Finite Equational Bases in Process Algebra: Results and Open Questions

Luca Aceto[1,4], Wan Fokkink[2,5], Anna Ingolfsdottir[1,4], and Bas Luttik[2,3]

[1] **BRICS** (**B**asic **R**esearch in **C**omputer **S**cience), Centre of the Danish National Research Foundation, Department of Computer Science, Aalborg University, Fr. Bajersvej 7B, 9220 Aalborg Ø, Denmark
{luca, annai}@cs.aau.dk
[2] CWI, Department of Software Engineering, PO Box 94079, 1090 GB Amsterdam, The Netherlands
[3] Department of Mathematics and Computer Science, Eindhoven Technical University, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
luttik@win.tue.nl
[4] Department of Computer Science, Reykjavík University, Ofanleiti 2, 103 Reykjavík, Iceland
{luca, annai}@ru.is
[5] Vrije Universiteit Amsterdam, Department of Computer Science, Section Theoretical Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
wanf@cs.vu.nl

Abstract. Van Glabbeek (1990) presented the linear time/branching time spectrum of behavioral equivalences for finitely branching, concrete, sequential processes. He studied these semantics in the setting of the basic process algebra BCCSP, and tried to give finite complete axiomatizations for them. Obtaining such axiomatizations in concurrency theory often turns out to be difficult, even in the setting of simple languages like BCCSP. This has raised a host of open questions that have been the subject of intensive research in recent years. Most of these questions have been settled over BCCSP, either positively by giving a finite complete axiomatization, or negatively by proving that such an axiomatization does not exist. Still some open questions remain. This paper reports on these results, and on the state-of-the-art in axiomatizations for richer process algebras with constructs like sequential and parallel composition.

1 Introduction

One of Jan Willem Klop's main contributions to the theory of concurrency is the development of the ACP family of process algebras in collaboration with Jan Bergstra—see the original papers [8, 9, 10, 11, 12], the textbooks [6, 18] and the historical paper [5]. Process algebras in the ACP style are defined, following the tradition of the algebraic specification of abstract data types, relying on tools from universal algebra and equational logic. More specifically, languages in the

A. Middeldorp et al. (Eds.): Processes... (Klop Festschrift), LNCS 3838, pp. 338–367, 2005.

ACP family are defined by specifying their signature—that is, the collection of algebraic operations that can be used to build new descriptions of reactive systems in terms of ones that we have already constructed—together with a collection of equational axioms that implicitly define the expected semantic properties of processes. This is an application of the classic axiomatic method, on which the development of modern algebra rests, to concurrency theory.

An example of a typical axiom that holds for all of the classic algebras in the ACP family, and is familiar from the theory of regular languages [16, 33], is

$$(x + y) \cdot z \approx (x \cdot z) + (y \cdot z) \ .$$

In the above equation, the operation symbols $+$ and $\cdot$ stand for "alternative composition" (or nondeterministic choice) and "sequencing", respectively. Intuitively, this axiom states that a process that can initially choose to behave either like x or like y, and then proceeds to behave like z, is "equivalent" to one that initially chooses to behave either like $x \cdot z$ or like $y \cdot z$.

On the other hand, the right-distributivity axiom of alternative composition over sequencing familiar from formal language theory, namely

$$x \cdot (y + z) \approx (x \cdot y) + (x \cdot z) \ ,$$

is usually *not* considered part of the axiom systems for process algebras since the left- and right-hand sides of the above equation may exhibit different deadlock potential, and should not be equated as descriptions of reactive systems.

Axiom systems arise from the desire of isolating the features that are common to a collection of algebraic structures—namely, their *models*. Early examples of models of the axiom systems for ACP style process algebras were the "projective limit" model—as employed in, e.g., [8]—, and the "graph model" adopted in [11].

Given a language in the ACP family, one may define intuitively appealing models of its axiom system as quotients of the collection of labelled transition systems modulo some behavioural congruence. *Labelled transition systems* (LTSs) [32] are a fundamental formalism for the description of concurrent computation, which is widely used in light of its flexibility and applicability. In particular, they underlie Plotkin's Structural Operational Semantics [41, 42] and, following Milner's pioneering work on CCS [36], are by now the standard formalism for describing the semantics of various process description languages.

LTSs model processes by explicitly describing their states and their transitions from state to state, together with the actions that produced them. Since this view of process behaviours is very detailed, several notions of behavioural equivalence and preorder have been proposed for LTSs. The aim of such behavioural semantics is to identify those (states of) LTSs that afford the same "observations", in some appropriate technical sense. The lack of consensus on what constitutes an appropriate notion of observable behaviour for reactive systems has led to a large number of proposals for behavioural equivalences for concurrent processes. (See the study [24], where van Glabbeek presents the linear time/branching time spectrum—a lattice of known behavioural equivalences and preorders over LTSs, ordered by inclusion.)