BRICS RS-05-19 Aceto et al.: Finite Equational Bases in Process Algebra: Results and Open Questions

BRICS

Basic Research in Computer Science

Finite Equational Bases in Process Algebra: Results and Open Questions

Luca Aceto
Willem Jan Fokkink
Anna Ingólfsdóttir
Bas Luttik

BRICS Report Series **RS-05-19**

ISSN 0909-0878 **June 2005**

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

http://www.brics.dk
ftp://ftp.brics.dk
This document in subdirectory RS/05/19/

Finite Equational Bases in Process Algebra: Results and Open Questions

Luca Aceto[1,5], Wan Fokkink[2,6], Anna Ingolfsdottir[1,3], and Bas Luttik[2,4]

[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@cs.aau.dk,annai@cs.aau.dk

[2] CWI, Department of Software Engineering, PO Box 94079, 1090 GB Amsterdam, The Netherlands

[3] Department of Computer Science, University of Iceland, 107 ReykjavÌk, Iceland, annaing@hi.is

[4] Department of Mathematics and Computer Science, Eindhoven Technical University, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, luttik@win.tue.nl

[5] School of Computer Science, ReykjavÌk University, Ofanleiti 2, 103 ReykjavÌk, Iceland, luca@ru.is

[6] 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 and ω-complete axiomatizations for them. (An axiomatization E is ω-complete when an equation can be derived from E if, and only if, all its closed instantiations can be derived from E.) 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 or ω-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 on axiomatizations for richer process algebras, containing 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 [8–12], 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 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, 34], 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) [33] 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 [42, 43] and, following Milner's pioneering work on CCS [37], 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.)

Having defined a model of an axiom system for a process algebra in terms of LTSs, it is natural to study the connection between the equations that are valid in the chosen model, and those that are derivable from the axioms using the rules of equational logic. The key questions here are:

- Is the axiom system complete? That is, can all of the equations that hold in the LTS model modulo the chosen notion of behavioural equivalence be derived from the axiom system using the rules of equational logic? (A complete axiom system is also referred to as a *basis* for the algebra it axiomatizes.) Researchers in concurrency theory often restrict themselves to studying axiom systems that are complete with respect to the collection of valid equations that do not contain occurrences of variables.
- Does the algebra of LTSs modulo the chosen notion of behavioural equivalence afford a finite equational axiomatization?

A complete axiomatization of a behavioural congruence yields a purely syntactic characterization, independent of LTSs and of the actual details of the definition of the chosen behavioural equivalence, of the semantics of the process algebra. This bridge between syntax and semantics plays an important role in both the practice and the theory of process algebras. From the point of view of practice, these proof systems can be used to perform system verifications in a purely syntactic way using general purpose theorem provers or proof checkers, and form the basis of purpose built axiomatic verification tools like, e.g., PAM [35]. A positive answer to the first basic question raised above is therefore not just theoretically pleasing, but has potential practical applications. From the theoretical point of view, complete axiomatizations of behavioural equivalences capture the essence of different notions of semantics for processes in terms of a basic collection of identities, and this often allows one to compare semantics which may have been defined in very different styles and frameworks. A review of existing complete equational axiomatizations for many of the behavioural semantics in van Glabbeek's spectrum is offered in [24]. The equational axiomatizations offered *ibidem* are over the language BCCSP, a common fragment of Milner's CCS [37] and Hoare's CSP [32] suitable for describing finite synchronization trees, and characterize the differences between behavioural semantics in terms of a few revealing axioms.

Negative answers to the second basic question mentioned above have instead served as a motivation for the development of auxiliary operations, whose addition to the original signature of the language under investigation yields an algebra with a finite basis. A classic example of this line of research, which can again be traced back to Jan Willem Klop's work in concurrency theory, is offered by the paper [10]. There Bergstra and Klop showed how to give a finite axiomatization of the language ACP using the auxiliary left and communication merge operators to characterize parallel composition. As shown by Moller [39, 40], auxiliary operators are needed to obtain a finite basis for that language because the process algebras CCS and ACP without the auxiliary left merge operator from [8] do not have a finite equational axiomatization modulo bisimulation equivalence.

An axiom system E is ω-*complete* when an equation can be derived from E if, and only if, all of its closed instantiations can be derived from E. In theorem proving applications, it is often convenient to work with axiomatizations that are ω-complete. In fact, using an ω-complete axiomatization one can avoid proofs by (structural) induction in favour of purely equational reasoning. Moreover, as argued by Heering in [27], ω-completeness of an axiom system is desirable in the partial evaluation of programs.

A classic example of an axiom system that is *not* ω-complete is that for the lambda-calculus—see [41].

Many of the existing axiomatizations of behavioural equivalences over expressive process description languages studied in concurrency theory are powerful enough to prove all of the valid equalities between terms that contain no occurrences of variables, but are *not* ω-complete. In fact, obtaining ω-complete axiomatizations in concurrency theory often turns out to be a difficult question, even in the setting of simple languages like BCCSP. This has raised a host of open questions that have been the subject of intensive investigation by process algebraists in recent years. Most of these questions have been settled over BCCSP and other simple process algebras, either positively by giving a finite ω-complete axiomatization, or negatively by proving that such an axiomatization does not exist. Still some open questions remain—especially for expressive process description languages and behavioural equivalences that, like observation equivalence [30, 37] and branching bisimilarity [25], abstract, in some formal sense, from events in process behaviours that are deemed to be directly unobservable.

In this paper, we report on positive and negative results pertaining to the existence of (finite) ω-complete axiomatizations for BCCSP and richer process algebras, containing constructs like sequential composition and interleaving. We hope that this survey of results will contribute to their dissemination in our research community, and will stimulate further investigations leading to the solution of the challenging open problems that are left.

The paper is organized as follows. We begin by presenting in Section 2 some basic background on universal algebra and equational logic that will be useful for the remainder of this study. In this general setting, we describe a collection of proof techniques that can be used to establish positive and negative results pertaining to the existence of finite, complete or ω-complete axiomatizations for algebras of processes. Section 3 reports on results and open problems on axiomatizations of behavioural equivalences over the language BCCSP studied by van Glabbeek in [24]. The paper concludes with a survey of the state-of-the-art on the equational theory of extensions of that language with more complex operators such as parallel composition and sequential composition (Sections 4 and 5).

2 General Techniques

Our aim in this section is to present some general techniques that can be used to establish results pertaining to the existence or non-existence of finite equational axiomatizations for behavioural equivalences and preorders over process description languages. A suitable general framework within which these techniques can be described is given by the classic fields of *universal algebra* and *equational logic*. We therefore begin by introducing the basic notions from these areas of mathematical research that will be used throughout this paper. We state at the outset that we shall not need very deep results or constructions from universal algebra in what follows, and that much more on it may be found in, e.g., the classic reference [15]. A self-contained presentation from a computer science perspective of the topics we now proceed to introduce may be found in [28].

2.1 Preliminaries

Σ-Algebras We start from a countably infinite set V of *variables* with typical elements x, y, w, z. A *signature* Σ consists of a set of *operation symbols*, disjoint from V, together with a function *arity* that assigns a natural number to each operation symbol. The set of *terms* over Σ is the least set such that

- Each $x \in V$ is a term.
- If f is an operation symbol of arity n, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is also a term.

An operation symbol f of arity 0 will be often called a *constant* symbol, and the term $f()$ will be abbreviated as f.

We write $\mathbb{T}(\Sigma)$ for the set of all terms over Σ and use t, u, v, possibly subscripted and/or superscripted, to range over terms. A term is *closed* (or *ground*) if it contains no occurrences of variables. We denote by $\mathrm{T}(\Sigma)$ the set of *closed* terms over Σ. A substitution is a mapping from variables to terms. A substitution is closed if it maps variables to closed terms. For every term t and substitution σ, the term obtained by replacing every occurrence of a variable x in t with the term $\sigma(x)$ will be written $\sigma(t)$. Note that $\sigma(t)$ is closed, if so is σ. Throughout this paper, we use the symbol "$=$" to stand for (syntactic) equality.

Example 1. A signature for the natural numbers with the operation $\max$ yielding the maximum between two numbers could contain a constant 0, a unary successor operation S and the binary operation symbol $\vee$. We shall use this signature as our running example throughout this section, and use $\vee$ in its customary infix notation for the sake of clarity.

Example 2. A process algebra that will be discussed extensively in Section 3 is BCCSP. Its signature consists of the constant $\mathbf{0}$, the binary operator $\_+\_$ called *alternative composition*, and unary *prefix* operators $a\_$, where a ranges over a nonempty set A of actions.

The collection of terms over a signature Σ yields a language. The semantics of this language can be defined canonically once we equip the set of intended denotations with the structure of a Σ-algebra. A *Σ-algebra* is a structure

$$\mathcal{A} = (\mathbf{A}, \left\{f^{\mathcal{A}} \mid f \in \Sigma\right\}) \ ,$$

where $\mathbf{A}$ is a non-empty set (often called the *carrier* of the algebra), and

$$f^{\mathcal{A}} : \mathbf{A}^n \to \mathbf{A}$$

for each operation symbol $f \in \Sigma$ of arity n. Note that if f is a constant symbol, then $f^{\mathcal{A}}$ can be viewed as an element of $\mathbf{A}$.

In order to interpret terms in $\mathbb{T}(\Sigma)$ in a *Σ-algebra* $\mathcal{A} = (\mathbf{A}, \left\{f^{\mathcal{A}} \mid f \in \Sigma\right\})$ we need the notion of an environment. An *environment* is a function ρ mapping variables to elements of $\mathbf{A}$. The mapping ρ can be extended homomorphically to $\mathbb{T}(\Sigma)$ in a unique way by stipulating that

$$\rho(f(t_1, \ldots, t_n)) = f^{\mathcal{A}}(\rho(t_1), \ldots, \rho(t_n))$$

for each operation symbol f of arity n and terms $t_1, \ldots, t_n$. Note that $\rho(t)$ is independent of ρ whenever t is closed. For each closed term t, we write $t^{\mathcal{A}}$ for the element of $\mathcal{A}$ that is the interpretation of t in the algebra $\mathcal{A}$. An element of the carrier set of $\mathcal{A}$ is *denotable* if it is the interpretation of some closed term.

Example 3. A suitable algebra $\mathcal{N}$ in which to interpret the collection of terms over the signature introduced in Example 1 has the set of natural numbers $\mathbb{N}$ as carrier set. The constant symbol 0 is interpreted as the natural number 0, the unary function symbol S is interpreted as the successor function—that is, the function mapping each natural number n to $n+1$—and the binary function symbol $\vee$ is interpreted as the function mapping each pair of natural numbers to the largest of the two.

It is easy to see that each element of $\mathcal{N}$ is denotable. Indeed, the natural number n is the interpretation of the term t_n defined thus:

$$\begin{aligned} t_0 &= 0 \quad \text{and} \\ t_{n+1} &= S(t_n) \ . \end{aligned}$$

The interpretation of the language $\mathbb{T}(\Sigma)$ in a Σ-algebra $\mathcal{A} = (\mathbf{A}, \{f^{\mathcal{A}} \mid f \in \Sigma\})$ naturally induces a congruence relation $=_{\mathcal{A}}$ over $\mathbb{T}(\Sigma)$. This is defined thus:

$$t =_{\mathcal{A}} u \ \text{ if, and only if, } \ \rho(t) = \rho(u), \ \text{ for each environment } \rho \ .$$

Example 4. Examples of identities that hold with respect to the congruence relation $=_{\mathcal{N}}$ induced by the interpretation of the language of terms over the signature for the natural numbers in our running example are

$$\begin{aligned} x \vee 0 &=_{\mathcal{N}} x \\ 0 \vee x &=_{\mathcal{N}} x \quad \text{and} \\ S(x) \vee S(y) &=_{\mathcal{N}} S(x \vee y) \ . \end{aligned}$$

The results reviewed in this paper all aim at using the classic logic of equality to offer a syntactic characterization of the relation $=_{\mathcal{A}}$ for algebras of processes. The study of such axiomatic characterizations of semantic equivalences falls therefore within the realm of equational logic, whose basics we now proceed to present.

Equational Logic An *axiom system* is a collection E of equations $t \approx u$ over the language $\mathbb{T}(\Sigma)$. (The equations in E are often referred to as *axioms*.) An equation $t \approx u$ is derivable from an axiom system E, notation $E \vdash t \approx u$, if it can be proven from the axioms in E using the rules of equational logic (viz. reflexivity, symmetry, transitivity, substitution and closure under Σ-contexts):

$$t \approx t \quad \frac{t \approx u}{u \approx t} \quad \frac{t \approx u \ \ u \approx v}{t \approx v} \quad \frac{t \approx u}{\sigma(t) \approx \sigma(u)}$$

$$\frac{t_i \approx u_i \ \ (1 \leq i \leq n)}{f(t_1, \ldots, t_n) \approx f(u_1, \ldots, u_n)} \ .$$

(The first three rules above state that $\approx$ is an equivalence relation, whereas the latter two state that $\approx$ is closed under substitutions, and is a congruence.) Formally, a proof of an equation $t \approx u$ from E is a sequence $t_i \approx u_i$ $(1 \leq i \leq n)$ of equations such that

- $t_n = t$ and $u_n = u$, and
- for each $1 \leq i \leq n$, the equation $t_i \approx u_i$ is obtained by applying one of the aforementioned inference rules using equations in E or some of the equations that precede it in the sequence as premises.

Without loss of generality one may assume that the substitution rule is only applied to axioms, i.e., that the rule

$$\frac{t \approx u}{\sigma(t) \approx \sigma(u)}$$

may only be used when $(t \approx u) \in E$. In this case, the equation $\sigma(t) \approx \sigma(u)$ is called a *substitution instance* of an axiom in E.

Moreover, by postulating that for each axiom in E also its symmetric counterpart is present in E, one may assume that there are no applications of the symmetry rule in equational proofs.

It is well-known (see, e.g., Sect. 2 in [26]) that if an equation relating two closed terms can be proven from an axiom system E, then there is a closed proof for it.

Definition 1 (Soundness). *Let $\mathcal{A}$ be a Σ-algebra. An equation $t \approx u$ is sound with respect to $=_{\mathcal{A}}$ iff $t =_{\mathcal{A}} u$. An axiom system is sound with respect to $=_{\mathcal{A}}$ iff so is each of its equations.*

In other words, an axiom system is sound with respect to $=_{\mathcal{A}}$ if it can only be used to prove equations that are valid in the algebra $\mathcal{A}$. This is, of course, a most natural requirement on an axiom system. However, ideally an axiom system should also allow us to prove all of the equations that hold in a given algebra. This is captured by the technical requirement of *completeness*.

Definition 2 (Completeness). *Let $\mathcal{A}$ be a Σ-algebra. An axiom system E is ground complete with respect to $=_{\mathcal{A}}$ iff $E \vdash t \approx u$ whenever $t =_{\mathcal{A}} u$, for all closed terms t, u.*

E is complete with respect to $=_{\mathcal{A}}$ iff $E \vdash t \approx u$ whenever $t =_{\mathcal{A}} u$, for all terms t, u.

Definition 3 (Equational Bases and Finitely Based Algebras). *An equational basis for an algebra $\mathcal{A}$ is a sound axiom system E that is complete with respect to $=_{\mathcal{A}}$. We say that an algebra $\mathcal{A}$ is finitely based if it has a finite equational basis.*

The notion of completeness of an axiom system relates the proof-theoretic notion of derivability using the rules of equational logic with the model-theoretic one of "validity in a model". From a proof-theoretic perspective, a useful property of an axiom system E is that, for all terms $t, u \in \mathbb{T}(\Sigma)$,

$$E \vdash t \approx u \text{ iff } E \vdash \sigma(t) \approx \sigma(u), \text{ for each closed substitution } \sigma \ . \tag{1}$$

An axiom system with the above property is called *ω-complete*. In theorem proving applications, it is convenient if an axiomatization is ω-complete, because this means that proofs by (structural) induction can be avoided in favour of purely equational reasoning. In fact, suppose that $\sigma(t) \approx \sigma(u)$ is provable from an axiom system E, for each closed substitution σ. If E is ω-complete, then we know that an equational proof of the actual equation $t \approx u$ from E exists. In general, the equation $t \approx u$ might not be derivable from E if E is just ground complete. In that case, we might have to content ourselves with showing that all closed instantiations of that equation are derivable from E, and this is usually done by induction on the structure of the closed terms that can be substituted for the variables occurring in t and u.

Example 5. The collection of equations corresponding to the congruences listed in Example 4 is easily seen to be ground complete with respect to $=_{\mathcal{N}}$. That axiom system is, however, neither complete nor ω-complete. For example, the equation

$$x \vee x \approx x \tag{2}$$

is valid in the algebra $\mathcal{N}$, and all of its closed instantiations are provable from the three equations in Example 4. However, the above equation itself is *not* derivable from the axioms in Example 4. (See Examples 7 and 8 for proofs of this claim.)

A finite basis for the algebra $\mathcal{N}$ is given by the following axiom system

$$\begin{aligned} x \vee 0 &\approx x \\ S(x) \vee S(y) &\approx S(x \vee y) \\ S(x) \vee x &\approx S(x) \\ x \vee x &\approx x \\ x \vee y &\approx y \vee x \quad \text{and} \\ x \vee (y \vee z) &\approx (x \vee y) \vee z \ . \end{aligned}$$

It turns out that completeness and ω-completeness are closely related properties of an axiom system. Indeed, assume that $\mathcal{A}$ is a Σ-algebra each of whose elements is denotable. Suppose that E is sound and complete with respect to $=_{\mathcal{A}}$. It is not hard to argue that, in this case, E is also ω-complete.

Remark 1. For the aforementioned connection between the model-theoretic notion of completeness and the proof-theoretic one of ω-completeness to hold, it is crucial that each element in the algebra $\mathcal{A}$ be denotable. To see this, consider the signature consisting of the constant $\perp$ and the unary function symbol P. Interpret this language over the algebra having $\{0, 1\}$ as carrier set, where $\perp$ is interpreted as 0, and P is interpreted as the constant function 0. We claim that no basis for this algebra can be ω-complete. To see that this holds, note, first of all, that each closed term over the aforementioned signature denotes the element 0. Therefore each closed instantiation of the equation $P(x) \approx x$ holds in the algebra, and is provable from the chosen basis. However, the equation $P(x) \approx x$ is itself *not* provable. This follows because E is sound, and that equation does not hold in the algebra, as can be seen by setting the variable x to 1.

Consider the Σ-algebra obtained by quotienting the set of closed terms $\mathrm{T}(\Sigma)$ with respect to the congruence relation that equates two closed terms t, u iff the equation $t \approx u$ is provable from an axiom system E. As a corollary of the aforementioned observation, we have that an equational basis for that algebra is also ω-complete.

One of the classic topics in the field of equational logic, and in its applications in process algebra, is the study of results pertaining to the existence or non-existence of finite bases for algebras. In the realm of concurrency theory, van Glabbeek presented in [23, 24] 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 ω-complete axiomatizations for them. In many cases this turns out to be a difficult question. Most of these finite basis questions have been settled, either positively by giving a finite ω-complete axiomatization, or negatively by proving that such an axiomatization does not exist. But some open questions remain. The main aim of this paper is to survey such results. Before doing so, however, we give a brief overview of some of the general proof techniques that have been developed in the literature on universal algebra, and more specifically within process algebra, to show that certain algebras afford a finite equational basis, or that no such basis exists. These strategies will then be used in Sections 3–5 to establish positive and negative results on the existence of finite bases for behavioural congruences over several process description languages.

2.2 Methods for Establishing Positive Results

Assume that we have an algebra $\mathcal{A}$ and a (finite) axiom system E that is sound with respect to $=_{\mathcal{A}}$. How can we show that E is complete or ground complete? There are a few general proof techniques that have been applied in the literature to answer this question, and we review some of those in the remainder of this section.

Normal Forms A classic strategy for showing that an axiom system is complete or ground complete that has had a wealth of applications in process algebra relies on the following two steps:

- **Isolation of normal forms.** In this step one finds a collection of terms, the so-called *normal forms*, with the property that each term t can be proved equal to a normal form using the equations in E. In other words, the set of normal forms is as expressive as the whole collection of terms modulo the equational theory generated by E. (If we are aiming at showing that our axiom system E is ground complete, then the normal forms are closed terms, and it suffices only to prove that each closed term is provably equal to a normal form using the equations in E.)
- **Distinctness of normal forms.** In this second step, one argues that two normal forms are related by $=_{\mathcal{A}}$ if, and only if, they are "identical". This is often done by showing that, for each pair of different normal forms, it is possible to construct an environment ρ distinguishing them.

In applications of this method in process algebra, the former step in this proof strategy is often carried out with the use of term rewriting techniques. In that case, the normal forms are precisely those of the term rewriting system, and the analysis is complicated by the need to consider rewriting modulo commutativity and associativity of certain operators like alternative composition. Moreover, the isolation of a suitable notion of normal form often requires considerable ingenuity, and is a difficult art.

Example 6. The aforementioned strategy based upon the isolation of suitable normal forms for terms can be used to show that the axiom system presented in Example 5 is, as claimed there, a finite basis for the algebra $\mathcal{N}$. Indeed, a suitable set of normal forms for terms over the signature of that algebra is given by the collection of terms of the form

$$\bigvee_{i \in I} S^{n_i}(x_i) \vee \bigvee_{j \in J} y_j \; [\vee S^n(0)] \ ,$$

where

- I and J are finite index sets,
- $n_i > 0$, for each $i \in I$, and
- the variables x_i ($i \in I$) and y_j ($j \in J$) are all different.

The notation $[\vee\{S^n(0)\}]$ used in defining normal forms means that the term $S^n(0)$ is optional. If that term is present then n must be larger than each of the n_i ($i \in I$).

It is not too hard to argue that

1. each term can be proven equal to a normal form using the equations in Example 5 and
2. if t and u are different normal forms, then there is an environment ρ mapping variables to natural numbers such that $\rho(t) \neq \rho(u)$.

Therefore, as claimed in Example 5, that axiom system is a finite basis for the algebra $\mathcal{N}$. Since each element of $\mathcal{N}$ is denotable (Example 3), E is also ω-complete.

Inverted Substitutions A proof technique that can be used to prove the ω-completeness of an axiom system, and that originates from research in process algebra, was offered by Groote in [26]. Groote's strategy is based on proof transformations, and proceeds as follows. Assume that we have an axiom system E, and an arbitrary equation $t \approx u$ all of whose closed instantiations are provable from E. The first step in Groote's "inverted substitutions" strategy is to find a closed substitution σ such that a proof of the equation $\sigma(t) \approx \sigma(u)$ from E can be transformed uniformly to a proof of the equation $t \approx u$. This proof transformation is achieved by means of a mapping $\hat{\sigma} : \mathrm{T}(\Sigma) \to \mathbb{T}(\Sigma)$ that intuitively maps each closed term representing a variable to the variable itself. This transformation yields the desired proof of the equation $t \approx u$ from E, provided that the technical conditions stated in the following theorem are met.

Theorem 1 (Groote [26]). *Let E be an axiom system over signature Σ. Assume that, for each equation $t \approx u$ all of whose closed instantiations can be proven from E, there exist a closed substitution σ and a mapping $\hat{\sigma} : T(\Sigma) \to \mathbb{T}(\Sigma)$, satisfying the following conditions:*

1. *E proves the equations $\hat{\sigma}(\sigma(t)) \approx t$ and $\hat{\sigma}(\sigma(u)) \approx u$,*
2. *for each operation symbol f and terms $u_1, \ldots, u_n, u'_1, \ldots, u'_n$, where n is the arity of f, the equation $\hat{\sigma}(f(u_1, \ldots, u_n)) \approx \hat{\sigma}(f(u'_1, \ldots, u'_n))$ is provable from those in E and the equations $u_i \approx u'_i$ and $\hat{\sigma}(u_i) \approx \hat{\sigma}(u'_i)$ $(1 \leq i \leq n)$ and*
3. *the equation $\hat{\sigma}(\sigma'(t_1)) \approx \hat{\sigma}(\sigma'(t_2))$ is provable from E for each $(t_1 \approx t_2) \in E$ and closed substitution σ'.*

Then E is ω-complete.

The strategy for proving the ω-completeness of axiom systems offered by the above result has been applied with success by Groote and other researchers in the field of process algebra, and, when applicable, often leads to simpler proofs than the standard one based on normal forms. As remarked by Groote in [26], the ω-completeness of the finite basis for the algebra $\mathcal{N}$ given in Example 5 *cannot* be shown using the technique in Theorem 1.

Giving Semantics to All Terms The algebras that are used in the field of process description languages to interpret terms over some signature Σ can all be characterized as the quotient algebra $\mathrm{T}(\Sigma)/\!\sim$ of closed terms over Σ modulo some notion of congruence $\sim$. The interpretation of a closed term in this algebra is its congruence class with respect to $\sim$, and two arbitrary terms are congruent if, and only if, so are all of their closed instantiations.

Another technique that has been developed in the field of process algebra to establish ω-completeness results for axiom systems relies on the following steps:

- **Define the congruence relation $\sim$ over all terms in $\mathbb{T}(\Sigma)$ directly.** The relation $\sim$ should, of course, be defined over $\mathbb{T}(\Sigma)$ in such a way that two terms are related by $\sim$ if, and only if, so are all of their closed instantiations. This means, in particular, that an equation $t \approx u$ is sound in the quotient algebra $\mathrm{T}(\Sigma)/\!\sim$ exactly when $t \sim u$ holds.
- **Completeness over terms.** In this second step, one proves that the candidate axiom system E is a basis for the quotient algebra of terms $\mathbb{T}(\Sigma)$ modulo $\sim$, and hence for the quotient algebra of *closed* terms $\mathrm{T}(\Sigma)$ modulo $\sim$. Since each element of the algebra $\mathrm{T}(\Sigma)/\!\sim$ is denotable, it follows that E is also ω-complete.

To the best of our knowledge, this technique was first applied in [36] by Milner to show completeness of his inference system for bisimulation equivalence over the regular fragment of the Calculus of Communicating Systems (CCS) [37].

Cover Equations This technique from Fokkink and Nain [20] is tailored to BCCSP. The aim is to obtain an explicit description of the equational theory for a particular semantics. The central idea is that if an equation $t \approx u$ is sound for BCCSP modulo some semantics in the linear time-branching time spectrum, then $u+t \approx t$ and $t+u \approx u$ are sound as well; and from the last two equations one can derive $t \approx u$. This implies that it is sufficient to only consider sound equations of the form $at + u \approx u$ (where a denotes an action and t, u are BCCSP terms). These are called the *cover equations*.

When the cover equations have been classified, one can proceed in two ways. Either one can determine an infinite family of cover equations that obstructs a finite basis, or one can determine a finite basis among the cover equations.

2.3 Methods for Establishing Negative Results

To prove that a set of equations cannot be derived from a given, possibly finite, subset of this set, we usually point out one specific equation in the superset, and prove that it is not derivable from the subset. To show that an equational theory—that is, the set of equations that hold in a given algebra—is not finitely based, we extend this reasoning by proving that for *each* finite subset of the theory, there is an equation that cannot be derived from this finite set. Often we obtain this result by establishing a stronger result: we identify a particular countably infinite sequence of equations in the theory with some suitable properties, and show that no finite subset of the theory can prove all of the equations in that sequence.

The proof techniques used for this purpose can roughly be divided into two categories: the model-theoretic techniques and the proof-theoretic ones. In what follows we will try to describe the essence of these two main methodologies.

Model-theoretic Techniques If a set of equations E is sound in an algebra $\mathcal{A}$, we say that $\mathcal{A}$ is a model for E. By Birkhoff's completeness theorem for equational logic [13], each equation that is derived from E holds in $\mathcal{A}$, if $\mathcal{A}$ is a model for E. Thus, to prove that an equation $t \approx u$ is *not* derivable from E it is sufficient to find an algebra that is a model for E but not of the equation $t \approx u$.

Example 7. In Example 5 we claimed that equation (2) is not derivable from the axioms in Example 4. As argued above, this can be proven by exhibiting a model of the axioms in Example 4 where $\vee$ is not idempotent. A simple example of such a model consists of the collection of all finite strings over the symbol a, where 0 is interpreted as the empty string, the unary operation symbol S is interpreted as the identity function, and $\vee$ is used to stand for concatenation.

In light of the previous observations, to prove that an equational theory is not finitely based, one may therefore proceed as follows:

- isolate a countably infinite collection of equations e_n ($n \geq 0$) in the equational theory,
- for each finite subset E of the equational theory, construct an algebra $\mathcal{A}_E$ that is a model of E, but in which some of the equations e_n fail.

Examples of the application of this strategy may be found in, e.g., [1, 2, 16, 22].

Proof-theoretic Techniques Recall that an equation $t \approx u$ is derivable from a set of equations E if there is a sequence $t_i \approx u_i$ ($1 \leq i \leq n$) of equations such that

- $t_n = t$ and $u_n = u$, and
- for each $1 \leq i \leq n$, the equation $t_i \approx u_i$ is obtained by applying one of the aforementioned inference rules using equations in E or some of the equations that precede it in the sequence as premises.

Proof-theoretic techniques aim at showing that $t \approx u$ is *not* derivable from E, by establishing that no such proof sequence exists. This is often done by finding a property of equations that

- holds true for each instantiation of the axioms in E,
- is preserved by the rules of equational logic—that is, if all of the equations that are premises of the rule have the property, then so does the conclusion of the rule—, and

– fails for the equation $t \approx u$.

This contradicts the existence of a proof for the equation $t \approx u$ from E, showing that $t \approx u$ is not derivable from that axiom system.

Example 8. The aforementioned proof-theoretic strategy can be used to give an alternative proof that the idempotence of $\vee$ is not derivable from the axioms in Example 4. To this end, observe that the left- and right-hand sides of each axiom in Example 4 contain the same number of occurrences of each variable. It is not hard to see that this property is preserved under equational derivations. On the other hand, the term $x \vee x$ contains two occurrences of the variable x, whereas the term x has only one. It follows that equation (2) is not derivable from the axioms in Example 4.

The proof-theoretic strategy we have just described can be applied to show that an equational theory is not finitely based as follows:

– isolate a countably infinite collection of equations e_n ($n \geq 0$) in the equational theory,
– for each finite subset E of the equational theory, show that there is a property of equations that is satisfied by all of the equations that can be derived from E, but that is not afforded by some of the equations e_n.

Proof-theoretic techniques have found wide application in establishing that algebras of processes do not afford a finite basis. In particular, all of the known proofs of the negative results we survey in Sections 4 and 5 are based on applications of the aforementioned proof-theoretic strategy.

Remark 2. An observation that can sometimes be used to show that an equational theory does not afford a finite equational axiomatization relies on the *compactness theorem* (see, e.g., [15]). Assume that we have an infinite axiomatization E for an equational theory T. If T had a finite axiomatization, then, by the compactness theorem, some finite subset of E would be a complete axiomatization for the theory T. Namely, since E is complete, each axiom in the finite axiomatization for T could be derived from E, and each of these derivations uses only finitely many axioms in E. To prove that T does not have a finite axiomatization, it therefore suffices to show that, for each finite subset E' of E, there is an equation in E that is not provable from E'. This can be achieved using either of the two general proof strategies described above. Applications of this proof methodology may be found in, e.g., [16, 17].

3 On Finite Bases for BCCSP

3.1 The Linear Time-Branching Time Spectrum

Van Glabbeek presented in [23, 24] the linear time-branching time spectrum of behavioural equivalences for finitely branching, concrete processes. In this section, for the sake of completeness, we define the semantics in this spectrum.

A *labelled transition system* contains a set of *states*, with typical element s, and a set of transitions $s \xrightarrow{a} s'$, where a ranges over some set of labels. The set $\mathcal{I}(s)$ consists of those labels a for which there exists a transition $s \xrightarrow{a} s'$.

First we define four semantics based on simulation.

Definition 4 (Simulations). *Assume a labelled transition system.*

- *A binary relation $\mathcal{R}$ on states is a* simulation *if $s_0\ \mathcal{R}\ s_1$ and $s_0 \xrightarrow{a} s_0'$ imply $s_1 \xrightarrow{a} s_1'$ with $s_0'\ \mathcal{R}\ s_1'$.*
- *A simulation $\mathcal{R}$ is a* ready simulation *if $s_0\ \mathcal{R}\ s_1$ and $a \notin \mathcal{I}(s_0)$ imply $a \notin \mathcal{I}(s_1)$.*
- *A simulation $\mathcal{R}$ is a* 2-nested simulation *if $\mathcal{R}^{-1}$ is included in a simulation.*
- *A* bisimulation *is a symmetric simulation.*

Next we define six semantics based on decorated versions of execution traces.

Definition 5 (Decorated Traces). *Assume a labelled transition system.*

- *A sequence $a_1 \cdots a_n$, with $n \geq 0$, is a* trace *of a state s_0 if there is a sequence of transitions $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots s_{n-1} \xrightarrow{a_n} s_n$. It is a* completed trace *of s_0 if moreover $\mathcal{I}(s_n) = \emptyset$.*
- *A pair $(a_1 \cdots a_n, X)$, with $n \geq 0$ and $X \subseteq A$, is a* ready pair *of a state s_0 if there is a sequence of transitions $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots s_{n-1} \xrightarrow{a_n} s_n$ with $\mathcal{I}(s_n) = X$. It is a* failure pair *of s_0 if $\mathcal{I}(s_n) \cap X = \emptyset$.*
- *A sequence $X_0 a_1 X_1 \ldots a_n X_n$, with $n \geq 0$ and $X_i \subseteq A$, is a* ready trace *of a state s_0 if there is a sequence of transitions $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots s_{n-1} \xrightarrow{a_n} s_n$ with $\mathcal{I}(s_i) = X_i$ for $i = 0, \ldots, n$. It is a* failure trace *of s_0 if $\mathcal{I}(s_i) \cap X_i = \emptyset$ for $i = 0, \ldots, n$.*

Finally, we define two semantics based on possible futures and on possible worlds.

Definition 6 (Possible Futures/Worlds). *Assume a labelled transition system.*

- *A pair $(a_1 \cdots a_n, X)$, with $n \geq 0$ and $X \subseteq A^*$, is a* possible future *of a state s_0 if there is a sequence of transitions $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots s_{n-1} \xrightarrow{a_n} s_n$ where X is the set of traces of s_n.*
- *A state s is* deterministic *if for each $a \in \mathcal{I}(s)$ there is exactly one state s' such that $s \xrightarrow{a} s'$, and moreover s' is deterministic.*
  A state s is a possible world *of a state s_0 if s is deterministic and $s\ \mathcal{R}\ s_0$ for some ready simulation $\mathcal{R}$.*

Two states s and s' are *simulation / ready simulation / 2-nested simulation equivalent* if there exist simulations / ready simulations / 2-nested simulations $\mathcal{R}_1$ and $\mathcal{R}_2$ with $s\,\mathcal{R}_1\,s'$ and $s'\,\mathcal{R}_2\,s$. They are *bisimilar* if there is a bisimulation that relates them. They are *possible futures / possible worlds / ready trace / failure trace / ready / failure / completed trace / trace equivalent* if they have the same possible futures / possible worlds / ready traces / failure traces / ready pairs / failure pairs / completed traces / traces.

The linear time-branching time spectrum is depicted in Figure 1, where a directed edge from one semantics to another means that the source of the edge is included in the target.

3.2 BCCSP

BCCSP is a basic process algebra for expressing finite process behaviour. Its signature consists of the constant $\mathbf{0}$, the binary operator $\_ + \_$ called *alternative composition*, and

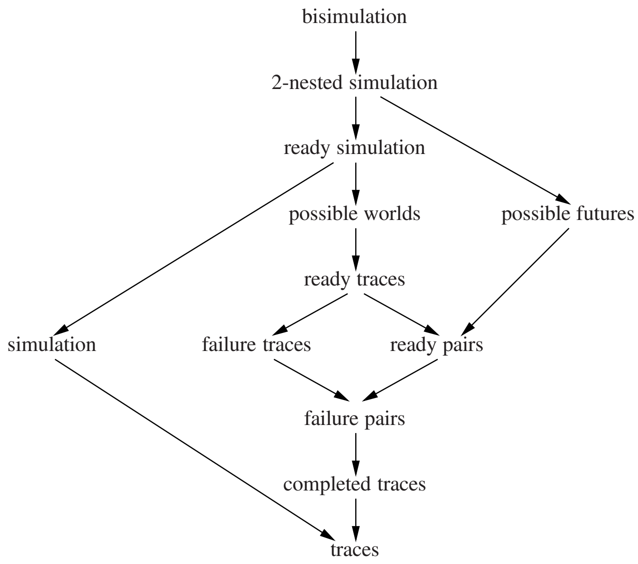

Fig. 1. The Linear Time-Branching Time Spectrum

unary *prefix* operators $a\_$, where a ranges over a nonempty set A of actions, called the *alphabet* (with typical elements a, b, c, d). Intuitively, closed BCCSP terms represent finite process behaviour, where **0** does not exhibit any behaviour, $p + q$ is the nondeterministic choice between the behaviours of p and q, and ap executes action a to transform into p. This intuition is captured by the transition rules below, in which a ranges over A. They give rise to A-labelled transitions between BCCSP terms.

$$\frac{}{ax \xrightarrow{a} x} \qquad \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$$

We use *summation* $\sum_{i=1}^{n} t_i$, with $n \geq 0$, to denote $t_1 + \cdots + t_n$, where the empty sum denotes **0**.

The semantics in the linear time-branching time spectrum all constitute a *congruence* for BCCSP, meaning that $p_1 \sim q_1$ and $p_2 \sim q_2$ imply $ap_1 \sim aq_1$ for $a \in A$ and $p_1 + p_2 \sim q_1 + q_2$, where $\sim$ ranges over the semantics in the spectrum.

3.3 Positive and Negative Results for BCCSP

In this section we will survey positive and negative results, and open questions, on the existence of a finite basis for the equational theories of BCCSP modulo the equivalences in the spectrum above. The axiomatizations that we will present for the different semantics in the spectrum were mostly taken from [24].

In case of an infinite alphabet, occurrences of action names in axioms are interpreted as variables (or action schemes).

Bisimulation The core axioms in Table 1 are sound and ground complete for BCCSP modulo bisimulation. Moller [38] proved using normal forms that this axiomatization is ω-complete; Groote provided an alternative proof of this result in [26] using inverted substitutions.

A1	$x + y$	$\approx y + x$
A2	$(x + y) + z$	$\approx x + (y + z)$
A3	$x + x$	$\approx x$
A6	$x + \mathbf{0}$	$\approx x$

Table 1. The axioms for bisimulation.

2-Nested Simulation and Possible Futures Aceto, Fokkink, van Glabbeek and Ingolfsdottir [4] proved that BCCSP modulo any semantics no coarser than possible futures and no finer than 2-nested simulation does not possess a finite sound and ground complete axiomatization. The infinite family of equations that they used to prove this negative result is defined as follows. Let E be any finite axiomatization for BCCSP that is sound modulo possible futures. Let the *depth* of a BCCSP term t be the largest number of transitions in sequence that t can exhibit. Pick an m such that

$$m > \max\{\, depth(t), depth(u) \mid (t \approx u) \in E\} \ .$$

For $n \geq 0$, let p_n and q_n be defined inductively as follows, for some $a \in A$:

$$\begin{array}{llll} p_0 & = a^{2m-1}\mathbf{0} & q_0 & = a^{m-1}\mathbf{0} \\ p_{n+1} & = ap_n + aq_n & q_{n+1} & = ap_n \end{array}$$

The equations $p_n \approx q_n$ for $n \geq 2$ are sound modulo 2-nested simulation. However, they cannot be derived from E.

Ready Simulation Van Glabbeek presented a conditional axiom for ready simulation equivalence: $\mathcal{I}(x) = \mathcal{I}(y) \Rightarrow a(x + y) \approx a(x + y) + ay$. Blom, Fokkink and Nain [14] showed that a sound and ground complete equational axiomatization for BCCSP modulo ready simulation exists. It is obtained by extending the four core axioms with

$$a(bx + by + z) \approx a(bx + by + z) + a(bx + z)$$

where a, b range over A. When A is infinite, Groote's technique of inverted substitutions can be applied to show that this axiomatization is ω-complete. When A is finite, it remains an open question whether BCCSP modulo ready simulation is finitely based.

Simulation A sound and ground complete axiomatization for BCCSP modulo simulation is obtained by extending the four core axioms with

$$a(x + y) \approx a(x + y) + ay$$

When A is infinite, Groote's technique of inverted substitutions can be applied to show that this axiomatization is ω-complete. When $1 < |A| < \infty$, it remains an open question whether BCCSP modulo simulation is finitely based. When $|A| = 1$, simulation equivalence coincides with trace equivalence, and we will see that in this case a finite basis does exist.

Possible Worlds A sound and ground complete axiomatization for BCCSP modulo possible worlds is obtained by extending the four core axioms with

$$a(bx + by + z) \approx a(bx + z) + a(by + z)$$

When A is infinite, Groote's technique of inverted substitutions can be applied to show that this axiomatization is ω-complete. Fokkink and Nain [20] showed that when $1 < |A| < \infty$, BCCSP modulo any semantics no coarser than ready equivalence and no finer than possible world equivalence does not possess a finite basis. (Note that ready traces are within this semantic range.) Their proof of this negative result, which uses cover equations and applies the compactness theorem to the equational theory for terms of depth 1, is based on the following infinite family of equations:

$$\begin{aligned}
&a(\sum_{i=1}^{|A|-1} x_i) + \sum_{j=1}^{|A|-1} a(\sum_{i=1}^{j-1} x_i + \sum_{i=j+1}^{n} x_i) + \sum_{j=|A|}^{n} a(\sum_{i=1}^{|A|-1} x_i + x_j + y_j) \approx \\
&\quad a(\sum_{i=1}^{|A|-1} x_i) + \sum_{j=1}^{|A|-1} a(\sum_{i=1}^{j-1} x_i + \sum_{i=j+1}^{n} x_i) + \sum_{j=|A|}^{n} a(\sum_{i=1}^{|A|-1} x_i + x_j + y_j) + a(\sum_{i=1}^{n} x_i)
\end{aligned}$$

These equations are sound modulo possible worlds for $n \geq |A|$. However, any finite axiomatization that is sound for BCCSP modulo ready pairs cannot derive them all. When $|A| = 1$, possible world equivalence coincides with completed trace equivalence, and we will see that in this case a finite basis does exist.

Ready Traces Van Glabbeek presented a conditional axiom for ready trace equivalence: $\mathcal{I}(x) = \mathcal{I}(y) \Rightarrow ax + ay \approx a(x + y)$. Blom, Fokkink and Nain [14] showed that when A is finite, a sound and ground complete equational axiomatization for BCCSP modulo ready traces exists. It is obtained by extending the four core axioms with

$$a(\sum_{i=1}^{|A|}(b_i x_i + b_i y_i) + z) \approx a(\sum_{i=1}^{|A|} b_i x_i + z) + a(\sum_{i=1}^{|A|} b_i y_i + z)$$

When A is infinite, they showed using the compactness theorem that a finite sound and ground complete axiomatization does not exist. Their proof is based on the following equations, for $n > 0$:

$$a(\sum_{i=1}^{n}(b_i c\mathbf{0} + b_i d\mathbf{0})) \approx a(\sum_{i=1}^{n} b_i c\mathbf{0}) + a(\sum_{i=1}^{n} b_i d\mathbf{0})$$

When $1 < |A| < \infty$, the aforementioned negative result from [20] (see the paragraph on possible worlds) implies that BCCSP modulo ready traces does not possess a finite basis. When $|A| = 1$, ready trace equivalence coincides with completed trace equivalence, and we will see that in this case a finite ω-complete axiomatization does exist.

Failure Traces Van Glabbeek presented a conditional axiom for failure traces (the same one as for ready traces). Blom, Fokkink and Nain [14] showed using normal forms that a sound and ground complete equational axiomatization for BCCSP modulo failure traces exists. It is obtained by extending the four core axioms with

$$a(bx + by + z) \approx a(bx + by + z) + a(by + z)$$
$$ax + ay \approx ax + ay + a(x + y)$$

When A is infinite, Groote's technique of inverted substitutions can be applied to show that this axiomatization is ω-complete. When $1 < |A| < \infty$, it remains an open question whether BCCSP modulo failure traces is finitely based. When $|A| = 1$, failure trace equivalence coincides with completed trace equivalence, and we will see that in this case a finite basis does exist.

Ready Pairs A sound and ground complete axiomatization for BCCSP modulo ready pairs is obtained by extending the four core axioms with

$$a(bx + by + z) \approx a(bx + by + z) + a(by + z)$$

When A is infinite, Groote's technique of inverted substitutions can be applied to show that this axiomatization is ω-complete. When $1 < |A| < \infty$, the aforementioned negative result from [20] (see the paragraph on possible worlds) implies that BCCSP modulo ready pairs does not possess a finite basis. When $|A| = 1$, ready equivalence coincides with completed trace equivalence, and we will see that in this case a finite basis does exist.

Failure Pairs A sound and ground complete axiomatization for BCCSP modulo failure pairs is obtained by extending the four core axioms with

$$a(bx + by + z) \approx a(bx + by + z) + a(bx + z)$$
$$ax + a(y + z) \approx ax + a(y + z) + a(x + y)$$

Fokkink and Nain [21] proved using cover equations that when A is infinite, this axiomatization is ω-complete. They also proved that when A is finite, one extra axiom is needed to obtain an ω-complete axiomatization:

$$a(\sum_{i=1}^{|A|} b_i x_i + y + z) \approx a(\sum_{i=1}^{|A|} b_i x_i + y + z) + a(\sum_{i=1}^{|A|} b_i x_i + y)$$

Completed Traces A sound and ground complete axiomatization for BCCSP modulo completed traces is obtained by extending the four core axioms with

$$a(bw + y) + a(cx + z) \approx a(bw + cx + y + z)$$

Groote [26] proved using normal forms that in order to obtain an ω-complete axiomatization, one extra axiom is needed:

$$ax + a(y + z) \approx ax + a(y + z) + a(x + y)$$

Traces A sound and ground complete axiomatization for BCCSP modulo traces is obtained by extending the four core axioms with

$$ax + ay \approx a(x + y)$$

Groote [26] proved using normal forms that this axiomatization is ω-complete when $|A| > 1$. When $|A| = 1$, it is not hard to see that one extra axiom, $ax + x \approx ax$, suffices to make the axiomatization ω-complete. Indeed, in that case, the algebra of closed BCCSP terms modulo trace equivalence is isomorphic to the algebra $\mathcal{N}$ in Example 3. (To the best of our knowledge this is the first time this last observation appears in print.)

3.4 Overview

Concluding, BCCSP has a finite sound and ground complete axiomatization for most of the semantics in the linear time-branching time spectrum. Only for 2-nested simulation and possible futures, and for ready traces in case of an infinite alphabet, such an axiomatization does not exist.

Regarding ω-completeness, matters are more mixed, especially when $1 < |A| < \infty$. The table below presents an overview, where **+** means that there a finite basis, **–** means that there is no finite basis, and **?** means that it is unknown whether a finite basis exists. We distinguish between an infinite alphabet, a finite alphabet with more than one element, and a singleton alphabet.

	$\lvert A\rvert = 1$	$1 < \lvert A\rvert < \infty$	$\lvert A\rvert = \infty$
bisim	+	+	+
2-nes sim	–	–	–
poss futu	–	–	–
ready sim	?	?	+
sim	+	?	+
poss worl	+	–	+
ready tr	+	–	–
failure tr	+	?	+
ready	+	–	+
failure	+	+	+
compl tr	+	+	+
traces	+	+	+

4 Parallelism

In this section we discuss extensions of BCCSP with a binary operation $\|$ for parallel composition. We only consider bisimulation semantics. The intuition is that $p \| q$ does a move from either component, or establishes some kind of synchronization between its components. The synchronization mechanism differs from one process description language to another. For the sake of generality, we make use of the mechanism incorporated in ACP, and show how it can be instantiated, e.g., to the synchronization mechanism of CCS.

ACP's synchronization mechanism presupposes a *communication function* γ, i.e., a partial function

$$\gamma : A \times A \rightharpoonup A$$

such that for all $a, b, c \in A$:

(i) if $\gamma(a,b)$ is defined, then so is $\gamma(b,a)$ and moreover $\gamma(a,b) = \gamma(b,a)$; and
(ii) $\gamma(a,\gamma(b,c))$ is defined iff $\gamma(\gamma(a,b),c)$ is defined, and if both are defined, then $\gamma(a,\gamma(b,c)) = \gamma(\gamma(a,b),c)$.

The operational semantics of $\|$ is then given by the following transition rules:

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'} \qquad \frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y',\ \gamma(a,b) = c}{x \parallel y \xrightarrow{c} x' \parallel y'}$$

By additional assumptions on γ we can obtain the different versions of parallel composition that are encountered in the literature; we give three examples:

1. The assumption $\gamma = \emptyset$ expresses that there is no communication at all, i.e., the operation $\|$ models *pure interleaving*.
2. The assumption that $\gamma(a,\gamma(b,c))$ is always undefined expresses that there is only *handshaking* communication.
3. We get the operation for parallel composition of CCS by assuming that
   (a) A contains a special action τ;
   (b) there is a bijection $\bar{\cdot}$ on $A-\{\tau\}$ such that $\bar{\bar{a}} = a$ and $\bar{a} \neq a$ for all $a \in A-\{\tau\}$;
   (c) $\gamma(a,\bar{a}) = \gamma(\bar{a},a) = \tau$ for all $a \in A - \{\tau\}$, and γ is undefined otherwise.

Let $\mathrm{BCCSP}_{\|}$ be the extension of BCCSP with $\|$. A ground complete axiomatization for $\mathrm{BCCSP}_{\|}$ modulo bisimulation equivalence is obtained by adding to the axioms A1–3,6 in Table 1 the equations generated by the so-called *Expansion Law*: for all $t = \sum_{i\in I} a_i x_i$ and $u = \sum_{j \in J} b_j y_j$:

$$t \parallel u \approx \sum_{i\in I} a_i\,(x_i \parallel u) + \sum_{j\in J} b_j\,(t \parallel y_j) + \sum_{i\in I}\sum_{j\in J} \gamma(a_i,b_j)(x_i \parallel y_j) \ , \tag{3}$$

with, for $i \in I$ and $j \in J$, the summand $\gamma(a_i,b_j)(t_i \parallel u_j)$ only present when $\gamma(a_i,b_j)$ is defined. The result was first established by Hennessy and Milner [30].

Since the Expansion Law generates infinitely many equations, the aforementioned ground complete axiomatization is infinite. If the set of actions A contains at least one element a such that $\gamma(a,a)$ is undefined, then a finite ground complete axiomatization is not possible, as shown by Moller [38, 40]. He establishes that there does not exist a finite set of $\mathrm{BCCSP}_{\|}$-equations, sound with respect to bisimulation equivalence, from which all equations of the form

$$a\mathbf{0} \parallel \varphi_n \approx a\varphi_n + \sum_{i=1}^{n} a a^n \quad (\text{with } \varphi_n = \sum_{i=1}^{n} a^i,\, n \geq 1) \tag{4}$$

are equationally derivable. Moller carries out his proof in a pure interleaving setting (i.e., $\gamma = \emptyset$), but it is easy to see that the assumption can be relaxed to: $\gamma(a,a)$ is undefined. First note that, with the relaxed requirement, the equations in (4) are still sound with respect to bisimulation equivalence. Now, suppose there does exist a finite basis E for $\mathrm{BCCSP}_{\|}$ modulo bisimulation equivalence. Then, since the equations in (4) are sound with respect to bisimulation equivalence, they are all derivable from E. Let

$E' \subseteq E$ be the set of equations in E that are involved in the derivations of the equations in (4). Then E' consists of equations in which no actions other than a occur (for if p and q are bisimulation equivalent closed $\mathrm{BCCSP}_{\|}$-terms, then p and q contain the same actions). Obviously, the equations in E' are all sound for $\mathrm{BCCSP}_{\|}$ with $A = \{a\}$ and $\gamma = \emptyset$, contradicting Moller's result.

Moller's result shows that for a finite axiomatization of parallel composition auxiliary operators are indispensable. Three such auxiliary operators have been proposed in the literature: Bergstra and Klop introduced the *left merge* ($\mathbin{\|\!\!\_}$) in [8] and the *communication merge* ($|$) in [10], and Hennessy [29] introduced an operation that we call *Hennessy's merge* ($|\!\!/$). In the remainder of this section we discuss these auxiliary operators in the context of BCCSP. In the next section we examine the consequences of replacing action prefixing in BCCSP by a binary operation for sequential composition.

4.1 Left merge

First we consider the special case of axiomatizing parallel composition under the pure interleaving assumption ($\gamma = \emptyset$). In that case, as can be seen from the transition rules for $\|$, a parallel composition $p \| q$ either does a move $p \xrightarrow{a} p'$ from its left component p and proceeds as $p' \| q$, or it does a move $q \xrightarrow{a} q'$ from its right component q and proceeds as $p \| q'$. So, intuitively, it is an alternative composition of two subprocesses. The auxiliary operation left merge is a device for expressing these subprocesses in terms of p and q; its operational semantics is given by the following transition rule:

$$\frac{x \xrightarrow{a} x'}{x \mathbin{\|\!\!\_}\, y \xrightarrow{a} x' \| y}$$

Using the left merge the intuition with respect to the behaviour of a parallel composition can be captured in a single equation:

$$\mathrm{M} \quad x \| y \approx x \mathbin{\|\!\!\_}\, y + y \mathbin{\|\!\!\_}\, x \ .$$

The axiom M and the axioms L1–3 in Table 2 allow the elimination of all occurrences of $\|$ and $\mathbin{\|\!\!\_}$ from closed terms. (Bergstra and Klop [10] established a similar result in the more general setting of ACP.) Hence, together with the axioms of BCCSP in Table 1, those equations constitute a ground complete axiomatization of $\mathrm{BCCSP}_{\|,\mathbin{\|\!\!\_}}$.

L1	$\mathbf{0} \mathbin{\|\!\!\_}\, x$	$\approx \mathbf{0}$
L2	$ax \mathbin{\|\!\!\_}\, y$	$\approx a(x \| y)$
L3	$(x + y) \mathbin{\|\!\!\_}\, z$	$\approx x \mathbin{\|\!\!\_}\, z + y \mathbin{\|\!\!\_}\, z$
L4	$(x \mathbin{\|\!\!\_}\, y) \mathbin{\|\!\!\_}\, z$	$\approx x \mathbin{\|\!\!\_}\, (y \| z)$
L5	$x \mathbin{\|\!\!\_}\, \mathbf{0}$	$\approx x$

Table 2. The axioms for left merge.

An ω-complete axiomatization is obtained by adding the axioms L4 and L5 in Table 2. Moller [38] proved this assuming that A is infinite. He used the technique based on normal forms: first he showed that every term is provably equal to a normal form, and then he argued that for distinct normal forms there is a distinguishing environment. Moller used a distinguishing environment that substitutes a special action (not already occurring in either normal form) for every variable, which is only possible if there are infinitely many actions. Both the proof that every term is provably equal to a normal form and the proof that normal forms can be distinguished are quite involved. It turns out that Groote's inverted substitutions technique also applies (see [26]), and the application is in fact quite straightforward.

The requirement that A is infinite seems essential for the application of Groote's technique. However, the authors have recently established that Moller's proof can be adapted with a distinguishing environment that only requires one action. So, if $\gamma = \emptyset$, then the axioms of BCCSP together with the axioms in Table 2 constitute a basis for $\mathrm{BCCSP}_{\|,\llfloor}$ modulo bisimulation equivalence, for each non-empty set of actions A. Hence, if A is finite, then $\mathrm{BCCSP}_{\|,\llfloor}$ modulo bisimulation equivalence is finitely based.

4.2 Communication merge

C1	$\mathbf{0} \mid x$	$\approx \mathbf{0}$	
C2	$ax \mid by$	$\approx c(x \parallel y)$	if $\gamma(a,b) = c$
C3	$ax \mid by$	$\approx \mathbf{0}$	if $\gamma(a,b)$ is undefined
C4	$(x+y) \mid z$	$\approx x \mid z + y \mid z$	
C5	$x \mid y$	$\approx y \mid x$	
C6	$(x \mid y) \mid z$	$\approx x \mid (y \mid z)$	
C7	$x \mid (y \llfloor z)$	$\approx (x \mid y) \llfloor z$	

Table 3. The axioms for communication merge.

If $p \xrightarrow{a} p'$ and $q \xrightarrow{b} q'$ and $\gamma(a,b) = c$, then the parallel composition $p \parallel q$ has the extra option to perform the synchronization move $p \parallel q \xrightarrow{c} p' \parallel q'$. The communication merge provides notation for this part of the behaviour of a parallel composition; its operational semantics is given by the following transition rule:

$$\frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y',\ \gamma(a,b) = c}{x \mid y \xrightarrow{c} x' \parallel y'}$$

Of course, if synchronization is possible, then the axiom M is not sound and needs to be replaced by:

$$\text{M'} \quad x \parallel y \approx (x \llfloor y + y \llfloor x) + x \mid y \ .$$

Using the axiom M', the axioms L1–3 in Table 2 and the axioms C1–5 in Table 3 all occurrences of $\|$, $\llfloor$ and $\mid$ can be eliminated from closed terms. Hence, together with the axioms of BCCSP in Table 1, those equations constitute a ground complete axiomatization of $\mathrm{BCCSP}_{\|,\llfloor,\mid}$.

Let us now consider ω-completeness. Note that it critically depends on γ whether certain equations between terms with variables are sound. For instance, the equation

$$x \mid y \approx \mathbf{0}$$

is sound if $\gamma = \emptyset$, but if there exist actions a and b such that $\gamma(a, b)$ is defined, then it is clearly not sound.

Groote [26] proved that if A is a commutative semigroup under γ (which means that γ is an associative and commutative total function on A), and A is moreover freely generated by some infinite subset, then the axioms of BCCSP in Table 1 together with M' and the axioms in Tables 2 and 3 constitute an ω-complete axiomatization. (Of course, since γ is total, the axiom C3 is superfluous in this axiomatization.) It is an open problem whether it is necessary to require A to be generated by an *infinite* subset.

If γ satisfies the requirement that $\gamma(a, \gamma(b, c))$ is undefined for all $a, b, c \in A$ (i.e., there is only handshaking communication), then the axiom

$$\mathrm{H} \quad x \mid y \mid z \approx \mathbf{0}$$

is sound. We conjecture that if A is non-empty and γ implements the CCS communication mechanism, then the axioms M' and H together with the axioms in Tables 1, 2 and 3 constitute an ω-complete axiomatization.

4.3 Hennessy's merge

In [29], Hennessy proposed another auxiliary operator, using it in his axiomatizations of observation congruence and timed congruence. *Hennessy's merge*, as we call it, combines the behaviour of the left merge and the communication merge. Its operational semantics is given by the following transition rules:

$$\frac{x \xrightarrow{a} x'}{x \mathbin{|\!\!/} y \xrightarrow{a} x' \parallel y} \qquad \frac{x \xrightarrow{a} x',\ y \xrightarrow{b} y',\ \gamma(a,b) = c}{x \mathbin{|\!\!/} y \xrightarrow{c} x' \parallel y'}$$

Note that with Hennessy's merge, parallel composition is definable with the following equation:

$$x \parallel y \approx x \mathbin{|\!\!/} y + y \mathbin{|\!\!/} x \ \ .$$

This may seem promising for the existence of a finite axiomatization of parallel composition that only uses Hennessy's merge as auxiliary operation. However, as was already conjectured by Bergstra and Klop in [10], it turns out that the operation itself cannot be finitely axiomatized. Assuming the CCS synchronization mechanism (see the beginning of Section 4), the authors recently proved in [3] that there does not exist a finite set of sound $\mathrm{BCCSP}_{\parallel, |\!\!/}$-equations from which all equations of the form

$$a\mathbf{0} \mathbin{|\!\!/} \psi_n \approx a\psi_n + \sum_{i=0}^{n} \tau a^i \quad (\text{with } \psi_n = \sum_{i=0}^{n} \bar{a}a^i, n \geq 0)$$

are equationally derivable.

4.4 Overview

In the table below we summarize the results and open problems discussed in this section. A **+** in the first (respectively, second) column means that there exists a *finite* ground complete (respectively, ω-complete) axiomatization, a **–** means that such an axiomatization does not exist, and a **?** means that it is unknown whether such an axiomatization exists.

	ground complete	ω-complete
$\mathrm{BCCSP}_{\parallel}$	**–**	**–**
$\mathrm{BCCSP}_{\parallel,\lfloor\!\lfloor}$	**+**	**+**
$\mathrm{BCCSP}_{\parallel,\lfloor\!\lfloor,\mid}$ (handshaking)	**+**	**?**
$\mathrm{BCCSP}_{\parallel,\not\mid}$	**–**	**–**

Moller's result shows that $\mathrm{BCCSP}_{\parallel}$ has no finite ground complete axiomatization. With the two auxiliary binary operations $\lfloor\!\lfloor$ and $\mid$ of Bergstra and Klop a finite ground complete axiomatization becomes possible. If one assumes pure interleaving, then adding only $\lfloor\!\lfloor$ suffices, and then there even exists a finite ω-complete axiomatization. It remains an open problem whether it is possible to axiomatize $\mathrm{BCCSP}_{\parallel}$ with arbitrary handshaking or the CCS synchronization mechanism adding only one auxiliary binary operation.

5 Sequential composition

In this section we discuss the consequences of having sequential composition instead of action prefixing. We remove the constant **0** and the unary prefixes $a\_$ from BCCSP, and replace them by a binary operation $\cdot$ for sequential composition, treating the actions in A as constant symbols. Thus we get the signature of BPA [10]. The transition rules for actions and sequential composition are as follows:

$$\frac{}{a \xrightarrow{a} \surd} \qquad \frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}$$

Note the special state $\surd$ that we use to write the transition rules; it signals successful termination. To make the rules work, we stipulate that $\surd \cdot x = x$ and $\surd \parallel x = x \parallel \surd = x$. We also require that bisimulations relate $\surd$ only to $\surd$.

A1	$x + y$	$\approx y + x$
A2	$x + (y + z)$	$\approx (x + y) + z$
A3	$x + x$	$\approx x$
A4	$(x + y) \cdot z$	$\approx x \cdot z + y \cdot z$
A5	$(x \cdot y) \cdot z$	$\approx x \cdot (y \cdot z)$

Table 4. The axioms for alternative and sequential composition.

The axioms of BPA are obtained by taking the first three axioms of BCCSP, adding that $\cdot$ distributes from the right over $+$ and that $\cdot$ is associative. For the sake of clarity, we list them all in Table 4. It is folklore that they constitute a ground complete axiomatization of BPA. Moreover, the axiomatization is ω-complete. As far as we know, this latter result does not explicitly appear in print, but a proof can be extracted from the ω-completeness proof for PA [19] that we discuss below.

In [39], Moller adapted his proof that $\mathrm{BCCSP}_{\|}$ is not finitely based to the setting with sequential composition. The infinite family of equations he uses to establish this result is obtained from the equations in (4) by the obvious translation (replace action prefixes by actions and sequential compositions, and omit all occurrences of **0**).

M1	$x \parallel y$	$\approx x \mathbin{\|\!\!\lfloor} y + y \mathbin{\|\!\!\lfloor} x$
M2	$a \mathbin{\|\!\!\lfloor} x$	$\approx a \cdot x$
M3	$a \cdot x \mathbin{\|\!\!\lfloor} y$	$\approx a \cdot (x \parallel y)$
M4	$(x + y) \mathbin{\|\!\!\lfloor} z$	$\approx x \mathbin{\|\!\!\lfloor} z + y \mathbin{\|\!\!\lfloor} z$
M5	$(x \mathbin{\|\!\!\lfloor} y) \mathbin{\|\!\!\lfloor} z$	$\approx x \mathbin{\|\!\!\lfloor} (y \parallel z)$
M6	$(x \cdot \alpha) \mathbin{\|\!\!\lfloor} \alpha$	$\approx (x \mathbin{\|\!\!\lfloor} \alpha) \cdot \alpha$

Table 5. The axioms for merge and left merge.

The signature of PA combines that of BPA with $\parallel$ and $\mathbin{\|\!\!\lfloor}$. Parallel composition in PA stands for pure interleaving (i.e., $\gamma = \emptyset$), so the relation between $\parallel$ and $\mathbin{\|\!\!\lfloor}$ is expressed by the axiom M1 in Table 5. For a ground complete axiomatization of PA it suffices to add the first four axioms in Table 5 to the axioms of BPA. Fokkink and Luttik proved in [19] that if M5 and M6 are added too, then the axiomatization is ω-complete. The α in M6 ranges over finite sums of distinct actions. The equation that results by replacing both occurrences of $\mathbin{\|\!\!\lfloor}$ in M6 with $\parallel$ is also sound (it is an instructive exercise to derive it using M6 and the other axioms). It is an example of a so-called *mixed equation*, equating a parallel composition and a sequential composition. There is a deep theory of mixed equations developed by Hirshfeld and Jerrum [31] for the benefit of their proof that bisimulation equivalence is decidable for normed PA. The proof in [19] that the presented axiomatization is ω-complete partly relies on their theory.

Incorporation of synchronization can be done by adding a communication merge and replacing the axiom M1 by M'. It is not difficult to adapt the axioms in Table 3 in such a way that all communication merges can be eliminated from closed terms; thus a ground complete axiomatization can be obtained. Note that this does require the addition of a special constant δ that will assume the rôle of **0**; it satisfies

$$\delta \cdot x \approx \delta$$
$$x + \delta \approx x \ .$$

There are no known ω-completeness results pertaining to the extension of BPA with $\parallel$, $\mid$ and δ. It would again be necessary to make some assumptions about the synchronization mechanism. If there is only handshaking communication, then the equations

$$(\cdots(((x_1 \mid x_2) \cdot y_1 \mathbin{\|\!\!\lfloor} z_1) \cdot y_2 \mathbin{\|\!\!\lfloor} z_2) \cdot \cdots y_n \mathbin{\|\!\!\lfloor} z_n) \mid x_3 \approx \delta \qquad (n \geq 0)$$

are sound. We conjecture that there does not exist a finite set of sound equations from which they are all derivable.

References

1. Luca Aceto, Zoltán Ésik, and Anna Ingolfsdottir. The max-plus algebra of the natural numbers has no finite equational basis. *Theoretical Computer Science*, 293(1):169–188, February 2003.
2. Luca Aceto, Wan Fokkink, and Anna Ingolfsdottir. A menagerie of non-finitely based process semantics over BPA*—from ready simulation to completed traces. *Mathematical Structures in Computer Science*, 8(3):193–230, June 1998.
3. Luca Aceto, Wan Fokkink, Anna Ingolfsdottir, and Bas Luttik. CCS with Hennessy's merge has no finite equational axiomatization. *Theoretical Computer Science*, 330(3):377–405, 2005.
4. Luca Aceto, Wan Fokkink, Rob van Glabbeek, and Anna Ingolfsdottir. Nested semantics over finite trees are equationally hard. *Information and Computation*, 191(2):203–232, June 2004.
5. Jos C. M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2–3):131–146, 2005.
6. Jos C. M. Baeten and P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.
7. Jos C.M. Baeten and Jan Willem Klop, editors. *Proceedings CONCUR 90*, Amsterdam, volume 458 of *Lecture Notes in Computer Science.* Springer-Verlag, 1990.
8. J.A. Bergstra and Jan Willem Klop. Fixed point semantics in process algebras. Report IW 206, Mathematisch Centrum, Amsterdam, 1982.
9. J.A. Bergstra and Jan Willem Klop. The algebra of recursively defined processes and the algebra of regular processes. In J. Paredaens, editor, *Proceedings 11^{th} ICALP, Antwerpen*, volume 172 of *Lecture Notes in Computer Science*, pages 82–95. Springer-Verlag, 1984.
10. J.A. Bergstra and Jan Willem Klop. Process algebra for synchronous communication. *Information and Control*, 60(1/3):109–137, 1984.
11. J.A. Bergstra and Jan Willem Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.
12. J.A. Bergstra and Jan Willem Klop. Algebra of communicating processes. In J.W. de Bakker, M. Hazewinkel, and J.K. Lenstra, editors, *Mathematics and Computer Science*, CWI Monograph 1, pages 89–138. North-Holland, Amsterdam, 1986.
13. Garrett Birkhoff. On the structure of abstract algebras. *Proc. Camb. Philos. Soc.*, 31:433–454, 1935.
14. Stefan Blom, Wan Fokkink, and Sumit Nain. On the axiomatizability of ready traces, ready simulation and failure traces. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Proceedings 30th Colloquium on Automata, Languages and Programming—ICALP'03, Eindhoven*, volume 2719 of *Lecture Notes in Computer Science*, pages 109–118. Springer-Verlag, 2003.
15. Stanley N. Burris and H.P. Sankappanavar. *A Course in Universal Algebra.* Graduate Texts in Mathematics. Springer-Verlag, 1981. The Millennium Edition of this book is available at `http://www.math.uwaterloo.ca/~snburris/htdocs/ualg.html`.
16. J. H. Conway. *Regular Algebra and Finite Machines.* Mathematics Series (R. Brown and J. De Wet eds.). Chapman and Hall, London, United Kingdom, 1971.
17. Zoltán Ésik and Satoshi Okawa. Series and parallel operations on pomsets. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science (Chennai, 1999)*,

volume 1738 of *Lecture Notes in Comput. Sci.*, pages 316–328. Springer-Verlag, Berlin, 1999.
18. Wan Fokkink. *Introduction to Process Algebra.* Texts in Theoretical Computer Science, An EATCS Series. Springer-Verlag, January 2000.
19. Wan Fokkink and Bas Luttik. An omega-complete equational specification of interleaving. In Ugo Montanari, J. Rolinn, and E. Welzl, editors, *Proceedings 27th Colloquium on Automata, Languages and Programming—ICALP'00, Geneva*, volume 1853 of *Lecture Notes in Computer Science*, pages 729–743. Springer-Verlag, July 2000.
20. Wan Fokkink and Sumit Nain. On finite alphabets and infinite bases: From ready pairs to possible worlds. In Igor Walukiewicz, editor, *Proceedings of Foundations of Software Science and Computation Structures, 7th International Conference, FOSSACS 2004*, volume 2897 of *Lecture Notes in Computer Science*, pages 182–194. Springer-Verlag, 2004.
21. Wan Fokkink and Sumit Nain. A finite basis for failure semantics. In *Proceedings of 32nd Colloquium on Automata, Languages and Programming - ICALP'05, Lisbon*, volume 3580 of *Lecture Notes in Computer Science*, pages 55–65. Springer-Verlag, July 2005. To appear.
22. Jay Loren Gischer. The equational theory of pomsets. *Theoretical Computer Science*, 61:199–224, 1988.
23. R. van Glabbeek. The linear time-branching time spectrum. In Baeten and Klop [7], pages 278–297.
24. R. van Glabbeek. The linear time-branching time spectrum. I. The semantics of concrete, sequential processes. In *Handbook of Process Algebra*, pages 3–99. North-Holland, Amsterdam, 2001.
25. R. van Glabbeek and P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.
26. Jan Friso Groote. A new strategy for proving ω-completeness with applications in process algebra. In Baeten and Klop [7], pages 314–331.
27. Jan Heering. Partial evaluation and ω-completeness of algebraic specifications. *Theoretical Computer Science*, 43(2-3):149–167, 1986.
28. M. Hennessy. *Algebraic Theory of Processes.* MIT Press, Cambridge, Massachusetts, 1988.
29. M. Hennessy. Axiomatising finite concurrent processes. *SIAM Journal on Computing*, 17(5):997–1017, 1988.
30. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.
31. Y. Hirshfeld and M. Jerrum. Bisimulation equivalence is decidable for normed process algebra. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Proc. of ICALP 1999*, LNCS 1644, pages 412–421, 1999.
32. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, Englewood Cliffs, 1985.
33. R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.
34. S.C. Kleene. Representation of events in nerve nets and finite automata. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.
35. Huimin Lin. An interactive proof tool for process algebras. In *9th Annual Symposium on Theoretical Aspects of Computer Science*, volume 577 of *Lecture Notes in Computer Science*, pages 617–618, Cachan, France, 13–15 February 1992. Springer.
36. R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.
37. R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.
38. F. Moller. *Axioms for Concurrency.* PhD thesis, Department of Computer Science, University of Edinburgh, July 1989. Report CST-59-89. Also published as ECS-LFCS-89-84.

39. F. Moller. The importance of the left merge operator in process algebras. In M. Paterson, editor, *Proceedings* 17^{th} *ICALP*, Warwick, volume 443 of *Lecture Notes in Computer Science*, pages 752–764. Springer-Verlag, July 1990.
40. F. Moller. The nonexistence of finite axiomatisations for CCS congruences. In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science*, Philadelphia, USA, pages 142–153. IEEE Computer Society Press, 1990.
41. Gordon D. Plotkin. The λ-calculus is ω-incomplete. *J. Symbolic Logic*, 39:313–317, 1974.
42. Gordon D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
43. Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60–61:17–139, 2004. This is a revised version of the original DAIMI memo [42].

Recent BRICS Report Series Publications

RS-05-19 Luca Aceto, , Willem Jan Fokkink, Anna Ingólfsdóttir, , and Bas Luttik. ***Finite Equational Bases in Process Algebra: Results and Open Questions*****. June 2005. 28 pp.**

RS-05-18 Peter Bogetoft, Ivan B. Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter, and Tomas Toft. ***Secure Computing, Economy, and Trust: A Generic Solution for Secure Auctions with Real-World Applications*****. June 2005. 37 pp.**

RS-05-17 Ivan B. Damgård, Thomas B. Pedersen, and Louis Salvail. ***A Quantum Cipher with Near Optimal Key-Recycling*****. May 2005.**

RS-05-16 Dariusz Biernacki, Olivier Danvy, and Kevin Millikin. ***A Dynamic Continuation-Passing Style for Dynamic Delimited Continuations*****. May 2005. ii+24 pp.**

RS-05-15 Małgorzata Biernacka and Olivier Danvy. ***A Concrete Framework for Environment Machines*****. May 2005. ii+25 pp.**

RS-05-14 Olivier Danvy and Henning Korsholm Rohde. ***On Obtaining the Boyer-Moore String-Matching Algorithm by Partial Evaluation*****. April 2005. ii+8 pp.**

RS-05-13 Dariusz Biernacki, Olivier Danvy, and Chung-chieh Shan. ***On the Dynamic Extent of Delimited Continuations*****. April 2005. ii+32 pp. Extended version of an article to appear in** ***Information Processing Letters*****. Subsumes BRICS RS-05-2.**

RS-05-12 Małgorzata Biernacka, Olivier Danvy, and Kristian Støvring. ***Program Extraction from Proofs of Weak Head Normalization*****. April 2005. 19 pp. Extended version of an article to appear in the preliminary proceedings of MFPS XXI, Birmingham, UK, May 2005.**

RS-05-11 Małgorzata Biernacka, Dariusz Biernacki, and Olivier Danvy. ***An Operational Foundation for Delimited Continuations in the CPS Hierarchy*****. March 2005. iii+42 pp. A preliminary version appeared in Thielecke, editor,** ***4th ACM SIGPLAN Workshop on Continuations*****, CW '04 Proceedings, Association for Computing Machinery (ACM) SIGPLAN Technical Reports CSR-04-1, 2004, pages 25–33. This version supersedes BRICS RS-04-29.**

RS-05-10 Dariusz Biernacki and Olivier Danvy. ***A Simple Proof of a Folklore Theorem about Delimited Control*****. March 2005. ii+11 pp.**