Static Analysis of Real-Time Component-based Systems Configurations *

Candida Attanasio, Flavio Corradini, and Paola Inverardi

Dipartimento di Matematica Pura ed Applicata
Università dell’Aquila
Via Vetoio, Loc. Coppito, L’Aquila, Italy
fax +390862433180
{flavio,inverard}@univaq.it

Abstract. Nowadays, more and more often, complex systems are built by assembling together different system components. This technology also affects the construction of heterogeneous and/or hybrid systems where components can represent hardware sensors, software controllers, etc. Moreover the resulting system is normally distributed. These systems have often real-time constraints/requirements and each component is characterized by its own speed determined by its local clock. In this paper we present a framework in which it is possible to specify and statically analyze the architecture of a system as a network of (parallel) components, each one with its own local clock. Then configuring the system means to formally define how to get the global clock out of the local clocks. This allows us, besides the usual behavioral and timing analysis, to, for example, verify if, and how changing the local speed of a component can affect the global performance of the system.

1 Introduction

Heterogeneous and hybrid systems are built by assembling together different system components like hardware sensors, software controllers, etc. Besides the traditional field of control systems [12, 2, 19], the need of assembling together heterogeneous components is more and more frequent due to the widespread diffusion of information technology in any application field, from multimedia applications to telecommunication systems. These systems have often real-time constraints/requirements and each component is characterized by its own speed determined by its local clock. This means that at each tick of its clock the component can perform an action. In our view, configuring a system out of such components means to decide two things, how the components are connected together in a given architecture and, with respect to time how to determine a global clock of the system in terms of the various local clocks. To a certain extent the latter consists in adjusting the local clocks to suitably relate each other. In

* This work has been partially funded by CNR, Progetto: Metodologie, e strumenti di analisi, verifica e validazione per sistemi software affidabili

physical terms this can be seen as deciding the starting time of each component. In the literature, much attention has been given to the the first configuration step, both at the architectural level and at the specification level [20, 15, 17, 4, 18], and to modeling time in various formalisms [7, 14, 21, 19, 9, 13]. In this paper we present a framework in which it is possible to specify the architecture of a system as a network of (parallel) components, each one with its own local clock. Thus, the configuration of the behavioral part is achieved by putting the components in parallel and let them communicate upon synchronization, similarly to [18]. The configuration of the clocks means to formally define the global clock out of the local clocks. This is done by modeling the clocks as higher order terms in a given signature, and by relating each other through a unification process [16], which allows a common clock to be defined. Then, if there exists a unifier, all the local clocks will be expressed as suitable linear functions of the global one. Due to the properties of the unification process, i.e. the existence of a unique most general unifier, this clock configuration step is optimal that is, it is the best way to relate the local clocks so that the maximum number of synchronizations in the system can happen. The ability of modeling the clock configuration step allows us, besides the usual behavioral and timing analysis, to statically analyze the systems with respect to different configurations. For example, we can verify if, and how, changing the local speed (i.e. the local clock) of a component can affect the global performance of the system. That is the amount of synchronizations in the system increases or decreases. We apply our analysis framework to two different classical examples, “The Mine Pump” and the “The Lip Synchronization Problem”. The paper is organized as follows: Section 2, introduces the language of components, and its operational semantics. The language is CCS-like [20]. It is worth noticing that our approach is highly independent from the language chosen to model the behavioral aspects of components. Our choice has been mainly motivated by the simplicity and by the diffusive familiarity this class of languages exhibit. Section 3, presents our first case study used to illustrate the approach. It is a well known and studied example in the literature of which several different specifications exist [19]. Section 4 presents the second case study, i.e. a formalization of the lip synchronization problem. Section 5 shows how we can analyze the system at configuration level, that is how we can compare the behavior of different configurations obtained by considering the same components but with different speed. Section 6 discusses related Works, and Section 7 presents conclusions and future works.

2 The Language and Its Transitional Semantics

The language we consider in this paper is deliberately simple. We consider Nets of Automata, a CCS-like language where parallelism can only appear at the top level. This language is close to the language presented in [18].

Similarly to [20], we assume a set of actions $A$ (ranged over by $\alpha$) from which we obtain the set of co-actions $\bar{A} = \{\bar{\alpha} | \alpha \in A\}$ useful to model process synchronizations. We use $\text{Act}$ (ranged over by $a, b, \ldots$) to denote $A \cup \bar{A}$, the set of