Model checking of component connectors

Izadi, M.

1 Introduction

The concept of *component based systems*, especially component-based software, is a new philosophy or way of thinking to deal with the complexity in designing large scale computing systems[13, 135]. One of the main goals of this approach is to compose reusable components by some glue codes. The model according to which these *components* are composed is called *coordination model*. *Coordination languages* are specification languages for coordination models. Reo is a coordination language which is based on a calculus of channels [13]. By using Reo specifications, complex component connectors can be organized as a network of channels and built in a compositional manner.

In this thesis, we investigate the formal verification of properties of Reo coordination models. The main question we address is: how can the desired properties of a coordination model specified in Reo be *formally* verified. In particular, the problem is interesting if the state space of the model is very large. To answer this question, we investigate an automata-theoretic model checking method for Reo specifications in the presence of the state explosion problem. In this way we first propose a formal semantics of Reo based on a generalization of the standard notion of Büchi automata and containing some ideas from constraint automaton (the first operational model for Reo [30]). In the following section, we introduce the context of this research and the main questions more precisely. Also, in the forthcoming sections, we introduce our main thesis and its motivations and a history of the research in this field. In the last section of this chapter, we introduce the outline of the thesis and for each chapter its contributions.

1.1 Research Context and Main Question

The work of this thesis is categorized under the computer science fields of *formal verification* and *coordination systems*. In fact, this thesis introduces a formal verification framework for coordination systems in particular and for *component-based systems* in general. Let us introduce these fields briefly.

A system that consists of a set of independent computing components and a coordinating subsystem is called a component based system. Coordination is defined as the study of the dynamic topologies of interactions among concurrent programs, processes and components of a system, with the goal of finding solutions to the problem of managing these interactions [12]. To be more precise about the coordination systems, we need to model or express the coordination strategies using some kind of modeling formalism or language. There exist many coordination models and languages in the literature [120]. In this thesis we concentrate on the coordination language *Reo*, that is, an exogenous coordination language which can specify coordination of a set of components through networks of channels or compositional connectors built out of primitive ones [13].

The main goal of verification methods is trying to ascertain that an actual system or program satisfies its requirements. In *formal* verification one tries to achieve the aim of verification by describing the system using a mathematical model, expressing the requirements as properties of this model and by showing through rigorous mathematical reasoning that the model of the system indeed has the required properties [50, 110]. There are two main methods for formal verification: *deductive verification* (theorem proving) and *model checking*. In de-

ductive verification, the system to be proved correct is described by a set of formulas called *axioms*. The process of verification is to derive a proof of the desired correctness property based on the set of axioms. In model checking, the system to be proved correct is modeled by a kind of finite transition system and the desired property is expressed by a formula in a formal language such as the language of a temporal logic. The process of verification is to check all of the state space of the model for the satisfaction of the property's formula. In this thesis we will consider verification methods that fall into the field of model checking, because they can be fully automated and hence more suitable for an implementation.

Thus, our main question or goal is *to find a model checking verification framework for coordination systems specified by Reo*. For this aim, the coordination systems specified by Reo should be modeled by some sort of an operational (transition systems-based) model. There are a number of operational semantic models for Reo, such as constraint automata [30], intentional automata [52] and Reo automata [36, 37]. However, these models have shortcomings in fully expressing some aspects of coordinations such as synchronization of I/O operations, context dependency, and fairness constraints or are not very suitable for model checking. Therefore, we present a new operational semantics for Reo called *Büchi automata of records (BAR)* and its augmented versions.

When the modeling formalism is a sort of automata on infinite objects (such as Büchi automata on infinite strings which is our selected formalism), the most suitable model checking method is that of *automata theoretic model checking*. In this method, the negation of the desired property is expressed by an automaton (directly or after translation from formulas of a temporal logic such as LTL) and the emptiness of the language of the (intersection) product of the two automata (system and property automata) is checked.

The model checking process can suffer from the problem of *state explosion* since the model of the system tends to be extremely large. We select two independent methods to tackle this problem. The first is to implement the state space symbolically using ordered binary decision diagrams (OBDD) and running the model checking algorithm over them, a method called *symbolic model checking*. The other is to minimize the models based on some proper equivalence relations, called *compositional minimization*. Also, when we obtain the property automaton from translation of linear temporal logic formulas, the translation can be done not only inductively but also by using an *on-the-fly* method.

1.2 This Thesis

In this thesis, we present a framework for automata theoretic model checking of coordination systems specified in Reo. As an operational modeling formalism that covers several intended behaviors of Reo connectors such as fairness, I/O synchronization, and context dependency, we introduce Büchi automata of records (BAR) and their augmented version (ABAR). We show that every constraint automaton (the first introduced operational semantics of Reo) can be translated into an essentially equivalent BAR. However, there are some Reo connectors' behaviors expressible by BAR's that constraint automata are not able to express.

To specify the properties to be verified, we introduce an action based linear temporal

logic called ρ-LTL interpreted over the executions of augmented Büchi automata of records and show how its formulas can be translated into their equivalent ABAR's. The translation can be done inductively or by using an on-the-fly method. To deal with the large state spaces, we show that ABAR's can be implemented using ordered binary decision diagrams (OBDD) as their dense data structures. For this purpose, we also introduce the necessary modifications over the basic model checking algorithm that can be applied directly over OBDD structures. The implementation and case studies show the applicability of our method over large state spaces.

We also show that the state explosion problem can be tackled by compositional minimization methods using some suitable equivalence relations. To this aim, we show that two failure based equivalence relations called CFFD and NDFD are congruence relations with respect to product and hiding operators of constraint automata. Therefore, based on the congruency results and because of the linear time temporal logic preservation properties of CFFD and NDFD equivalences and their minimality properties, they can be used for compositional minimization of constraint automata models in the field of model checking. The method is applied on some practical case studies.

1.3 Related Work

Reo [13] is a coordination language based on connectors for the orchestration of components in a component based system. Primitive connectors such as synchronous channels or FIFO queues are composed to build circuit-like component connectors which exhibit complex behavior and play the role of glue code in exogenously coordinating the components to produce a system.

In contrast to many connector languages for components that focus on stateless connectors in a control flow setting (e.g. BIP [33]), Reo generalizes dataflow networks and Kahn networks because it allows to express behavior including state-based, context dependent, multi-party synchronization and mutual exclusion. The original description of Reo was purely informal [13] and no formal semantics for it existed. Subsequently, a number of models were developed to capture the desired behavior of Reo connectors and of their composition. These include models based on constraint automata [30], timed data streams (also known as abstract behavioral types) [27], connector colouring [47], structural operational semantics [116], linear logic [46] and intentional automata [52] . None of these models, however, is entirely satisfactory. Timed data streams model the possible data flow of a network, but because of their declarative nature they have no support for model checking. All other models are more operational and more suitable for analysis techniques, but either they do not give the desired semantics for certain connectors, or they suffer from technical problems such as not being able to give semantics to all connectors, or both.

Constraint automata [30] are acceptors of timed data streams, but are much more concrete and suitable for model checking analysis. A constraint automaton is a labeled transition system in which each transition label contains two parts: a set N of port names that are *synchronized* if the transition is taken and a proposition g on the data. The latter acts as constraint

on data that can be communicated through the ports in N. The data flowing through the ports in N is *mutually exclusive* with respect to any communication by a port not in N.

Two specific shortcomings of modeling Reo by constraint automata, for example, are that it cannot model desired fairness constraints and it cannot model operations that depend upon pending I/O operations on the communication ports of a connector. This latter feature is called *context dependency*, which occurs when the behavior of a connector can change depending upon not only the presence of requests on a connector boundary, but also on their absence. In such cases, the behavior of a connector can change dramatically with changing context. Both connector coloring and Reo automata [37] address the context dependency issue, but connector coloring does not include a description of the temporal unfolding of a Reo connector, and Reo automata do not address fair behaviors. Both models are incomplete in that they cannot give semantics to many reasonable connectors.

Because Reo is one of the most recently proposed coordination languages, there are only a few works on formal verification of the properties of coordination systems specified in Reo. Selecting the formal verification techniques depends on the choice of the formal semantics of Reo. Algorithms for verifying Reo specifications on the basis of their constraint automata semantics have been presented in [30] for checking (bi)simulation and language equivalence and in [17, 18, 45] for temporal logic specifications. In [17, 18] a timed version of constraint automaton was presented and the problem of model checking of timed CTL for it was considered. In [88, 89], timed constraint automata also have been considered as the modeling formalism in presenting a SAT-based approach for bounded model checking of real-time component connectors. The main theme of the work presented in [44, 45] is reasoning about the reconfigurability of Reo networks using CTL like temporal logics and model checking techniques. Also, there is a work on symbolic model checking of a CTL-like temporal logic (called *BTSL* temporal logic) for constraint automata using ordered binary decision diagrams (OBDD) [99, 98]. The implemented tool based on this work is called *Vereofy* [7]. The common features of all of the above mentioned works on verification of Reo networks and constraint automata are: 1- They suppose that all components of the whole system that we need to verify, can be modeled by constraint automata and 2- The temporal logics they use for specification of the properties are branching time. The reconfigurability of Reo networks through algebraic graph transformations and their model checking using the behavioral specification language mCRL2 [61], based on their constraint automata semantics have been considered in [95] and [91, 94].

Compositional verification has been used in a variety of different ways in the analysis of models of concurrency. Clarke et al. in [51] used interface processes to model the environment for a component. They modeled systems as finite transition systems and used CTL to specify their properties. There are some works on compositional verification of systems modeled by I/O automata [106, 117]. Failure based equivalence checking is a technique for compositional verification in which, the goal is to construct a reduced state space that is equivalent to the full state space in the sense of some process-algebraic equivalences (for more theory and references see [139, 140]). There are some experiments on compositional minimization of state spaces using failure based equivalences, such as [105, 86, 141, 142]. One of the main common features of all of these works is that all components of the actual system should be modeled by a general labeled transition system and there is no distinction between the components and the connectors and their properties which we need to verify. For more details

on this method of verification and also some of the experimental results see [140].

There are some tools for describing and then minimizing labeled transition systems, such as the ARA tool set and its most recent version TVT designed in Tampere University of Technology in Finland [151]. One of the most useful tool sets for this purpose is CADP designed in INRIA, France [1]. It contains many useful components for modeling, analysis and minimization of labeled transition systems and it is free for academic use. For the purpose of model checking, there are some tool sets that are specially successfully used in analysis of real systems, like NuSMV [4] and Spin [6]. The NuSMV system is a tool for checking finite state systems against their specifications in the temporal logics LTL and CTL. It uses the symbolic model checking technique on the ordered binary decision diagrams (OBDDs). Spin is a widely distributed software package that supports the formal verification of distributed systems. It uses a high level language for specifying systems descriptions, called Promela, and LTL is its specification language.

1.4 Thesis Outline, Contributions, and Results

This thesis proceeds as follows:

- In the last section of this chapter, we review our own research leading to this thesis. In this way, we introduce the publications on which this thesis is based.

- In chapter 2, we introduce the context and the background of the thesis. We briefly introduce the notions of component based systems and coordination. The problem of formal verification of reactive systems and its solution methods, namely, deductive verification, model checking and their combinations are introduced. Also, the method of automata theoretic model checking and a set of advanced techniques of model checking to deal with the problem of state explosion, including on-the-fly and symbolic model checking and equivalence based compositional reduction, are presented. In addition, we introduce the framework of temporal logics in the context of the linear or branching time views and explain shortly the reason for selecting the linear time view as our logical framework in this thesis.

- In chapter 3, we describe the Reo coordination language and the theory of constraint automata as an operational semantics for Reo that is suitable for model checking. In this chapter, we also briefly describe the other semantic formalisms that have been introduced for Reo, including co-algebraic models, connector coloring, intentional automata, guarded and Reo automata, process algebraic and structural operational semantics.

- In chapter 4, we introduce Büchi automata of records and unconditional fair constraint automata as alternative models for the operational semantics of Reo. We compare their expressiveness with respect to the original model of constraint automata discussed in the previous chapter. In addition, we review some shortcomings of constraint automata and of their timed data streams based semantics in modeling component connectors and

motivate the use of records and Büchi automata of records as operational semantics for Reo:

  - We introduce *records* as data structures for modeling the simultaneous executions of events: ports in the domain of the record are allowed to communicate simultaneously the data assigned to them, while ports not in the domain of the record are blocked so that no communication can happen. The behavior of a network of components is given in terms of (infinite) sequences of records, to specify the order of occurrences of the events. In addition, streams and languages of streams of records are introduced.
  - We give a bidirectional translation of TDS-languages (that are used as the semantics of constraint automata) and record-based languages.
  - The notion of *Büchi automaton of records (BAR)* is introduced and it is shown that each constraint automaton can be translated into a Büchi automaton of records.
  - We show that BAR's can be used to model Reo connectors, in particular connectors with some fairness conditions on their behaviors, using some examples. This proves that BAR's are semantically more expressive than constraint automata.
  - We introduce a join composition operator for Büchi automata on streams of records and show that it is correct with respect to the join operator for constraint automata.
  - Also, we present a method to recast the join operation on BAR's using the standard product operator of Büchi automata.
  - We introduce a more expressive version of constraint automaton, called *fair constraint automaton*, whose syntax is the same as constraint automaton but now with final (accepting) states and its semantics is based on the languages of streams of records.

- In chapter 5, in order to address context-dependent behaviors, we extend our BAR models with the possibility of testing if some ports of the environment are ready to communicate or not. That is, we consider a Büchi variant of Kozen's finite automata on guarded strings [100] which we call *augmented Büchi automata of records (ABAR)*. Our model has an advantage over previous models in that it covers the basic concepts of Reo as well as the context sensitive behavior within a standard automata theoretical framework. The benefits are a clear and easy notation for the representation of a component connector, as well as efficient existing tool support for automatic analysis. The chapter introduces the following notions and results:

  - We introduce augmented Büchi automata of records (ABAR) as acceptors of infinite guarded strings of records.
  - We show that in addition to the fairness constraints, the context-dependent behavior of Reo connectors can be modeled using ABAR.
  - We introduce a join composition operator for augmented Büchi automata on streams of records.

  - Also, we present a method to recast the join operation on ABAR's using the standard product operator of Büchi automata.
  - We introduce a context dependent version of fair constraint automaton, called *augmented fair constraint automaton*, with the same syntax as constraint automaton but now it has final (accepting) states and labels on states and with a semantics based on languages of infinite guarded strings of records.

- In chapter 6, we present an automata theoretic method of model checking for coordination systems modeled by ABAR's. For this aim, we follow the following steps:
  - First, we introduce an action (or transition) based linear temporal logic (called ρLTL) interpreted over computations of ABAR's.
  - Then, we show that ρLTL formulas can be translated into ABAR's using an inductive translation method.
  - Also, we present an on-the-fly method to translate ρLTL formulas into ABAR's.

- In chapter 7, we introduce the main theoretical and practical concepts we used to implement a BDD based model checking tool for Reo specifications. This implementation is based on the augmented Büchi automata of records semantic models introduced in the previous chapters. Moreover, this tool accepts properties expressed in the ρLTL linear temporal logic as input and verifies the Reo specification against these properties. The chapter also presents some case studies.

- In chapter 8, we investigate the method of *equivalence based compositional minimization* of models of Reo to deal with the state explosion problem in model checking. In this method components of a system are reduced with respect to an equivalence relation before building the complete system [60, 50, 79, 82]. An equivalence relation should have two properties in order to be useful in the equivalence based compositional reduction method: it should *preserve* the class of properties to be verified and it should be a *congruence* with respect to the syntactic operators that are used to compose the components of the model. By congruence relation we mean that the replacement of a component of a model by an equivalent one should always yield a model that is equivalent with the original one. Fortunately, in the context of compositional failure based semantic models of process description languages such as CCS and LOTOS, there are two equivalence relations, called CFFD and NDFD [86, 141, 142], that have a significant property: CFFD-equivalence preserves the fragment of linear time temporal logic that has no next-time operator and has an extra operator for distinguishing deadlocks [141, 142]; and NDFD-equivalence preserves linear time temporal logic without next-time operator [86]. It was also shown that CFFD and NDFD are the minimal equivalences preserving the above mentioned fragments of linear time temporal logic with respect to all composition operators of CCS and LOTOS [86]. Thus, if we use CCS-like composition operators, CFFD and NDFD equivalences can be suitable equivalences for using in the context of equivalence based compositional reduction method.

  In chapter 8, we investigate the above mentioned results for the case of constraint automata. In other words, we consider the failure based semantics for constraint automata

as labeled transition systems with compound labels, instead of their timed data streams-based semantics. Thus, we follow the following steps:

  - First we define CFFD and NDFD equivalences for the case of constraint automata.
  - We show that the temporal logic preservation results hold in the case of constraint automata.
  - We consider the congruency results and prove that:
    * The failure-based equivalence relation CFFD is a congruence with respect to the join operator of constraint automata.
    * The failure-based equivalence relation CFFD is a congruence with respect to the hiding operator of constraint automata.
    * The failure-based equivalence relation NDFD is a congruence with respect to the join operator of constraint automata.
    * The failure-based equivalence relation NDFD is a congruence with respect to the hiding operator of constraint automata.
  - We show that the minimality properties of CFFD and NDFD also hold in the case of constraint automata, that is, CFFD and NDFD are the minimal equivalences preserving the above mentioned fragments of linear time temporal logic with respect to the composition operators of constraint automata.
  - Then, we introduce the compositional model checking method for component based systems whose coordination subsystem and the interfaces of all components are modeled by constraint automata.
  - We introduce an implementation of the above mentioned method of model minimization and show its usefulness in practice by summarizing the results of some case studies.

- In chapter 9, we summarize our ideas and results and also suggest some research directions that can be considered as future work based on this thesis.

1.5 Research History and Publications

As mentioned before, the goal of this thesis is to find a *formal verification* framework for component based systems in general and for their coordination subsystems in particular. We have chosen *model checking of temporal logic properties* as the verification method. From a historical point of view, our research can be divided into four periods or steps:

- In the first step, our focus was on working on constraint automata as the operational semantics of coordination systems and as the models of the interfaces of components. Thus, in this phase, our aim was to prepare a compositional and hierarchal environment for model checking of linear time properties of constraint automata models in

the presence of the state explosion problem. Our main ideas to do this were presented in [82, 77]. To tackle the state explosion problem, we investigated two different solution methods:

  - *Compositional minimization* of models using suitable equivalence relations. For this aim, in [79, 78] we introduced two failure-based equivalence relations CFFD and NDFD as new semantics for constraint automata and our proposal for using them to reduce the size of models for model checking. We also proved that CFFD and NDFD are congruences with respect to join and hiding composition operators of constraint automata [75, 74]. Moreover, we showed that the temporal logics preservation and minimality properties of CFFD and NDFD hold for the case of constraint automata [76]. Based on these results, we introduced a method for compositional model checking of component based systems, reduction algorithms, and their implementations and application to case studies [76, 69].
  - As another way to deal with the state explosion, we considered *abstraction* techniques. In this method, using some suitable mapping from a large state space into a smaller one that preserves desired sets of linear temporal properties, we try to do model checking over the smaller model. A suitable mapping can be obtained by reducing the number of state variables from the model whose state variables are more to another one with less variables. Suitable mappings preserve linear time fairness and liveness properties [69, 119, 118].

- In the second step, based on the shortcomings of constraint automata in modeling all aspects of the behavior of connectors and the complication of their timed data streams based semantics, we focused on presenting a new operational semantics for Reo and on their model checking. Our main motivation in this phase was to use the classical theory of Büchi automata and their model checking using translation of linear temporal logics into automata [71, 73, 38, 72]. This step consisted of three phases:

  - In the first phase, we focus on the shortcomings of constraint automata in modeling some fairness constraint over the behavior of connectors and also their complicated timed data streams based semantics [71, 72]. To solve these problems:
    * We introduced *records* as data structures for modeling the simultaneous executions of events: ports in the domain of the record are allowed to communicate simultaneously the data assigned to them, while ports not in the domain of the record are blocked so that no communication can happen. The behavior of a network of components is specified in terms of (infinite) sequences of records, which give the order of occurrences of the events.
    * Also, we introduced *Büchi automata of streams of records* (BAR) as the operational semantics of coordination systems that cover the synchronization of the I/O operations and several fairness constraints over the behaviors of channels.
    * We obtained a main result that every constraint automaton can be translated into an essentially equivalent Büchi automaton of records.

  - In the second phase, we focused on the shortcomings of constraint automata in modeling the context dependent behaviors of some connectors. We introduced *augmented Büchi automata of records* (ABAR) which extend our BAR model with the possibility of testing if some ports of the environment are ready to communicate or not. ABAR's are defined as acceptors of infinite guarded strings of records. We also introduced a composition operator for ABAR's and showed its correctness with respect to the intended semantics by means of several examples. Finally, we showed that our composition operator can be decomposed into two operators: record extension and ordinary automata product [73, 72].
  - In the third phase, we intended to do model checking of linear time properties of ABAR models [38]. To this aim:
    * We defined an action based linear time temporal logic for expressing properties of Reo connectors, called ρLTL.
    * We showed that ρLTL formulas can be synthesized into Büchi automata representing Reo connectors, thus leading to an automata based model checking algorithm.
    * By generalizing standard automata based model checking algorithms for linear time temporal logic, we gave both global (inductive) and on-the-fly algorithms for the model checking of formulas for Reo connectors.

- In the third step, we implemented a tool set for model checking of Reo specifications using all of the above mentioned results. Now, we have two implemented tools designed independently based on the results of the two above steps. The first one, called *ArQuVer* (Architecture Quality Verification tool), is a tool that is intended to prepare an environment for specification of software architectures using constraint automata and verification of their properties, especially nonfunctional and qualitative properties of software architectures. It contains components for minimization of constraint automata using bisimulation, trace, CFFD and NDFD equivalence relations. The other, is a tool for model checking of ρLTL formulas over ABAR models. It implements the models using BDD data structure and checks the formulas directly over the BDD's. As a future work, we will incorporate these tools into an integrated one. Some results of the first tool were reported in [76, 69, 74]. For the other one, the first paper reporting our results is currently under review.

- The fourth step which is our intended future work, is to extend our theoretical and practical results into real-time and probabilistic coordination systems, using timed and probabilistic extensions of our BAR and ABAR models. Recently, we have presented a set of results on timed Büchi automata of records and the model checking of the linear time properties [8].