Available online at www.sciencedirect.com

ScienceDirect

Computer Languages, Systems & Structures 33 (2007) 111 – 133

COMPUTER LANGUAGES, SYSTEMS & STRUCTURES

www.elsevier.com/locate/cl

A bulk-synchronous parallel process algebra

Armelle Merlin*, Gaétan Hains

LIFO, Université d'Orléans-CNRS, France

Received 29 November 2005; accepted 16 November 2006

Abstract

The calculus of communicating systems (CCS) process algebra is a well-known formal model of synchronization and communication. It is used for the analysis of safety and liveness in protocols or distributed programs. In more recent work, it is used for the analysis of security properties. Bulk-synchronous parallelism (BSP) is an algorithm and programming model of data-parallel computation. It is useful for the design, analysis and programming of scalable parallel algorithms.

Many current evolutions require the integration of distributed and parallel programming: grid systems for sharing resources across the Internet, secure and reliable global access to parallel computer systems, geographic distribution of confidential data on randomly accessible systems, etc. Such software services must provide guarantees of safety, liveness, security together with scalable and reliable performance. Formal models are therefore needed to combine parallel performance and concurrent behavior. With this goal in mind, we propose here an integration of BSP with CCS semantics, generalize its cost (performance) model and sketch its application to scheduling problems in meta-computing.

Keywords: Performance model; BSP model; Parallel programming; Process algebras; Path algebras

1. Introduction

Programming of massively parallel architectures is difficult and error prone. To ease this kind of programming several approaches are possible. One of those, BS-λ [1] associates the bulk-synchronous parallel (BSP) execution and cost model [2–4], which provides portable and predictable performance on a wide variety of architectures, with functional programming. We propose to apply more expressive non-deterministic models such as process algebras to meta-computing (to several data-parallel calculations on a given machine or several parallel machines (GRID [5,6])), keeping the portability and predictability of BSP.

To associate process algebra to parallel programming, we use explicit processors through a constructor $\langle \ldots \rangle$. This permits to distinguish interleaving (due to concurrency) and synchronization (due to parallelism).

Calculus of communicating systems (CCS) [7–9] is a well-known and simple process algebra which we extend to write BSP barriers as atomic events (avoiding thus one source of combinatory explosion e.g. replacing certain sets of interleavings by barrier synchronizations (Fig. 2). Both theories combine in our BSPA "bulk-synchronous process algebra". We build on this new algebra a path algebra [10] via a specific "vectorial" semi-ring to compute BSP costs.

Fig. 1 illustrates the notions of interleaving, synchronism and costs in BSPA. Notations will be defined in Section 2.

* Corresponding author. School of Computer Science, University of St. Andrews, St. Andrews, Fife KY16 8HZ, UK. Tel.: +44 1334463260.
E-mail addresses: merlin@lifo.univ-orleans.fr, ab@dcs.st-and.ac.uk (A. Merlin), ghains@lifo.univ-orleans.fr (G. Hains).

1477-8424/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cl.2006.11.001

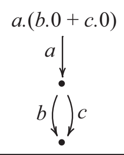
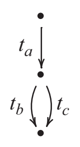
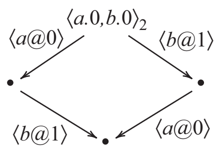
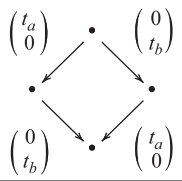
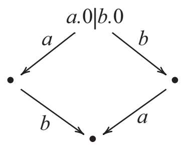
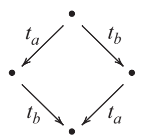
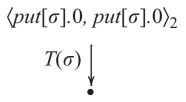
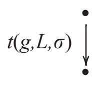
	Processus paths	Graphs's cost	Cost
Choice			$t_a + \max(t_b, t_c)$
Asynchronism	 $\langle a.0, b.0\rangle_2$ $\langle a@0\rangle$ $\langle b@1\rangle$ $\langle b@1\rangle$ $\langle a@0\rangle$		$\begin{pmatrix} t_a \\ t_b \end{pmatrix} = \max(t_a, t_b)$
Interleaving			$\max(t_a + t_b, t_b + t_a) = t_a + t_b$
Barrier Synchronization			$t(g,L,\sigma)$

Fig. 1. Interleaving, synchronism and costs in BSPA. Notations will be defined in Section 2.

We will first present here preliminaries on BSP and CCS. In the first section we will propose a CCS extension to express BSP-like processes. The second section will present the formal usual tools for a process algebra. We then describe a formal cost model for this extended algebra in the standard graph theoretic manner: to associate "customized" semi-ring elements to sets of paths in the transition systems of CCS processes. We finally outline its application to several scheduling problems in meta-computing, and some future developments.

1.1. Related work

Krishnan's distributed CCS model [11] and Rebeuf's asynchronous performance model [12] allow arbitrarily complex communications without taking advantage of barriers which minimize combinatorial explosion either in process algebraic calculations or in cost calculations. We have therefore favored the BSP point of view on parallel computation. In [13], the authors apply a process algebra with explicit processors for parallel programs (pipeline skeleton) with costs but do not give an explicit communication cost calculus.

Prasad [14] has studied synchronous variants of CCS with their axiomatizations. This work is not directly applicable to ours in its current form because we rely on asynchronous parallel CCS composition. But our BSP parallel composition, with its barrier, could possibly be axiomatized along the same lines as the synchronous CCS. This would involve a deeper study of our data-parallel constructor ($\langle \ldots \rangle$) and could lead to simplifications of BSPA which are not currently obvious.

Various works have studied timed and probabilistic process algebras, for example, [15–17], mostly from the point of view of process equivalence and axiomatization. The novelty of our approach is to concentrate on path composition whose associativity allows the application of efficient algorithms (matrix powers, transitive closures, etc.) when