Science of Computer Programming 98 (2015) 293–316

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Optimized distributed implementation of multiparty interactions with Restriction ☆

Saddek Bensalem [a], Marius Bozga [a], Jean Quilbeuf [a,*], Joseph Sifakis [a,b]

[a] UJF-Grenoble 1/CNRS VERIMAG UMR 5104, Grenoble, F-38041, France
[b] RISD Laboratory, EPFL, Lausanne, CH-1015, Switzerland

ARTICLE INFO

Article history:
Received 3 March 2013
Received in revised form 30 January 2014
Accepted 5 February 2014
Available online 20 February 2014

Keywords:
Multiparty interaction
Priority
Observation
Conflict resolution
Distributed systems

ABSTRACT

Using high level coordination primitives allows enhanced expressiveness of component-based frameworks to cope with the inherent complexity of present-day systems designs. Nonetheless, their distributed implementation raises multiple issues, regarding both the correctness and the runtime performance of the final implementation. We propose a novel approach for distributed implementation of multiparty interactions subject to scheduling constraints expressed by priorities. We rely on a new composition operator named Restriction, whose semantics dynamically restricts the set of interactions allowed for execution, depending on the current state. We show that this operator provides a natural encoding for priorities. We provide a knowledge-based optimization that modifies the Restriction operator to avoid superfluous communication in the final implementation. We complete our framework through an enhanced conflict resolution protocol that natively implements Restriction. A prototype implementation allows us to compare performances of different optimizations.

1. Introduction

Correct design and implementation of computing systems has been an active research topic over the past three decades. This problem is significantly more challenging in the context of distributed systems due to a number of factors such as non-determinism, asynchronous communication, race conditions, fault occurrences, etc. Model-based development of such applications aims to ensure correctness through the usage of explicit model transformations from high-level models to code.

In this paper, we focus on distributed implementation for models defined using the BIP framework [6]. BIP (Behavior, Interaction, Priority) is based on a semantic model encompassing composition of heterogeneous components. The *behavior* of components is described as an automaton extended by data and associated functions written in C. BIP uses an expressive set of composition operators for obtaining composite components from a set of components. The operators are parameterized by a set of *multiparty interactions* between the composed components and by *priorities*, used to specify different scheduling mechanisms between interactions.[1]

☆ This article extends two papers, presented at the AGERE!2012 workshop and at the FMOODS/FORTE 2012 conference. The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007–2013] under grant agreement No. 248776 (PRO3D) and No. 257414 (ASCENS) and from ARTEMIS JU grant agreement ARTEMIS-2009-1-100230 (SMECY).

* Corresponding author.
E-mail addresses: bensalem@imag.fr (S. Bensalem), bozga@imag.fr (M. Bozga), quilbeuf@fortiss.org (J. Quilbeuf), sifakis@imag.fr (J. Sifakis).

[1] Although our focus is on BIP, all results in this paper can be applied to any model that is specified in terms of a set of components synchronized by interactions with priorities.

http://dx.doi.org/10.1016/j.scico.2014.02.013
0167-6423/

A multiparty interaction is a high-level construct that expresses a strong synchronization between a fixed set of components. Such an interaction takes place only if all its participant components agree to execute it. If two multiparty interactions involve a common component, they are conflicting because the common component cannot participate in both interactions. Transforming a BIP model into a distributed implementation consists in addressing three fundamental issues:

1. *Enabling concurrency*. Components and interactions should be able to run concurrently while respecting the semantics of the high-level model.
2. *Conflict resolution*. Interactions that share a common component can potentially conflict with each other. Such interactions should be executed in mutual exclusion.
3. *Enforcing priorities*. When two interactions are simultaneously enabled, only the one with higher priority can be chosen for execution. Priorities can be applied indifferently between conflicting or non-conflicting interactions.

We developed a general method based on source-to-source transformations of BIP models with multiparty interactions leading to distributed models that can be directly implemented [16,17]. This method has been later extended to handle priorities [18] and optimized by exploiting knowledge [12]. The target model consists of components representing processes and Send/Receive interactions representing asynchronous message passing. Correct coordination is achieved through additional components implementing conflict resolution and enforcing priorities between interactions.

In particular, the conflict resolution issue has been addressed by incorporating solutions to the *committee coordination problem* [20] for implementing multiparty interactions. Intuitively, this problem consists in scheduling several meetings, every one involving a set of professors. A meeting requires the whole attendance. A professor cannot participate in more than one meeting at a time. Bagrodia [3] proposes solutions to this problem with different degrees of parallelism. The most distributed solution is based on the drinking philosophers problem [19], and has inspired the later approaches of Pérez et al. [41] and Parrow and Sjödin [39]. In the context of BIP, a transformation addressing all the three challenges through employing a *centralized scheduler* is proposed in [5]. Moreover, in [16], the transformation is extended to address both the concurrency issue by breaking the atomicity of interactions and the conflict resolution issue by embedding a solution to the committee coordination problem in a distributed fashion.

Distributed implementation of priorities is usually considered as a separate issue, and solved using completely different approaches. However, such an implementation should simultaneously enforce the priority rules and the mutual exclusion of conflicting interactions. In [18], priorities are eliminated by adding explicit scheduler components. This transformation leads to potentially more complex models, having definitely more interactions and conflicts than the original one. In [7], situations where priorities and multiparty interactions are intermixed, called confusion, are avoided by adding more priorities.

Enforcing priority rules is done when deciding execution of low priority interactions, by checking that no interaction with more priority is ready to execute. This check requires a synchronous view of the components involved in higher priority interactions. The distributed knowledge [22] of a component consists of all the information that it can infer about other components state, based on its current state and the reachable states. In [15,8,4], the focus is on reducing the overhead for implementing priorities by exploiting knowledge. Yet, these approaches make the implicit assumption that multiparty interactions are executed atomically and do not consider conflict resolution.

In [13], we introduce a new composition operator called Restriction. This operator associates a state predicate to each multiparty interaction. The semantics of Restriction allows executing an interaction only if the associated predicate evaluates to true in the current state. For instance, an interaction “start” representing a car starting at a crossing might be restricted with the predicate “the traffic light is green”. Note that the predicate possibly depends on components that do not participate in the interaction. In our example, the traffic light component is not necessarily a participant in the “start” interaction.

This paper is an extension of both [12] and [13], and combines the two approaches. This combination yields several methods for obtaining a distributed implementation of multiparty interactions subject to priorities. These methods rely on an appropriate intermediate model and transformations towards fully distributed models. Each transformation, depicted by an arrow in Fig. 1, either refines the composition operator used to glue components of the model or optimizes the model. The contribution is manyfold:

1. First, we introduce an alternative semantics for BIP that relies on the Restriction operator. We show that this operator is general enough to encompass priorities through a simple transformation (Transformation 1 of Fig. 1). The Restriction operator reveals two types of conflicts occurring between interactions, that can be handled using different conflict resolution mechanisms (see below).
2. Second, we show that the knowledge-based optimization originally presented in [12] can be extended to handle the Restriction and reduce the overall coordination of the model. This optimization (Transformation 2 of Fig. 1) modifies only the predicates used by the Restriction operator.
3. Third, a model with Restriction can be used as an intermediate step in the transformations leading to a distributed implementation. We show that *observation conflicts*, that usually follow from encoding of priorities, can be dealt more efficiently than *structural conflicts*, where two multiparty interactions involve a common component. In particular, we compare two approaches for generating a distributed implementation. The first consists in encoding Restriction with