A Framework for Embedded Real-time System Design *

Jin-Young Choi[1], Hee-Hwan Kwak[2], and Insup Lee[2]

[1] Department of Computer Science and Engineering, Korea Univerity
choi@formal.korea.ac.kr
[2] Department of Computer and Information Science, University of Pennsylvania
heekwak@saul.cis.upenn.edu, lee@cis.upenn.edu

Abstract. This paper describes a framework for parametric analysis of real-time systems based on process algebra. The Algebra of Communicating Shared Resources (ACSR) has been extended to ACSR with Value-passing (ACSR-VP) in order to model the systems that pass values between processes and change the priorities of events and timed actions dynamically. The analysis is performed by means of bisimulation or reachability analysis. The result of the analysis is predicate equations. A solution to them yields the values of the parameters that satisfy the design specification. We briefly describe the proposed framework in which this approach is fully automated and identify future work.

1 Introduction

There have been active research on formal methods for the specification and analysis of real-time systems [4, 5] to meet increasing demands on the correctness of embedded real-time systems. However, most of the work assumes that various real-time system attributes, such as execution time, release time, priorities, etc., are fixed *a priori*, and the goal is to determine whether a system with all these known attributes would meet required timing properties. That is to determine whether or not a given set of real-time tasks under a particular scheduling discipline can meet all of its timing constraints.

Recently, parametric approaches which do not require to guess the values of unknown parameters *a priori* have been proposed as general frameworks for the design analysis of real-time systems. Gupta and Pontelli [3] proposed a unified framework where timed automata has been used as a front-end, and the constraint logic programming (CLP) languages as a back-end. We [7] proposed a parametric approach based on real-time process algebra ACSR-VP (Algebra of Communicating Shared Resources with Value Passing). The scheduling problem is modeled as a set of ACSR-VP terms which contain the unknown variables as parameters. As shown in [7], a system is schedulable when it is bisimilar to a non-blocking process. Hence, to obtain the values for these parameters we

* This research was supported in part by NSF CCR-9619910, ARO DAAG55-98-1-0393, ARO DAAG55-98-1-0466, and ONR N00014-97-1-0505.

check a symbolic bisimulation relation between a system and a non-blocking process described both in ACSR-VP terms. The result of the bisimulation relation checking with the non-blocking process is a set of predicate equations of which solutions are the values for parameters that make the system schedulable. In this way, our approach reduces the analysis of scheduling problems into finding solutions of a recursive predicate equation system. We have demonstrated in [7] that CLP techniques can be used to solve predicate equations.

Before we explain an extension of our approach [7], we briefly present some background material below. Due to the space limitation we omit the formal definition of ACSR-VP. Instead, we illustrate the syntax and semantics using the following example process P.

$$P(t) = (t > 0) \rightarrow (a!t + 1, 1).P'(t)$$

The process P has a free variable t. The instantaneous action $(a!t+1, 1)$ outputs a value $t + 1$ on a channel a with a priority 1. The behavior of the process P is as follows. It checks the value of t. If t is greater than 0, then it performs the instantaneous action $(a!t + 1, 1)$ and becomes P' process. Otherwise it becomes NIL. For more information on ACSR-VP a reader refers to [7].

To capture the semantics of an ACSR-VP term, we proposed a Symbolic Graph with Assignment (SGA). SGA is a rooted directed graph where each node has an associated ACSR-VP term and each edge is labeled by boolean, action, assignment, (b, α, θ). Given an ACSR-VP term, an SGA can be generated using the rules shown in [7].

The notion of bisimulation is used to capture the semantics of schedulability of real-time systems. The scheduling problem is to determine if a real-time system with a particular scheduling discipline meets all of its deadlines and timing constraints. In ACSR-VP, if no deadline and constraints are missed along with any computation of the system, then the process that models the system always executes an infinite sequence of timed action. Thus by checking the bisimulation equivalence between the process that models the system and the process that idles infinitely, the analysis of schedulability for the real-time systems can be achieved.

2 A Fully Automatic Approach for the Analysis of Real-time Systems.

In the approach published in [7] a bisimulation relation plays a key role to find solutions for parameters. However, the disadvantage with a bisimulation relation checking method is that it requires to add new τ edges. These new edges will increase the size of a set of predicate equations and the complexity to solve them. To reduce the size of a set of predicate equations, we introduced a parametric reachability analysis techniques.

As noted in [7], finding conditions that make system schedulable is equivalent to finding symbolic bisimulation relation with an infinite idle process. Furthermore, checking the symbolic bisimulation relation with an infinite idle process

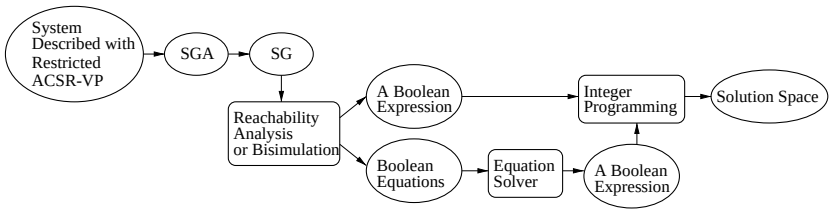

Fig. 1. Our Approach for the Real-time System Analysis

is equivalent to finding conditions that guarantee there is always a cycle in an SGA regardless of a path taken. That is, there is no deadlock in the system under analysis. Hence, we can obtain a condition that guarantees there is no deadlock in the system under analysis by checking possible cycles in an SGA for the system under analysis. We illustrate that this reachability analysis can replace a bisimulation relation checking procedure. With a reachability analysis we can avoid adding new τ edges and reduce the complexity of solving predicate equations.

Utilizing existing CLP techniques seems to be a natural way of solving predicate equations. However, it is not possible to determine if a CLP program terminates. This leads us to identify a decidable subset of ACSR-VP terms. This subset can be classified by defining variables in ACSR-VP terms into two types: control variable and data variable. Control variable is a variable with finite range. The value of a control variable can be modified while a process proceeds. Data variable is the variable that does not change its value. That is, it just hold values "passively" without modification to them. Data variables may assume values from infinite domains. A detailed explanation on a decidable subset of ACSR-VP is given in [6]. We use the term "restricted ACSR-VP" to denote a decidable subset of ACSR-VP.

With a restricted ACSR-VP terms we can reduce a real-time system analysis into solving either a boolean expression or boolean equations with free variables. A decidable subset of ACSR-VP allow us to generate a boolean expression or boolean equations with free variables (BESfv) as a result of reachability analysis or symbolic bisimulation checking. We have developed a BESfv solving algorithm, which is based on maximal fixpoint calculation.

Here we explain the overview of our fully automatic approach, which is a refined version of our previous one [7]. A simplified version of the overall structure of our approach is shown in Figure 1. We describe a system with restricted ACSR-VP terms. With a given set of restricted ACSR-VP processes, an SGA is generated from a restricted ACSR-VP term in order to capture the behavior of a system. Once an SGA is generated, we instantiate all the control variables in each SGA node to form an Symbolic Graph (SG). An SG is a directed graph in which every edge is labeled by (b, α), where b is a boolean expression and α is an action. As an analysis either the symbolic bisimilarity is checked on an SG with an SG of infinite idle process or reachability analysis can be directly

performed on an SG of the system. The result is a set of boolean equations or a boolean expression. In the case that a boolean expression with free variables is produced, it can be solved by existing integer programming tools such as Omega Calculator [8]. In the other case that boolean equations with free variables are generated, an algorithm presented in [6] can be applied.

We have applied our framework into several real-time scheduling problems. For real-time scheduling problems, the solution to boolean expression or a set of boolean equations with free variables identifies, if it exists, under what values of unknown parameters the system becomes schedulable. For instance, in the shortest job first scheduling, we may want to know the period of certain jobs that guarantee the scheduling of the system. We let those periods be unknown parameters and describe the system in ACSR-VP process terms. Those unknown parameters are embedded into the derived boolean expression or boolean equations, and consequently the solutions of them represent the values of unknown parameters that make them satisfiable. These solutions represent the valid ranges of periods (i.e., unknown parameters) of the jobs that make the system schedulable.

Our method is expressive to model complex real-time systems in general. Furthermore, the resulting boolean-formulas can be solved efficiently. For instance, there has been active research [2] to solve a boolean expression efficiently, and there are existing tools such as Omega Calculator [8] for a Presburger formulas. Another significant advantage of our method is the size of graphs. Due to the abstract nature of SGA, the size of an SGA constructed from an ACSR-VP term is significantly smaller than that of Labeled Transition Systems (LTS) which requires all the parameters to be known *a priori*. Consequently, this greatly reduces the state explosion problem, and thus, we can model larger systems and solve problems which were not possible by the previous approaches due to state explosions.

Furthermore, our approach is decidable whereas other general framework as [3] is not, and thus, it is possible to make our approach fully automatic when we generate a set of boolean equations or a boolean expression. Since our approach is fully automatic, it can also be used to check other properties as long as they can be verified by reachability analysis.

3 Conclusion

We have overviewed a formal framework for the specification and analysis of real-time systems design. Our framework is based on ACSR-VP, symbolic bisimulation, and reachability analysis. The major advantage of our approach is that the same framework can be used for scheduling problems with different assumptions and parameters. In other real-time system analysis techniques, new analysis algorithms need to be devised for problems with different assumptions since applicability of a particular algorithm is limited to specific system characteristics.

We believe that restricted ACSR-VP is expressive enough to model any real-time system. In particular, our method is appropriate to model many complex

real-time systems and can be used to solve the *priority assignment problem*, *execution synchronization problem*, and *schedulability analysis problem* [9]. We are currently investing how to adapt the proposed frame for embedded hybrid systems, that is, systems with both continous and discrete components.

The novel aspect of our approach is that schedulability of real-time systems can be described formally and analyzed automatically, all within a process-algebraic framework. It has often been noted that scheduling work is not adequately integrated with other aspects of real-time system development [1]. Our work is a step toward such an integration, which helps to meet our goal of making the timed process algebra ACSR a useful formalism for supporting the development of reliable real-time systems. Our approach allows the same specification to be subjected to the analysis of both schedulability and functional correctness.

There are several issues that we need to address to make our approach practical. The complexity of an algorithm to solve a set of boolean equations with free variables grows exponentially with respect to the number of free variables. We are currently augmenting PARAGON, the toolset for ACSR, to support the full syntax of ACSR-VP directly and implementing a symbolic bisimulation algorithm. This toolset will allow us to experimentally evaluate the effectiveness of our approach with a number of large scale real-time systems.

References

1. A. Burns. Preemptive priority-based scheduling: An appropriate engineering approach. In Sang H. Song, editor, *Advances in Real-Time Systems*, chapter 10, pages 225–248. Prentice Hall, 1995.
2. Uffe Engberg and Kim S. Larsen. Efficient Simplification of Bisimulation Formulas. In *Proceedings of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 111–132. LNCS 1019, Springer-Verlag, 1995.
3. G. Gupta and E. Pontelli. A constraint-based approach for specification and verification of real-time systems. In *Proceedings IEEE Real-Time Systems Symposium*, December 1997.
4. Constance Heitmeyer and Dino Mandrioli. *Formal Methods for Real-Time Computing*. Jonh Wiley and Sons, 1996.
5. Mathai Joseph. *Real-Time Systems: Specification, Verification and Analysis*. Prentice Hall Intl., 1996.
6. Hee Hwan Kwak. *Process Algebraic Approach to the Parametric Analysis of Real-time Scheduling Problems*. PhD thesis, University of Pennsylvania, 2000.
7. Hee-Hwan Kwak, Jin-Young Choi, Insup Lee, Anna Philippou, and Oleg Sokolsky. Symbolic Schedulability Analysis of Real-time Systems. In *Proceedings IEEE Real-Time Systems Symposium*, December 1998.
8. William Pugh. The Omega test: a fast and practical integer programming algorithm for dependence analysis. *Communications of the ACM*, 8:102–114, August 1992.
9. Jun Sun. *Fixed-priority End-to-end Scheduling in Distributed Real-time Systems*. PhD thesis, University of Illinois at Urbana-Champaign, 1997.