Journal of Logical and Algebraic Methods in Programming 86 (2017) 134–156

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Combine and conquer: Relating BIP and Reo

Kasper Dokter [a,*], Sung-Shik Jongmans [a,b,c], Farhad Arbab [a], Simon Bliudze [d]

[a] *Centrum Wiskunde & Informatica, Science Park 123, 1098 XG, Amsterdam, Netherlands*
[b] *Open University of the Netherlands, Valkenburgerweg 177, 6419 AT, Heerlen, Netherlands*
[c] *Radboud University Nijmegen, Toernooiveld 212, 6525 EC, Nijmegen, Netherlands*
[d] *École Polytechnique Fédérale de Lausanne, Station 14, 1015, Lausanne, Switzerland*

A R T I C L E   I N F O

Article history:
Received 7 November 2015
Received in revised form 30 September 2016
Accepted 30 September 2016
Available online 6 October 2016

Keywords:
Coordination
Formal translations
Correctness proofs
BIP
Reo

A B S T R A C T

Coordination languages simplify design and development of concurrent systems. Particularly, exogenous coordination languages, like BIP and Reo, enable system designers to express the interactions among components in a system explicitly. A formal relation between exogenous coordination languages comprises the basis for a solid comparison and consolidation of their fundamental concepts. In this paper we establish a formal relation between BI(P) (i.e., BIP without the priority layer) and Reo, by defining transformations between their semantic models. We show that these transformations preserve all properties expressible in a common semantics. We use these transformations to define data-sensitive BIP architectures and their composition.

1. Introduction

The main challenge in concurrency consists of coordination of interacting processes. Poor coordination results in systems that can suffer from corruption of shared resources, deadlocks, and starvation. To avoid these issues, we need explicit full control over interactions. A language that supports concurrency provides constructs that allow processes to interact. Such constructs include synchronous and asynchronous message passing and shared memory. However, most concurrent languages do not provide constructs that also control interaction among processes. To stay in charge of interaction, system designers need to use constructs such as locks and semaphores. This blends the code that controls interaction with other code of the program, and complicates the analysis, optimization and reusability of the implemented coordination.

Exogenous coordination languages, like BIP [1,2] and Reo [3,4], address this coordination problem by separating coordination of interactions from computation in processes [5]. This enables designers to control interaction using language constructs, making coordination visible to tools like model checkers and compilers.

In BIP, a concurrent system consists of a superposition of three layers: behavior, interaction and priorities. The behavior layer contains the processes that need to be coordinated. The interaction layer explicitly specifies which interactions are possible, which gives full control over the interactions in the system. Mutually exclusive execution of these interactions ensures that overlapping interactions do not cause a conflict. If multiple interactions are possible, then the priority layer selects a preferred one.

* Corresponding author.
E-mail address: K.P.C.Dokter@cwi.nl (K. Dokter).

http://dx.doi.org/10.1016/j.jlamp.2016.09.008
2352-2208/© 2016 Elsevier Inc. All rights reserved.

In Reo, processes interact by means of a coordination protocol. A protocol consists of a graph-like structure, called a connector, that models the synchronization and dataflow among the processes. Reo connectors may compose together to form more complex connectors, allowing reusability and compositional construction of coordination protocols.

Although BIP and Reo address the same coordination problem, their underlying design principles and toolchains (containing tools for editing, code generation and model checking [6,7,4]) differ significantly. By combining their principles and tools, we would conquer new terrain in the field of concurrent languages. However, some principles (visible in the formal definitions of each language) may be conflicting, and prevent such a complete unification. A formal relation between BIP and Reo is necessary to identify these conflicts.

In this paper, we provide such a formal relation between BIP and Reo by relating their semantic models. We consider two kinds of semantic models for BIP and Reo: data-agnostic and data-sensitive. In the data-agnostic domain, we relate port automata as semantics of Reo and BIP architectures [8,9]. We show that connectors in BIP and Reo coincide modulo internal transitions and independent progress of transitions. In the data-sensitive domain, we relate stateless constraint automata as semantics of Reo to BIP interaction models [8,10]. The restriction to stateless constraint automata arises from the fact that BIP interaction models are stateless. We show that stateless constraint automata and BIP interaction models have the same observable behavior.

Stateful data-sensitive Reo connectors require stateful constraint automata for their semantics, which informally correspond to data-sensitive BIP architectures. A data-sensitive BIP architecture consists of a (data-sensitive) BIP interaction model together with a set of coordinating components. However, current literature on BIP does not provide definitions that allow composition of data-sensitive BIP architectures. Indeed, only hierarchical composition of interaction models is defined in [10], which is insufficient to define a full composition of data-sensitive BIP architectures.

We address this problem by using our formal translations to propose a composition operator for data-sensitive BIP architectures. In addition, we show that it is possible to relate (stateful) constraint automata and data-sensitive BIP architectures.

Although BIP's notion of priority is equally applicable to the constraint automata semantics of Reo, Reo provides no syntax to specify such global priority preferences.[1] Therefore, in this paper, "BIP" generally refers to "BI(P)", an name that others have already used to designate BIP without its priority layer.

The rest of this paper is organized as follows: In Section 2, we recall the semantic models of BI(P) and Reo. In Section 3, we relate port automata in Reo and BIP architectures. In Section 4, we relate BIP interaction models with stateless constraint automata in Reo. In Section 5, we propose an extension of data-agnostic BIP architectures to the data-sensitive domain, and show how this enables incremental translation from stateful constraint automata to data-sensitive BIP architectures. In Section 6, we discuss related work. In Section 7, we conclude and point out future work.

This paper extends a paper presented at ICE 2015 [13]. The main additional contribution of this extended version consists of the proposal of data-sensitive BIP architectures and their composition in Section 5. Furthermore, we added the proofs of Theorem 1 and Lemma 2, and revised the introduction, conclusion and related work.

2. Overview of BIP and Reo

2.1. BIP

A BIP system consists of a superposition of three layers: Behavior, Interaction, and Priority. The behavior layer encapsulates all computation, consisting of *atomic components* processing sequential code. *Ports* form the interface of a component through which it interacts with other components. BIP represents these atomic components as *Labelled Transition Systems* (LTS) having transitions labelled with ports and extended with data stored in local variables. The second layer defines component coordination by means of *BIP interaction models* [10]. For each *interaction* among components in a BIP system, the interaction model of that system specifies the set of ports synchronized by that interaction and the way data is retrieved, filtered and updated in each of the participating components. In the third layer, priorities impose scheduling constraints to resolve conflicts in case alternative interactions are possible.

In the rest of this paper, we disregard priorities and focus mainly on interaction models (cf. footnote 1).

Data-agnostic semantics. We first introduce a data-agnostic semantics for BIP.

Definition 1 *(BIP component [9]).* A *BIP component* C over a set of ports P_C is a labelled transition system $(Q, q^0, P_C, \rightarrow)$ over the alphabet 2^{P_C}. If $\mathcal{C}$ is a set of components, we say that $\mathcal{C}$ is *disconnected* iff $P_C \cap P_{C'} = \emptyset$ for all distinct $C, C' \in \mathcal{C}$. Furthermore, we define $P_{\mathcal{C}} = \bigcup_{C \in \mathcal{C}} P_C$.

Then, BIP defines an *interaction model* over a set of ports P to be a set of subsets of P. Interaction models are used to define synchronizations among components, which can be intuitively described as follows. Given a disconnected set of BIP components $\mathcal{C}$ and an interaction model γ over $P_{\mathcal{C}}$, the state space of the corresponding *composite component* $\gamma(\mathcal{C})$ is the

[1] Reo does have a weaker priority mechanism to specify local preferences, called *context-sensitivity*. A premier example in the Reo literature is the context-sensitive channel LossySync, which prefers locally maximal dataflow. Clarke et al. first studied context-sensitivity through a special context-sensitive semantic model for Reo [11]; later, Jongmans et al. showed how to encode context-sensitivity in non-context-sensitive models [12].