $i \in K'$ and $\alpha \in Int_{\mathcal{A}'}(q_i)$ implies $comp(\alpha) \subseteq K'$ and therefore $j \in K'$.

Setting $Int'(q_k) = Int_{\mathcal{A}'}(q_k)$ for all $q_k \in Q_k$, the conditions for Lemma 3.8.6 are met. We conclude $q_j \in PS'_j(q_i, \alpha) = PS_{j,\mathcal{A}'}(q_i, \alpha)$ where the equality follows from Lemma 5.5.1. □

Proposition 5.3.1. *Let Sys be a deadlock-free tree-like interaction system with strongly exclusive communication. Let $\mathcal{A}' \subsetneq \mathcal{A}$ be a nonempty subset of ports.*

If the following two conditions hold then deadlock-freedom is robust with respect to failure of $\mathcal{A}'$ in Sys.

1. $\forall i \forall q_i : q_i$ *complete with respect to* $\mathcal{A}' \vee q_i^0 \notin BWS(q_i, need_{\mathcal{A}'}(q_i)) \vee$
   $\exists \alpha \in Int_{\mathcal{A}'}(q_i)$ *such that* $\forall j \in comp(\alpha) \setminus \{i\}$:
   $q_j^0 \notin BWS\left(PS_{j,\mathcal{A}'}(q_i, \alpha), \mathfrak{need}_{\mathcal{A}'}(PS_{j,\mathcal{A}'}(q_i, \alpha))\right)$

2. $\forall \tilde{\alpha} \in Int$ *with* $|comp(\tilde{\alpha})| \geq 2$ $\exists i \in comp(\tilde{\alpha})$ *such that for all* $q_i \in Q_i$ *that are incomplete with respect to* $\mathcal{A}'$*:*

$$i(\tilde{\alpha}) \not\subseteq \bigcap_{\alpha \in Int_{\mathcal{A}'}(q_i)} \bigcup_{j \in comp(\alpha)} \mathcal{PA}(q_i, \alpha, j, \mathcal{A}')$$

The conditions can be checked in time polynomial in the size of Sys.

Proof. Assume that the two conditions in the proposition hold but there is $q \in reach(Sys)$ such that no interaction not involving $\mathcal{A}'$ is enabled: There is a path $\sigma := q^0 \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_{l-1}} q^l \xrightarrow{\alpha_l} q$ and no $\alpha \in excl(\mathcal{A}')$ is enabled in q.

First we will show $need_{\mathcal{A}'}(q_i) \neq \emptyset$ for all q_i. It is clear that q_i is incomplete with respect to $\mathcal{A}'$ for all i. Otherwise there would be a component i such that an interaction $\alpha = \{a_i\} \in excl(\mathcal{A}')$ would be enabled in q. This is not possible. Therefore, there can only be a q_i with $need_{\mathcal{A}'}(q_i) = \emptyset$ if $\boldsymbol{en}(q_i) \subseteq \mathcal{A}'$ because this implies $Int_{\mathcal{A}'}(q_i) = \emptyset$ and therefore also $need_{\mathcal{A}'}(q_i) = \emptyset$. In this case, from $need_{\mathcal{A}'}(q_i) = \emptyset$ and from the fact that q_i is reachable from q_i^0 in T_i we conclude $q_i^0 \in BWS(q_i, need_{\mathcal{A}'}(q_i))$. Then q_i is a local state which does not satisfy the first condition in the proposition (the last term in the condition is not satisfied because $Int_{\mathcal{A}'}(q_i) = \emptyset$). This is a contradiction. We conclude $need_{\mathcal{A}'}(q_i) \neq \emptyset$ for all q_i, and we may choose a minimal set K' for q according to Corollary 5.5.1. There are two cases:

1. For all $s \leq l$ there is $k \notin K'$ with $k(\alpha_s) \neq \emptyset$ or $|\alpha_s| = 1$. In the first case Corollary 5.5.3 shows $i(\alpha_s) \neq \emptyset \Rightarrow i(\alpha_s) \not\subseteq \bigcup_{j \in need_{\mathcal{A}'}(q_i)} \boldsymbol{comm}_i(j)$ for all $i \in K'$. If $\alpha_s = \{a_i\}$ for a component $i \in K'$ we also have $i(\alpha_s) \not\subseteq \bigcup_{j \in need_{\mathcal{A}'}(q_i)} \boldsymbol{comm}_i(j)$ because Sys has strongly exclusive communication. Thus, for any $i \in K'$ the projection of σ to i is a path in T_i from q_i^0 to q_i that is only labeled with ports in $\mathcal{A}_i \setminus \bigcup_{j \in need_{\mathcal{A}'}(q_i)} \boldsymbol{comm}_i(j)$. For all $i \in K'$ we get $q_i^0 \in BWS(q_i, need_{\mathcal{A}'}(q_i))$.

   Corollary 5.5.4 shows that for all $i \in K'$ and all $\alpha \in Int_{\mathcal{A}'}(q_i)$ there is $j \in comp(\alpha) \setminus \{i\}$ with $q_j \in PS_{j,\mathcal{A}'}(q_i, \alpha)$. Using the first property of K' and $i \in K'$ we get $comp(\alpha) \subseteq K'$. Therefore $j \in K'$, and consequently $q_j^0 \in BWS(q_j, need_{\mathcal{A}'}(q_j))$. Thus, for all $i \in K'$ and all $\alpha \in Int_{\mathcal{A}'}(q_i)$ there is a component $j \in comp(\alpha) \setminus \{i\}$ with $q_j^0 \in BWS\left(PS_{j,\mathcal{A}'}(q_i, \alpha), \mathfrak{need}_{\mathcal{A}'}(PS_{j,\mathcal{A}'}(q_i, \alpha))\right)$. This is a contradiction to the first condition because we have already seen above that q_i is incomplete with respect to $\mathcal{A}'$ and $q_i^0 \in BWS(q_i, need_{\mathcal{A}'}(q_i))$.

2. Otherwise, let $s_0 \leq l$ be the largest index with $comp(\alpha_{s_0}) \subseteq K'$ and $|comp(\alpha_{s_0})| \geq 2$. For all $s_0 < s \leq l$ there is $h \notin K'$ with $h(\alpha_s) \neq \emptyset$ or $|comp(\alpha_s)| = 1$. As above, we see that for all $i \in K'$ the projection to i of the segment of σ starting in q^{s_0+1} yields a path in T_i that is only labeled with ports $a_i \in \mathcal{A}_i \setminus \bigcup_{j \in need_{\mathcal{A}'}(q_i)} \boldsymbol{comm}_i(j)$. We get $q_i^{s_0+1} \in BWS(q_i, need_{\mathcal{A}'}(q_i))$ for all $i \in K'$.

   Fix $i \in comp(\alpha_{s_0}) \subseteq K'$ and $\alpha' \in Int_{\mathcal{A}'}(q_i)$. According to Corollary

5.5.4, there is $j \in comp(\alpha') \setminus \{i\}$ with $q_j \in PS_{j,\mathcal{A}'}(q_i, \alpha')$. Also fix j. We have $\alpha_{s_0} \downarrow_{\{i,j\}} \in Int \downarrow_{\{i,j\}}$ and $i(\alpha_{s_0}) \subseteq \alpha_{s_0} \downarrow_{\{i,j\}}$. Furthermore, $(q_i^{s_0}, q_j^{s_0})$ is reachable in $Sys \downarrow_{\{i,j\}}$ (Lemma 3.2.2). Because $i \in K'$ we have $q_i^{s_0+1} \in BWS(q_i, need_{\mathcal{A}'}(q_i))$. The first property of K' implies $j \in comp(\alpha') \subseteq K'$. Therefore we also have $q_j^{s_0+1} \in BWS(q_j, need_{\mathcal{A}'}(q_j))$, and because $q_j \in PS_{j,\mathcal{A}'}(q_i, \alpha')$ we further get $BWS(q_j, need_{\mathcal{A}'}(q_j)) \subseteq BWS(PS_{j,\mathcal{A}'}(q_i, \alpha'), \mathfrak{need}_{\mathcal{A}'}(PS_{j,\mathcal{A}'}(q_i, \alpha')))$. Finally, Corollary 5.5.2 shows $need_{\mathcal{A}'}(q_i) \cap comp(\alpha_{s_0}) \neq \emptyset$. This means that we have $i(\alpha_{s_0}) \subseteq \bigcup_{k \in need_{\mathcal{A}'}(q_i)} \mathit{comm}_i(k)$. Altogether this produces:

$$\begin{aligned} i(\alpha_{s_0}) &\subseteq \mathcal{EA}(q_i, need_{\mathcal{A}'}(q_i), PS_{j,\mathcal{A}'}(q_i, \alpha'), \mathfrak{need}_{\mathcal{A}'}(PS_{j,\mathcal{A}'}(q_i, \alpha'))) \\ &= \mathcal{PA}(q_i, \alpha', j, \mathcal{A}') \subseteq \bigcup_{m \in comp(\alpha')} \mathcal{PA}(q_i, \alpha', m, \mathcal{A}') \end{aligned}$$

Repeating this argument for all $\alpha \in Int_{\mathcal{A}'}(q_i)$ we get

$$i(\alpha_{s_0}) \subseteq \bigcap_{\alpha \in Int_{\mathcal{A}'}(q_i)} \bigcup_{m \in comp(\alpha)} \mathcal{PA}(q_i, \alpha, m, \mathcal{A}').$$

Analogously we get

$$k(\alpha_{s_0}) \subseteq \bigcap_{\alpha \in Int_{\mathcal{A}'}(q_k)} \bigcup_{m \in comp(\alpha)} \mathcal{PA}(q_k, \alpha, m, \mathcal{A}')$$

for all other $k \in comp(\alpha_{s_0})$. This is a contradiction to the second condition.

The assumption that q is a reachable state where no $\alpha \in excl(\mathcal{A}')$ is enabled is therefore wrong, and deadlock-freedom is robust with respect to failure of $\mathcal{A}'$ in *Sys*.

The conditions can be checked in time polynomial in the size of *Sys*, because all parameters can be computed by analyzing *Int*, the local behaviors, and the subsystems $Sys \downarrow_{\{i,j\}}$ for interacting components i and j. □

5.5.2 Proofs for Section 5.3.2

We only slightly have to adjust the proofs given in Section 4.6.2.

Lemma 5.3.1. *Let Sys be an interaction system and let $\mathcal{A}_0, \mathcal{A}' \subsetneq \mathcal{A}$ be disjoint subsets of ports.*

1. *For $m \geq 1$ we have*

$$\mathcal{E}^m_{1,\mathcal{A}'} = \{(i,j) \,|j \neq i, 0 \text{ and } excl_j(\mathcal{A}')\backslash excl_j(\mathcal{A}_{r^{m-1}_{1,\mathcal{A}'}(i)}) \text{ is inevitable in } T_j^{excl_j(\mathcal{A}')}\}.$$

2. *Let $1 \leq d' < d \leq n$ and $m \geq 1$ be given and let $i \in \mathcal{V}$ and $j \in \mathcal{V}\backslash\{i,0\}$. If there is an edge $(i,j) \in \mathcal{E}^m_{d',\mathcal{A}'}$, then we also have $(i,j) \in \mathcal{E}^m_{d,\mathcal{A}'}$.*

Proof. We treat the statements separately.

1. The proof is analogous to the proof of the first part of Lemma 4.3.5. We simply replace $Int\downarrow_{\{j\}}$ and $\mathcal{A}_j$ by $excl(\mathcal{A}')\downarrow_{\{j\}}$ respectively $excl_j(\mathcal{A}')$ and $Sys\downarrow_{\{j\}}$ and T_j by $T^{excl(\mathcal{A}')\downarrow_{\{j\}}}_{Sys\downarrow_{\{j\}}}$ respectively $T_j^{excl_j(\mathcal{A}')}$.

2. The proof is analogous to the proof of the second part of Lemma 4.3.5. In the start of the induction we replace $Int\downarrow_{K_{d'}}$ and $Int\downarrow_{K_d}$ by $excl(\mathcal{A}')$ projected to $K_{d'}$ respectively K_d, and we replace $Sys\downarrow_{K_{d'}}$ and $Sys\downarrow_{K_d}$ by $T^{excl(\mathcal{A}')\downarrow_{K_{d'}}}_{Sys\downarrow_{K_{d'}}}$ respectively $T^{excl(\mathcal{A}')\downarrow_{K_d}}_{Sys\downarrow_{K_d}}$. Using a proof by contradiction, in $T^{excl(\mathcal{A}')\downarrow_{K_d}}_{Sys\downarrow_{K_d}}$ we find a cycle σ_d with properties corresponding to the ones stated in the proof of Lemma 4.3.5. Since all interactions on σ_d are in $excl(\mathcal{A}')\downarrow_{K_d}$ and because j participates in σ_d, the projection $\sigma_{d'}$ of σ_d to $K_{d'}$ again results in a cycle in $T^{excl(\mathcal{A}')\downarrow_{K_{d'}}}_{Sys\downarrow_{K_{d'}}}$. Using the same argument that was used in Lemma 4.3.5 from $\sigma_{d'}$ and σ_d we deduce the existence of $\alpha_{d'} \in excl(\mathcal{A}')\downarrow_{K_{d'}} \backslash excl(\mathcal{A}_i)\downarrow_{K_{d'}}$ and $\alpha_d \notin Int\downarrow_{K_d} \backslash excl(\mathcal{A}_i)\downarrow_{K_d}$ with $\alpha_d\downarrow_{K_{d'}} = \alpha_{d'}$ which then result in an interaction $\alpha \in Int$ with $\alpha \cap \mathcal{A}_i = \emptyset$ and $\alpha \cap \mathcal{A}_i \neq \emptyset$ yielding the desired contradiction[5].

   The inductive step follows the same steps (adjusted appropriately) that were taken in the proof of Lemma 4.3.5 where this time we use the inductive hypothesis to show $\mathcal{A}_{r^m_{d',\mathcal{A}'}(i)} \subseteq \mathcal{A}_{r^m_{d,\mathcal{A}'}(i)}$.

□

Lemma 5.3.2. *Let Sys be an interaction system, and let $1 \leq d \leq n$ be given. Let $\mathcal{A}_0, \mathcal{A}' \subsetneq \mathcal{A}$ be disjoint subsets of ports.*

[5]Note that α does not necessarily have to be in $excl(\mathcal{A}')$.

$\mathcal{G}_{d,\mathcal{A}'}$ can be constructed in time polynomial in the size of Sys. An upper bound for the cost is given by

$$O(|Int|^2 n^{d+4} + |Int| n^{d+3} |T_{max}|^d)$$

where $|T_{max}|$ denotes the size of the largest local transition system.

Proof. Since $|excl(\mathcal{A}')| \in O(|Int|)$ and $|T_j^{excl_j(\mathcal{A}')}| \in O(|T_{max}|)$ for all $j \in K$ the upper bound is derived in exactly the same way as in Lemma 4.3.6. □

Proposition 5.3.2. *Let Sys be a deadlock-free interaction system and let $\mathcal{A}_0, \mathcal{A}' \subsetneq \mathcal{A}$ be disjoint subsets of ports. Let deadlock-freedom be robust with respect to failure of $\mathcal{A}'$ in Sys. Let $1 \leq d \leq n$ be given and let $\mathcal{G}_{d,\mathcal{A}'}$ be defined as above.*

If all nodes in $\mathcal{V}$ are reachable from 0 in $\mathcal{G}_{d,\mathcal{A}}$ then $\mathcal{A}_0$ makes progress without $\mathcal{A}'$ in Sys. The condition can be checked in time polynomial in the size of Sys.

Proof. Again the proof is analogous to the proof of Proposition 4.3.2. This time we consider a run σ *without* $\mathcal{A}'$. Using the same arguments as before, the following two facts can be shown by a nested induction over $m \in \mathbb{N}$:

1. If $(i,j) \in \mathcal{E}^m_{d,\mathcal{A}'}$ then i participates infinitely often in σ if j does so.
2. If j is reachable from i in $(\mathcal{V}, \bigcup_{l=1}^m \mathcal{E}^l_{d,\mathcal{A}'})$ then i participates infinitely often in σ if j does so.

The induction follows the same steps that were taken in the proof of Proposition 4.3.2. Replacing $Int \downarrow_{K_d}$ and $T_{Sys_{K_d}}$ by $excl(\mathcal{A}') \downarrow_{K_d}$ respectively $T_{Sys_{K_d}}^{excl(\mathcal{A}')\downarrow_{K_d}}$ in the start of the induction for the first statement, we see that the first statement above holds for $m = 1$. The proof of the second statement is independent of Int and $excl(\mathcal{A}')$ it simply extends the significance derived for an edge in the first statement to a path of arbitrary length. Therefore the start of the induction for the second statement is the same as the corresponding argument in the proof of Proposition 4.3.2.

In the inductive step we replace $\mathcal{A}_{r_d^m(i)}$ by $\mathcal{A}_{r^m_{d,\mathcal{A}'}(i)}$ before reproducing the argument given in the proof of Proposition 4.3.2. For the same reasons as

mentioned above the inductive step for the second statement can be adopted without change.

Using the second statement, the proof of the proposition is now analogous to the proof of Proposition 4.3.2. Using Lemma 5.3.2, we see that the conditions can be checked in polynomial time. □

Chapter 6

Conclusion and Discussion

We recapitulate the results presented in this thesis. We want to see to what extend we have achieved the goals that we formulated in Chapter 1. Furthermore we want to rate the results in the context of the methodologies discussed in Section 1.2.1. Finally, we point out directions for future work.

6.1 Achievements

We have investigated deadlock-freedom and progress in interaction systems. The major achievement consisted of exhibiting sufficient conditions for both properties. Furthermore, we defined robustness of deadlock-freedom with respect to failure of a set $\mathcal{A}'$ of ports respectively progress without $\mathcal{A}'$. We adapted the conditions to also be applicable in situations where $\mathcal{A}'$ fails.

The condition for deadlock-freedom was derived using a restriction of the communication architecture. We first presented results about reachability. Then, we introduced the interaction graph of an interaction system, and we required it to be a tree. Based on this restriction we characterized pairs of states that could possibly be involved in a deadlock. We combined the results in order to exclude the possibility that combinations of these states that might cause a deadlock are reachable. This approach was suitable not only to treat deadlock-freedom but also freedom of local deadlocks. Furthermore, we simplified the results for systems that only allow binary interactions. Besides stating the sufficient conditions, the achievements also lay in the fact that the interaction graph and the results about reachability can be

used in other settings (not necessarily pertaining to deadlock-freedom), as well.

Treating progress we first stated a theorem characterizing progress. Next we went about stating a sufficient condition for progress. This condition is based on a directed graph whose intuitive meaning conveyed which components were needed by others in order to be able to proceed. The definition of the graph involved a parameter d influencing the precision of the criterion on the one hand and the complexity of the construction of the graph on the other hand. We explained how to generalize the graph respectively the criterion based on it by iterating the construction of the edges. Finally, we exemplarily motivated a remedy in cases where the condition is not satisfied. We proposed one way how the information obtained from the condition might be combined with the characterization of progress in order to be able to make a statement about progress even in cases where the criterion fails.

The situation was slightly different as far as the consideration of failure of a set $\mathcal{A}'$ was concerned because we were not able to resort to a suitable notion of how failure of ports might influence a given system. Thus, we first had to *define* the notions of robustness of deadlock-freedom with respect to failure of $\mathcal{A}'$ respectively progress without $\mathcal{A}'$. Having given the formal definitions, they facilitated an adaptation of the conditions presented before for deadlock-freedom and progress because the definitions are akin to the definitions of the original properties. We presented this adaptation in detail for robustness of deadlock-freedom of a tree-like interaction system with respect to failure of $\mathcal{A}'$ and for progress without $\mathcal{A}'$ pointing out that other properties could have been treated likewise.

Next, we want to see whether our results live up to the paradigms that we had intended to follow throughout this thesis. These stated that the conditions should

- be sufficient for the property in question but efficiently checkable,
- be compositional, and
- allow for reuse of the information gathered if the condition is not satisfied.

As stated above, we have clearly abode by the first paradigm in that we exhibited *sufficient* conditions for deadlock-freedom and progress respectively for these properties under the assumption that ports may fail. We have proved the sufficiency of the conditions. Furthermore, we have indeed benefited in terms of efficiency. The conditions can be checked in time polynomial in the size of the system in question. This fact is actually closely related to the second paradigm. The criteria can be checked efficiently precisely because they are compositional. Compositionality is obtained by projecting the global system to subsystems whose analysis is feasible — as opposed to an investigation of the global system — and by then combining the information extracted from the subsystems. Depending on the property in question these subsystems were of size two (for deadlock-freedom) respectively of arbitrary size d (for progress) where d determined the degree of the polynomial bounding the complexity of the checks and therefore also the degree of efficiency. The important point to note is that the polynomial complexity indeed represents an improvement compared to the exponential cost necessary for an analysis of the global state space.

Finally, we want to explain in what ways we adhered to the last of the three paradigms. We did so in various ways. In Section 3.7.1 we stated that a violation of the conditions presented is also always a hint as to where a possible deadlock — if there exists one in the global system — has to be searched for. Often the information gathered may be used to isolate small(er) parts of the global state space of an interaction system which have to be subjected to a direct analysis. In the case of progress of a set $\mathcal{A}_0$ of ports, on the other hand, we motivated a different approach towards reuse of the data collected. There we combined the information obtained from analyzing $\mathcal{G}_d$ with a different condition for progress in order to show that the set components that are reachable from the node 0 in $\mathcal{G}_d$ makes progress — already knowing that these components cannot proceed without $\mathcal{A}_0$. In addition, increasing the variable parameter d presented one possible option to counteract the failure of the condition we presented. At last, note that Chapter 5 can also be seen along the lines of the last paradigm if we interpret it a little more liberally: In this chapter we did not exactly reuse information obtained from checking a condition. Instead, we reused the

existing conditions themselves by only adapting them slightly to the new situation instead of devising completely new ideas.

6.2 Classification of the Results

Next, we discuss where to rank our results in the context of the other methodologies that we mentioned in Section 1.2.1. There we discussed a range of approaches towards establishing properties of concurrent systems. These included model checking and preorder/equivalence checking, various abstraction methods, and testing. Furthermore, there are proof systems (cf. Francez [64], for example) or approaches following the correctness-by-construction paradigm that has already been mentioned in Section 3.7.1. Even though we more or less exclusively stuck to the approach resulting from our paradigms, i.e., we sought for sufficient but efficiently checkable conditions for a given property, we do not want our results to be understood such that this approach is the "silver bullet" with respect to proving properties of component based systems or more generally of concurrent systems. From our point of view, it is not useful to only focus on a limited set of methods all using similar ideas. Instead, we argue that any attempt to investigate concurrent systems should be based on a combination of as many different approaches as possible. The underlying idea of such a combined approach is of course the hope that the respective weaknesses of one technique can be compensated for by the strengths of the others. The more so as some of the techniques cannot even be uniquely attached to a single methodology. For example, we saw that it was also possible to interpret the results presented in Chapter 3 along the lines of the correctness-by-construction paradigm. Similarly, the projection to subsystems (obtained by hiding components) is an abstraction technique. Combining different methodologies to obtain more precise results is of course not a new idea. For example, Broy et al. [39, Part VI] is devoted to techniques which extend the testing approach taken there by investigating how it can be combined with model checking.

In summary, we see the results presented in this thesis as one step towards a better understanding of (properties of) interaction systems. We do not see them as an exclusive technique but as one approach which is equal

among a selection of various techniques that can and should be chosen from and be combined. The possibilities for such combinations are manifold: Concretely, checking the sufficient conditions we presented could be added as a preprocessing to model checking, for example, possibly rendering this second — usually more complex step — unnecessary. Even if the condition is not satisfied, we have already explained, how additional information can be derived which could be incorporated into the process of further investigations and which might help to reduce the complexity of subsequent model checking. If our conditions are to be combined with a testing approach one could similarly attempt to derive very specific test cases from the components and states which violate the conditions we presented. This way it might be possible to find out if the property in question is really violated. Likewise, a system may be constructed trying to maintain a tree-like structure for as long as possible where only those edges which cause cycles in the interaction graph require special attention in the end.

6.3 Future Work

Still a lot remains to be done with regard to both the theoretical groundwork of the formalism and its practical application. We list a few issues that require further research:

Equivalences: In Section 1.2.1 we mentioned behavioral equivalences in the context of the paradigms for establishing properties. In fact, for more or less every formalism for concurrent systems one field of research deals with various notions of equivalences. Such equivalences allow to identify processes that "exhibit the same behavior". This way it is possible to replace equivalent processes, and to transfer statements that have been proven for one process to other equivalent processes. Furthermore, equivalences allow to lump states of a given process together making it possible to consider a smallest possible representative of an equivalence class of processes. Not least with respect to efficiency of investigating such a process this is an interesting feature. Since we have emphasized the importance of deadlock-freedom for interaction systems it is clear that the minimum requirement to be put on such an

equivalence is that it should preserve deadlock-freedom. Thus, trace-equivalence, for example, is not suitable. On the other hand, it is desirable that the equivalence is a congruence with respect to composition, i.e., replacing component i in an interaction system *Sys* by an equivalent component i' should result in an interaction system which is equivalent to *Sys*. Equivalences such as weak bisimulation [107] seem to be a suitable choice. It has to be clarified how such an equivalence can be defined. In particular, it has to be seen how "silent" τ-ports can be introduced into the concept of interaction systems. Lambertz [92] presents some first considerations and results with regard to these questions.

The investigation of appropriate behavioral equivalences is also interesting because as stated in Section 3.7.2 such equivalences might be combined with the results in Chapter 3. If it is possible to hide those parts of the behavior of a component i from component j which are not meant for communication with j, then there would be a notion of "partial behavior of i with respect to j". It might suffice to only investigate the composition of the corresponding partial behaviors of the two components instead of investigating the subsystem consisting of i and j. It has to be resolved in what way the partial behavior of i must be related to T_i. Such an approach can be interpreted according to the requirement mentioned in Section 1.1 that a component should encapsulate its internals. It only makes that part of a component's behavior available to possible communication partners which is relevant for the respective cooperation.

With regard to introducing behavioral equivalences to interaction systems we refer to Arnold [16] (respectively Barros et al. [26, 27] in the context of component systems). A formalism for parallel processes is presented which is quite similar to interaction systems. The communication between the processes is also realized by specification of all allowed synchronizations. In particular, the formalism allows for the use of one port in various synchronizations and, in principle, for synchronizations of arbitrary degree. In practice, though, n-ary com-

munication is enforced, where n is the number of processes, by means of null actions. This is not advisable in the context of component based systems because it requires adaptation of a component's behavior depending on the cooperations it will or will not participate in. On the other hand, Arnold [16] presents some interesting ideas with regard to suitable logics for specification and verification of properties and with regard to equivalences for such processes. It might prove fruitful to try to carry these ideas over to interaction systems.

Tool support: It is essential that the existing theoretical results are supported by a comprehensive tool in order to allow for an automatic check of the various conditions. This way it will be possible to apply our ideas to large (benchmark-)examples and to also present *quantitative* results showing how our approach compares to others and where its strengths and weaknesses lie. As mentioned in Section 3.4.2 an algorithm based on Corollary 3.4.2 has been implemented in the tool PrInSESSA [127] which offers a data structure for interaction systems. Furthermore, this tool offers an implementation of the conditions for deadlock-freedom presented by Majster-Cederbaum et al. [100, 102]. Currently, the tool is being revised in order to provide a new data structure for interaction systems which is based on binary decision diagrams[1] (BDDs) [36, 42, 43]. BBDs allow for an efficient representation of the local transition systems. Further, it is possible to perform operations such as reachability analyses or computation of subsystems on these local transition systems efficiently. Therefore BDDs are well suited for an implementation of the conditions we presented. Particular importance should be attached to realizing the various methods that are necessary efficiently in order to possibly improve the rough bounds that were partly presented in the previous chapters.

Specification and Verification: So far, most research — in this thesis but also in general — in interaction systems has been directed at the investigation of *generic* properties. Of course it is important to know

[1] More precisely an extension of BDDs called Reduced Ordered Binary Decision Diagrams is used.

that a system is deadlock-free, for example. Nonetheless, it is also clear that further investigation is necessary to find out whether the system really does what it is supposed to do, i.e., it is necessary to obtain a well-founded way of specifying *functional* properties of a system and of verifying that a system meets such a specification. Of course, model checking would be an option for verifying interaction systems. There exist translations [24] from interaction systems to the component model of the Vereofy setting [6] and vice versa. Because Vereofy provides a model checker this translation offers one way to incorporate model checking into interaction systems. Contrariwise, it might be possible to apply some of the concepts that have been developed for interaction systems (in this thesis but also elsewhere [100, 102]) in the Vereofy setting and to combine them with the model checking approach taken there. In this regard, it is also interesting in general to see to what extend ideas pertaining to reachability such as the ones from Section 3.2 or the cross-checking techniques by Majster-Cederbaum and Minnameier [100] can be combined with model checking in order to reduce the complexity of computing the reachable state space. It may be possible to verify a property *without* having to explore the complete global state space. Instead, one could try to identify states which violate the specification and to exclude their reachability by means of the techniques above. As stated in Section 1.2.1 a suitable set of equivalences can also be used for verifying that a system meets certain specifications if the specification is also given in terms of a labeled transition system. In this case verification consists of checking whether the system is able to simulate every behavior that can be obtained from the specification. A lot remains to be done with regard to these questions. If nothing else it is necessary to agree upon an adequate logic for the formulas describing the properties to be checked.

Bibliography

[1] Beyond Safety International Workshop. See: `http://cs.nyu.edu/acsys/beyond-safety/`, April 25 - 28 2004. Schloss Ringberg, Germany.

[2] Microsoft Component Object Model. See: `http://msdn.microsoft.com/en-us/library/aa139693.aspx`.

[3] CORBA. See: `http://www.omg.org`.

[4] Enterprise JavaBeans. See: `http://java.sun.com/products/ejb`.

[5] The Unified Modelling Language. See: `http://www.uml.org/`.

[6] Vereofy/TU-Dresden. See: `http://www.vereofy.de`.

[7] P. A. Abdulla, G. Delzanno, and A. Rezine. Automatic Verification of Directory-Based Consistency Protocols . In *Proceedings of LIX Colloquium Reachability Problems'09*, volume 5797 of *LNCS*, pages 36–50. Springer, 2009.

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

[9] B. Alpern and F. B. Schneider. Defining Liveness. *Inf. Process. Lett.*, 21(4):181–185, 1985.

[10] B. Alpern and F. B. Schneider. Recognizing Safety and Liveness. *Distributed Computing*, 2:117–126, 1986.

[11] N. Aoumeur and G. Saake. A component-based Petri net model for specifying and validating cooperative information systems. *Data Knowl. Eng.*, 42(2):143–187, 2002.

[12] K. R. Apt, N. Francez, and W. P. de Roever. A Proof System for Communicating Sequential Processes. *ACM Trans. Program. Lang. Syst.*, 2(3), 1980.

[13] F. Arbab. A Channel-Based Coordination Model for Component Composition. Technical report SEN-R0203, CWI, 2002.

[14] F. Arbab. Reo: A Channel-Based Coordination Model for Component Composition. *Mathematical Structures in Computer Science*, 14(3): 329–366, 2004.

[15] F. Arbab, C. Baier, J. Rutten, and M. Sirjani. Modeling Component Connectors in Reo by Constraint Automata: (Extended Abstract). In *Proceedings of FOCLASA'03*, volume 97 of *ENTCS*, pages 25 – 46. Elsevier, 2004.

[16] A. Arnold. *Finite Transition Systems.* Series in Computer Science. Prentice/Hall International, 1994.

[17] A. Arora and S. S. Kulkarni. Detectors and Correctors: A Theory of Fault-Tolerance Components. In *Proceedings of ICDCS'98*, pages 436–443. IEEE Computer Society, 1998.

[18] C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Paech, J. Wust, and J. Zettel. *Component-Based Product Line Engineering with UML.* Component Software Series. Addison-Wesley Professional, 2001.

[19] C. Atkinson, P. Bostan, D. Brenner, G. Falcone, M. Gutheil, O. Hummel, M. Juhasz, and D. Stoll. Modeling Components and Component-Based Systems in KobrA. In *The Common Component Modeling Example*, volume 5153 of *LNCS*, pages 54–84. Springer, 2008.

[20] P. C. Attie and H. Chockler. Efficiently Verifiable Conditions for Deadlock-Freedom of Large Concurrent Programs. In *Proceedings of VMCAI'05*, volume 3385 of *LNCS*, pages 465–481, 2005.

[21] P. C. Attie and E. A. Emerson. Synthesis of Concurrent Systems with Many Similar Processes. *ACM Trans. Program. Lang. Syst.*, 20(1): 51–115, 1998.

[22] E. Badouel, A. Benveniste, M. Bozga, B. Caillaud, O. Constant, B. Josko, Q. Ma, R. Passerone, and M. Skipper. SPEEDS Metamodel Syntax and Draft Semantics, January 2007. Deliverable D2.1c.

[23] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[24] C. Baier, M. Majster-Cederbaum, C. Lambertz, N. Semmelrock, C. Minnameier, M. Martens, S. Klüppelholz, T. Blechmann, and J. Klein. Reo and Constraint Automata vs. Interaction Systems — A Comparison, 2009. Technical Report to be published.

[25] J. Barnat, L. Brim, I. Cerná, P. Moravec, P. Rockai, and P. Simecek. DiVinE — A Tool for Distributed Verification. In *Proceedings of CAV'06*, volume 4144 of *LNCS*, pages 278–281. Springer, 2006.

[26] T. Barros, R. Boulifa, and E. Madelaine. Parameterized Models for Distributed Java Objects. In *Proceedings of FORTE'04*, volume 3235 of *Lecture Notes in Computer Science*, pages 43–60. Springer, 2004.

[27] T. Barros, L. Henrio, and E. Madelaine. Behavioural Models for Hierarchical Components. Technical Report INRIA Research Report RR-5591, INRIA, Sophia-Antipolis, 2005.

[28] R. Bastide and E. Barboni. Software Components: A Formal Semantics Based on Coloured Petri Nets. In *Proceedings of FACS'05*, ENTCS, pages 57 – 73, 2005.

[29] A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-Time Components in BIP. In *Proceedings of SEFM'06*, pages 3–12. IEEE Computer Society, 2006.

[30] H. Baumeister, F. Hacklinger, R. Hennicker, A. Knapp, and M. Wirsing. A Component Model for Architectural Programming. In *Pro-*

ceedings of FACS'05, volume 160 of *ENTCS*, pages 75–96. Elsevier, 2006.

[31] M. Ben-Ari. *Principles of Concurrent Programming*. Prentice Hall Professional Technical Reference, 1982.

[32] K. Bergner, A. Rausch, M. Sihling, A. Vilbig, and M. Broy. A Formal Model for Componentware. In G. T. Leavens and M. Sitaraman, editors, *Foundations of Component-Based Systems*, pages 189–210. Cambridge University Press, 2000.

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

[34] A. Biere, C. Artho, and V. Schuppan. Liveness Checking as Safety Checking. In *Proceedings of FMICS'02*, ENTCS, 2002.

[35] M. Bozga, O. Constant, B. Josko, Q. Ma, and M. Skipper. SPEEDS Metamodel Syntax and Static Semantics, February 2007. Deliverable D2.1b.

[36] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient Implementation of a BDD Package. In *Proceedings of DAC'90*, pages 40–45. IEEE Computer Society Press, 1990.

[37] L. Brim, I. Černá, P. Vařeková, and B. Zimmerova. Component-Interaction Automata as a Verification-Oriented Component-Based System Specification. In *Proceedings of SAVCBS'05*, pages 31–38. Iowa State University, USA, 2005.

[38] S. Brookes and A. Roscoe. Deadlock Analysis in Networks of Communicating Processes. *Distributed Computing*, 4(4):209–230, 1991.

[39] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, editors. *Model-Based Testing of Reactive Systems*, volume 3472 of *LNCS*, 2005. Springer.

[40] M. Broy, J. Siedersleben, and C. Szyperski. CoCoME Jury Evaluation and Conclusion. In *The Common Component Modeling Example*, volume 5153 of *LNCS*, pages 449–458. Springer, 2008.

[41] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani. The FRACTAL Component Model and its Support in Java. *Softw., Pract. Exper.*, 36(11-12):1257–1284, 2006.

[42] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

[43] R. E. Bryant. Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[44] L. Bulej, T. Bureš, T. Coupaye, M. Děcký, P. Ježek, P. Parízek, F. Plášil, T. Poch, N. Rivierre, O. Šerý, and P. Tůma. CoCoME in Fractal. In *The Common Component Modeling Example*, volume 5153 of *LNCS*, pages 357–387. Springer, 2008.

[45] T. Bures, P. Hnetynka, and F. Plasil. SOFA 2.0: Balancing Advanced Features in a Hierarchical Component Model. In *Proceedings of SERA'06*, pages 40–48, 2006.

[46] I. Černá, P. Vařeková, and B. Zimmerova. Component-Interaction Automata Modelling Language. Technical Report FIMU-RS-2006-08, Masaryk University, Faculty of Informatics, 2006.

[47] R. C. Cheung. A User-Oriented Software Reliability Model. *IEEE Trans. Software Eng.*, SE-6(2):118–125, 1980.

[48] S. Chouali, M. Heisel, and J. Souquières. Proving Component Interoperability with B Refinement. In *Proceedings of FACS 05*, volume 160 of *ENTCS*, pages 67–84, 2006.

[49] E. M. Clarke and E. A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic. In *Proceedings of Logics of Programs'81*, volume 131 of *LNCS*, pages 52–71. Springer, 1982.

[50] E. M. Clarke, T. Filkorn, and S. Jha. Exploiting Symmetry In Temporal Logic Model Checking. pages 450–462.

[51] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[52] E. M. Clarke, D. E. Long, and K. L. McMillan. Compositional Model Checking. In *Proceedings of LICS'89*, pages 353–362. IEEE Computer Society, 1989.

[53] R. Cleaveland and S. A. Smolka. Strategic Directions in Concurrency Research. *ACM Comput. Surv.*, 28(4):607–625, 1996.

[54] R. Cleaveland and O. Sokolsky. *Equivalence and Preorder Checking for Finite-State Systems*, chapter 6, pages 391–424. Elsevier, 2001.

[55] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proceedings of POPL'77*, pages 238–252, 1977.

[56] L. de Alfaro and T. A. Henzinger. Interface Automata. In *Proceedings of FSE'01*, pages 109–120, 2001.

[57] M. S. Deutsch and R. R. Willis. *Software Quality Engineering: A Total Technical and Management Approach*. Series in software engineering. Prentice Hall, 1988.

[58] E. Dijkstra and C. Scholten. A Class of Simple Communication Patterns, EWD643. In E. Dijkstra, editor, *Selected Writings on Computing*, pages 334–337. Springer, 1982.

[59] E. W. Dijkstra. Cooperating sequential processes. In F. Genuys, editor, *Programming Languages: NATO Advanced Study Institute*, pages 43–112. Academic Press, London, 1968. Originally appeared as Tech. Rep. EWD-123, Technical University of Eindhoven, the Netherlands, 1965.

[60] E. W. Dijkstra. Hierarchical Ordering of Sequential Processes. *Acta Inf.*, 1:115–138, 1971.

[61] A. Dimov and S. Punnekkat. On the Estimation of Software Reliability of Component-Based Dependable Distributed Systems. In *Proceedings of QoSA/SOQUA'05*, volume 3712 of *Lecture Notes in Computer Science*, pages 171–187. Springer, 2005.

[62] C. Ellis. Team Automata for Groupware Systems. In *Proceedings of GROUP '97*, pages 415–424. ACM, 1997.

[63] E. A. Emerson and A. P. Sistla. Symmetry and Model Checking. In *Proceedings of CAV'93*, volume 697 of *LNCS*, pages 463–478. Springer, 1993.

[64] N. Francez. *Program Verification*. Addison-Wesley, 1992.

[65] R. Gawlick, R. Segala, J. F. Søgaard-Andersen, and N. A. Lynch. Liveness in Timed and Untimed Systems. Technical Report MIT/LCS/TR-587, Massachusetts Institute of Technology, Cambridge, 1993.

[66] R. Gawlick, R. Segala, J. F. Søgaard-Andersen, and N. A. Lynch. Liveness in Timed and Untimed Systems. In *Proceedings of ICALP '94*, volume 820 of *LNCS*, pages 166–177. Springer-Verlag, 1994.

[67] P. Godefroid. Using Partial Orders to Improve Automatic Verification Methods. In *Proceedings of CAV'90*, volume 531 of *LNCS*, pages 176–185. Springer, 1990.

[68] P. Godefroid and P. Wolper. Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties. *Form. Methods Syst. Des.*, 2(2):149–164, 1993. ISSN 0925-9856.

[69] G. Gössler. Component-based Design of Heterogeneous Reactive Systems in Prometheus. Technical report 6057, INRIA, December 2006.

[70] G. Gössler and J. Sifakis. Component-Based Construction of Deadlock-Free Systems. In *Proceedings of FSTTCS'03*, volume 2914 of *LNCS*, pages 420–433, 2003.

[71] G. Gössler and J. Sifakis. Composition for component-based modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005.

[72] G. Gössler and J. Sifakis. Composition for Component-Based Modeling. In *Proceedings of FMCO'02*, volume 2852 of *LNCS*, pages 443–466, 2003.

[73] G. Gössler, S. Graf, M. Majster-Cederbaum, M. Martens, and J. Sifakis. Ensuring Properties of Interaction Systems. In *Program Analysis and Computation, Theory and Practice*, volume 4444 of *LNCS*, pages 201–224, 2007.

[74] G. Gössler, S. Graf, M. Majster-Cederbaum, M. Martens, and J. Sifakis. An Approach to Modelling and Verification of Component Based Systems. In *Proceedings of SOFSEM 07*, volume 4362 of *LNCS*, pages 295–308, 2007.

[75] M. G. Gouda. Closed Covers: To Verify Progress for Communicating Finite State Machines. *IEEE Trans. Software Eng.*, 10(6):846–855, 1984.

[76] B. Hamid, A. Radermacher, A. Lanusse, C. Jouvray, S. Gérard, and F. Terrier. Designing Fault-Tolerant Component Based Applications with a Model Driven Approach. In *Proceedings of SEUS'08*, volume 5287 of *LNCS*, pages 9–20. Springer, 2008.

[77] B. Hamid, A. Radermacher, P. Vanuxeem, A. Lanusse, and S. Gérard. A Fault-tolerance Framework for Distributed Component Systems. In *Proceedings of SEAA'08*, pages 84–91. IEEE, 2008.

[78] D. Harel. Statecharts: A Visual Formulation for Complex Systems. *Sci. Comput. Program.*, 8(3):231–274, 1987.

[79] R. Hennicker, S. Janisch, and A. Knapp. On the Observable Behaviour of Composite Components. In C. Canal and C. Pasareanu, editors, *Proceedings of FACS'08*, ENTCS, pages 1–26. Elsevier, 2008.

[80] T. A. Henzinger. Masaccio: A formal model for embedded components. In *Proceedings of the First IFIP International Conference on Theoretical Computer Science*, pages 549–563, 2000.

[81] M. P. Herlihy and J. M. Wing. Specifying Graceful Degradation. *IEEE Trans. Parallel Distrib. Syst.*, 2(1):93–104, 1991.

[82] S. Herold, H. Klus, Y. Welsch, C. Deiters, A. Rausch, R. Reussner, K. Krogmann, H. Koziolek, R. Mirandola, B. Hummel, M. Meisinger, and C. Pfaller. CoCoME — The Common Component Modeling Example. In *The Common Component Modeling Example*, volume 5153 of *LNCS*, pages 16–53. Springer, 2008.

[83] C. A. R. Hoare. *Communicating Sequential Processes*. Series in Computer Science. Prentice/Hall International, 1985.

[84] P. Inverardi and S. Uchitel. Proving Deadlock Freedom in Component-Based Programming. In *Proceedings of FASE'01*, volume 2029 of *LNCS*, pages 60–75, 2001.

[85] P. Inverardi, A. L. Wolf, and D. Yankelevich. Static Checking of System Behaviors Using Derived Component Assumptions. *ACM Trans. Softw. Eng. Methodol.*, 9(3):239–272, 2000.

[86] P. Jalote. *Fault Tolerance in Distributed Systems*. Prentice Hall, 1998.

[87] P. C. Kanellakis and S. A. Smolka. CCS Expressions, Finite State Processes, and Three Problems of Equivalence. *Inf. Comput.*, 86(1), 1990.

[88] E. Kindler. Safety and Liveness Properties: A Survey. *EATCS-Bulletin*, 53:268–272, 1994.

[89] J. C. Knight, E. A. Strunk, and K. J. Sullivan. Towards a Rigorous Definition of Information System Survivability. In *Proceedings of DISCEX (1)'03*, pages 78–89. IEEE Computer Society, 2003.

[90] J. Kofron. Extending Behavior Protocols With Data and Multisynchronization. Technical Report 2006/10, Dep. of SW Engineering, Charles University in Prague, 2006.

[91] D. Kozen. Results on the Propositional μ-Calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[92] C. Lambertz. Exploiting Architectural Constraints and Branching Bisimulation Equivalences in Component-Based Systems. 2009. To be published as a technical report of Eindhoven University of Technology.

[93] L. Lamport. Proving the Correctness of Multiprocess Programs. *IEEE Trans. Software Eng.*, 3(2):125–143, 1977.

[94] K.-K. Lau and Z. Wang. Software Component Models. *IEEE Trans. Software Eng.*, 33(10):709–724, 2007.

[95] P. A. Lee and T. Anderson. *Fault Tolerance Principles and Practice*, volume 3 of *Dependable Computing and Fault-Tolerant Systems*. Springer Verlag, second, revised edition, 1990.

[96] N. A. Lynch and M. R. Tuttle. An Introduction to Input/Output Automata. *CWI-Quarterly*, 2(3):219–246, 1989.

[97] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer. Specifying Distributed Software Architectures. In W. Schafer and P. Botella, editors, *Proceedings of ESEC'95*, volume 989, pages 137–153, Sitges, Spain, 1995. Springer-Verlag, Berlin.

[98] M. Majster-Cederbaum and M. Martens. Robustness in Interaction Systems. In *Proceedings of FORTE'07*, volume 4574 of *LNCS*, pages 325–340, 2007.

[99] M. Majster-Cederbaum and C. Minnameier. Deriving Complexity Results for Interaction Systems from 1-safe Petri Nets. In *Proceedings of SOFSEM'08*, volume 4910 of *LNCS*, pages 352–363. Springer, 2008.

[100] M. Majster-Cederbaum and C. Minnameier. Cross-Checking — Enhanced Over-Approximation of the Reachable Global State Space of Component-based Systems. In *Proceedings of the LIX Colloquium on Reachability Problems'09*, 2009. accepted for publication.

[101] M. Majster-Cederbaum and C. Minnameier. Everything is PSPACE-complete in Interaction Systems. In *Proceedings of ICTAC'08*, volume 5160 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2008.

[102] M. Majster-Cederbaum, M. Martens, and C. Minnameier. A Polynomial-Time Checkable Sufficient Condition for Deadlock-Freedom of Component Based Systems. In *Proceedings of SOFSEM'07*, volume 4362 of *LNCS*, pages 888–899, 2007.

[103] M. Martens and M. Majster-Cederbaum. Compositional Analysis of Tree-Like Component Architecures. In *Proceedings of EMSOFT'08*, pages 199–206. ACM, 2008.

[104] M. Martens and M. Majster-Cederbaum. Using Architectural Constraints for Deadlock-Freedom of Component Systems with Multiway Cooperation. In *Proceedings of TASE'09*, pages 225–232. IEEE Conference Publishing Services, 2009.

[105] V. Matena, S. Krishnan, L. DeMichiel, and B. Stearns. *Applying Enterprise JavaBeans 2.1: Component-Based Development for the J2EE Platform*. Addison-Wesley Professional, 2nd edition, 2003.

[106] D. Messerschmitt and C. Szyperski. *Software Ecosystem — Understanding an Indispensable Technology and Industry*. MIT Press, 2003.

[107] R. Milner. *Communication and Concurrency*. Prentice/Hall, 1989.

[108] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Parts I and II. *Information and Computation*, 100(1):1 – 77, 1992.

[109] S. Moschoyiannis and M. W. Shields. Component-Based Design: Towards Guided Composition. In *Proceedings ACSD'03*, pages 122–131. IEEE Computer Society, 2003.

[110] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[111] G. J. Myers. *The Art of Software Testing*. Wiley, revised 2nd edition, 2004.

[112] O. Nierstrasz and F. Achermann. A Calculus for Modeling Software Components. In *Proceedings of FMCO'02*, volume 2852 of *LNCS*, pages 339–360, 2003.

[113] S. S. Owicki and L. Lamport. Proving Liveness Properties of Concurrent Programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, 1982.

[114] R. Paige and R. E. Tarjan. Three Partition Refinement Algorithms. *SIAM J. Comput.*, 16(6), 1987.

[115] P. Parizek and F. Plasil. Modeling Environment for Component Model Checking from Hierarchical Architecture. In *Proceedings of FACS'06*, volume 182 of *ENTCS*, pages 139–153. Elsevier, 2007.

[116] D. Peled. All from One, One for All: on Model Checking Using Representatives. In *Proceedings of CAV'93*, volume 697 of *LNCS*, pages 409–423. Springer, 1993.

[117] C. A. Petri. *Kommunikation mit Automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. English translation available as *Communication with Automata*, New York: Griffiss Air Force Base, Technical Report RADC-TR-65–377, Vol. 1, Pages: Suppl. 1.

[118] F. Plasil and S. Visnovsky. Behavior Protocols for Software Components. *IEEE Transactions on Software Engineering*, 28(11):1056–1076, 2002.

[119] F. Plasil, D. Balek, and R. Janecek. SOFA/DCUP: Architecture for Component Trading and Dynamic Updating. In *Proceedings of ICCDS'98*. IEEE Computer Society, 1998.

[120] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of FOCS'77*, pages 46–57. IEEE, 1977.

[121] A. Pnueli, N. Shankar, and E. Singerman. Fair Synchronous Transition Systems and Their Liveness Proofs. In *Proceedings of FTRTFT'98*, volume 1486 of *Lecture Notes in Computer Science*, pages 198–209. Springer-Verlag, 1998.

[122] A. Pretschner and M. Leucker. Model-Based Testing — A Glossary. In *Model-Based Testing of Reactive Systems*, volume 3472 of *LNCS*, pages 607–609. Springer, 2005.

[123] R. Reussner, S. Becker, J. Happe, H. Koziolek, K. Krogmann, and M. Kuperberg. The Palladio Component Model. Technical Report Interner Bericht 2007-21, Universität Karlsruhe, 2007.

[124] R. H. Reussner, H. W. Schmidt, and I. Poernomo. Reliability Prediction for Component-Based Software Architectures. *Journal of Systems and Software*, 66(3):241–252, 2003.

[125] T. Saridakis. Graceful Degradation for Component-Based Embedded Software. In *Proceedings of IASSE'04*, pages 175–182. ISCA, 2004.

[126] T. Saridakis. Surviving Errors in Component-Based Software. In *Proceedings of EUROMICRO-SEAA'05*, pages 114–125. IEEE Computer Society, 2005.

[127] R. Schaube. Effiziente Überapproximation des globalen Zustandsraums komponenten-basierter Systeme durch Cross-Checking in Subsystemen. Master's thesis, University of Mannheim, 2008.

[128] N. Semmelrock and M. Majster-Cederbaum. Reachability in Tree-Like Component Systems is PSPACE-Complete. In *Proceedings of FACS'09*, 2009. accepted for publication.

[129] C. P. Shelton and P. Koopman. Improving System Dependability with Functional Alternatives. In *Proceedings of DSN'04*, pages 295–304. IEEE Computer Society, 2004.

[130] A. P. Sistla. On Characterization of Safety and Liveness Properties in Temporal Logic. In *Proceedings of PODC '85*, pages 39–48. ACM, 1985.

[131] I. Sommerville. *Software Engineering*, chapters 11 and 12. Pearson Education Limited, 8th edition, 2007.

[132] R. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[133] R. Tarjan. Enumeration of the Elementary Circuits of a Directed Graph. *SIAM Journal on Computing*, 2(3):211–216, 1973.

[134] M. H. ter Beek, C. A. Ellis, J. Kleijn, and G. Rozenberg. Synchronizations in Team Automata for Groupware Systems. *Computer Supported Cooperative Work*, 12(1):21–69, 2003.

[135] E. Troubitsyna. Developing fault-tolerant control systems composed of self-checking components in the action systems formalism. In H. D. Van and Z. Liu, editors, *Proceedings of the Workshop on Formal Aspects of Component Software FACS'03*, Pisa, Italy, Sep 2003.

[136] M. van der Bijl and F. Peureux. I/O-automata Based Testing. In *Model-Based Testing of Reactive Systems*, volume 3472 of *LNCS*, pages 173–200. Springer, 2004.

[137] B. Zimmerova, P. Vareková, N. Benes, I. Cerná, L. Brim, and J. Sochor. Component-Interaction Automata Approach (CoIn). In *The Common Component Modeling Example*, volume 5153 of *LNCS*, pages 146–176. Springer, 2008.

Index