Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 343 (2005) 285–304
Theoretical Computer Science
www.elsevier.com/locate/tcs

Polarized process algebra with reactive composition

J.A. Bergstra[a,b], I. Bethke[a,*]

[a] *Faculty of Science, University of Amsterdam, Programming Research Group, The Netherlands*
[b] *Department of Philosophy, Utrecht University, Applied Logic Group, The Netherlands*

Abstract

Polarized processes are introduced to model the asymmetric interaction of systems. The asymmetry stems from the distinction between service and request. The scheduled concurrent composition of two polarized processes is called client–server composition or reactive composition, placing one process in the role of a client and the other process in the role of a server which is supposed to react on requests. The technical goal of this paper is to provide a definition of reactive composition for polarized processes and to prove that reactive composition thus defined is associative.

Keywords: Client–server composition; Polarized processes; Program algebra

1. Introduction

Program algebra is a tool for the conceptualization of programs and programming (see [5]). It is assumed that a program is executed in a context composed of components complementary to the program. While a program's actions constitute requests to be processed by an environment, the complementary system components in an environment are involved in processing such requests. The requests are understood as client actions whereas the complementary actions used to process requests are viewed as server actions. Server actions may be viewed as reactions on client actions. After each request the environment may undergo a state change, or a sequence of state changes, whereupon it replies with a boolean value. Returning the reply is itself a server action. The boolean return value is used 'by the program' to decide how the execution of the program will continue.

* Corresponding author.
E-mail addresses: janb@science.uva.nl (J.A. Bergstra), inge@science.uva.nl (I. Bethke).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.06.014

At the semantic level requests are modeled as actions, and programs as well as other deterministic system components are modeled as processes. System steps used for processing requests are modeled as process actions as well. The distinction between clients and servers will be made explicit by using request actions (for clients) and service actions (for servers) which have a different syntax. In the polarized composition of systems request actions interact with service actions adequate for serving the request.

A process will consist of three types of actions: request actions, service actions and internal actions. Due to the difference in status of these actions, the actions and processes involved are termed 'polarized' and actions have a polarity (request, service or neutral). Processes, moreover, have polarized roles in process composition. One process uses another one, where use involves getting requests processed via the service interface of the used process.

From the point of view of object orientation this work is of interest because it provides a new semantic setting—polarized processes with request and service actions—that integrates the behavior of an active object, e.g., a running program, with that of a passive object, e.g., a data structure. From the point of view of software components a possible perspective of this work is as follows: the program notation PGA—as briefly outlined below—can be considered as a notation for software components with process algebra as a format for its semantics. With these viewpoints reactive composition emerges as a new composition of software components.

The paper is organized as follows. In Section 2, we briefly recall the essentials of the basic polarized process algebra (BPPA) and the program algebra PGA. Section 3 introduces the basic internal action t and the abstraction operator τ. In Section 4, we accommodate BPPA with a new kind of composition yielding the thread algebra (TA) for reactive composition and give a couple of example components providing the service of simple data structures such as natural number and integer counters. In Section 5, we give an axiomatization of reactive composition and prove the main result of this paper, namely that reactive composition thus defined is associative. Section 6 applies this result and shows how to emulate a component that provides the service of an integer number counter by combining a component that counts natural numbers with a component that behaves as a one-bit cell.

2. Basic polarized process algebra

Most process algebras (e.g. CCS from [10], ACP from [3] and TCSP from [8]) are non-polarized. This means that in a parallel composition of process P and Q, both processes and their actions have a symmetric status. In a polarized setting each action has a definite asymmetric status. Either it is a request or it is (part of) the processing of a request. When a request action is processed a boolean value is returned to the process issuing the request. When this boolean value is returned the processing of the request has completed.

Non-polarized process algebra may be considered the case in which always `true` is returned. Polarized process algebra is significantly less elegant than non-polarized process algebra. Its advantage concerns the more direct modeling of sequential deterministic systems. The principal example of polarized composition concerns the composition of the behavior of a program with the behavior of a processor on which that program is running.