Behaviour, Interaction and Dynamics*

Roberto Bruni[1], Hernán Melgratti[2], and Ugo Montanari[1]

[1] Dipartimento di Informatica, Università di Pisa, Italy
[2] Departamento de Computación, FCEyN, Universidad de Buenos Aires - Conicet, Argentina

Abstract. The growth and diffusion of reconfigurable and adaptive systems motivate the foundational study of models of software connectors that can evolve dynamically, as opposed to the better understood notion of static connectors. In this paper we investigate the interplay of behaviour, interaction and dynamics in the context of the BIP component framework, here denoted BI(P), as we disregard priorities. We introduce two extensions of BIP: 1) *reconfigurable* BI(P) allows to reconfigure the set of admissible interactions, while preserving the set of interacting components; 2) *dynamic* BI(P) allows to spawn new components and interactions during execution. Our main technical results show that reconfigurable BI(P) is as expressive as BI(P), while dynamic BI(P) allows to deal with infinite state systems. Still, we show that reachability remains decidable for dynamic BI(P).

1 Introduction

Recent years have witnessed an increasing interest about a rigorous modelling of (different classes of) connectors. The term *connector*, as used here, has been coined within the area of component-based software architectures, to name entities that can regulate the interaction of a collection of components [15]. This has led to the development of different mathematical frameworks that are used to specify, design, analyse, compare, prototype and implement suitable connectors. Our previous efforts have been focused at unifying different frameworks, in particular, the BIP component framework [2], Petri nets with boundaries [16] and the algebras of connectors [7,1] based on the tile model [12]. In [8] we have shown that BIP without priorities, written BI(P) in the following, is equally expressive as nets with boundaries. Thanks to the correspondence results in [16,10], we can define an algebra of connectors as expressive as BI(P), where a few basic connectors can be composed in series and parallel to generate any BI(P) system.

All above approaches deal with systems that have static structures, i.e., systems in which the possible interactions among components are all defined at design time and remain unchanged during runtime. Nevertheless, when shifting to connectors for systems that adapt their behaviour to changing environments,

* Research supported by the EU Integrated Project 257414 ASCENS, the Italian MIUR Project CINA (PRIN 2010/11), ANPCyT Project BID-PICT-2008-00319, and EU FP7-project MEALS 295261.

S. Iida, J. Meseguer, and K. Ogata (Eds.): Futatsugi Festschrift, LNCS 8373, pp. 382–401, 2014.

the situation is less well-understood. For example, approaches based on mobile calculi (like the π-calculus [14]) are not suited, because there the notion of connector / component is lost. In fact, a general and uniform theory for dynamic connectors is still lacking. On the one hand, static structures of connectors can be studied and executed efficiently. On the other hand, systems that can traverse a large or infinite number of connector configurations are better dealt with concise computational models that are tailored to dynamic structures.

Some recent progress has been done in [6], where Dy-BIP is proposed. We remind that an ordinary BIP component is defined by a set of ports P and a finite automaton whose transitions carry subsets of P as labels. An ordinary BIP system consists of a finite number of components (fixing the "Behaviour") whose ports are disjoint, together with a set of admissible synchronisations between the transitions of components (fixing the "Interaction"). Neither the set of components, nor the set of interactions can change over time. In contrast to BIP, the set of interactions can change dynamically in Dy-BIP, but this is obtained by ad-hoc design choices. As a consequence, the definition of Dy-BIP systems can be error-prone or lead to incomplete specifications unless the complex methodology outlined in [6] is adopted.

In order to contribute to the development of a general theory for dynamic connectors, in this paper we study two other extensions of the BI(P) framework with different degrees of "dynamism" that allow enhanced conciseness, modularity and expressiveness.

As a first step, we focus on a *reconfigurable* version of BI(P), analogous to but simpler than Dy-BIP. A reconfigurable BI(P) system allows for the dynamic modification of interactions among components, i.e., the set of available interactions changes as a side-effect of an interaction between components. Our first result proves that any reconfigurable BI(P) system is equivalent to an ordinary BI(P) system. This result is achieved by introducing a "controller" component for each interaction that can be added or removed at run-time. Roughly, the controller keeps track of whether the managed interaction is currently available or not and forces the components willing to use that interaction to synchronise also with the controller. This mapping shows that the reconfiguration capabilities provided by reconfigurable BI(P) do not increase the expressive power of BI(P). In fact, reconfigurable BI(P) only provides a more compact representation of ordinary systems, while ordinary BI(P) representations may require an exponential blow up in the number of components (it requires one controller for each possible interaction, and the interactions are subsets of ports). The crux of the proof is the fact that the set of controller components can be defined statically. In fact, the interfaces of components in reconfigurable BI(P) are static, i.e., the set of available ports in every component is fixed. As a consequence, the set of all possible interactions in a system are determined at design time (despite the fact that they can be enabled/disabled at run-time).

Our next step is to explore situations in which the interfaces of the components may change dynamically (i.e., to support the creation/elimination of ports). This requirement also imposes the necessity of handling interactions that can be