Bisimilarity is not Finitely Based over BPA with Interrupt

Luca Aceto[1,3], Wan Fokkink[4], Anna Ingolfsdottir[1,2], and Sumit Nain[1]

[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,nain@cs.aau.dk

[2] Department of Computer Science, University of Iceland, 107 Reykjavík, Iceland, annaing@hi.is

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

[4] Vrije Universiteit Amsterdam, Department of Computer Science, Section Theoretical Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands, wanf@cs.vu.nl

Abstract. This paper shows that bisimulation equivalence does not afford a finite equational axiomatization over the language obtained by enriching Bergstra and Klop's Basic Process Algebra with the interrupt operator. Moreover, it is shown that the collection of closed equations over this language is also not finitely based. In sharp contrast to these results, the collection of closed equations over the language BPA enriched with the disrupt operator is proven to be finitely based.

1 Introduction

Programming and specification languages often include constructs to specify mode switches (see, e.g., [8, 11, 23, 24, 26]). Indeed, some form of mode transfer in computation appears in the time-honoured theory of operating systems in the guise of, e.g., interrupts, in programming languages as exceptions, and in the behaviour of control programs and embedded systems as discrete "mode switches" triggered by changes in the state of their environment.

In light of the ubiquitous nature of mode changes in computation, it is not surprising that classic process description languages either include primitive operators to describe mode changes—for example, LOTOS [15, 23] offers the so-called *disruption operator*—or have been extended with variations on mode transfer operators. For instance, examples of such operators that may be added to CCS are discussed by Milner in [25, pp. 192–193], and the reference [17] offers some discussion of the benefits of adding one of those, viz. the *checkpointing operator*, to that language.

In the setting of Basic Process Algebra (BPA), as introduced by Bergstra and Klop in [12], some of these extensions, and their relative expressiveness, have been discussed in the early paper [11]. That preprint of Bergstra's has later been revised and extended in [7]. There, Baeten and Bergstra study the equational theory and expressiveness of BPA_δ (the extension of BPA with a constant δ to describe "deadlock") enriched with two mode transfer operators, viz. the *disrupt* and *interrupt* operators. In particular, they offer

an equational axiomatization of bisimulation equivalence [25, 29] over the resulting extension of the language BPA_δ. This axiomatization is finite, if so is the underlying set of actions—a state of affairs that is most pleasing for process algebraists.

However, the axiomatization of bisimulation equivalence offered by Baeten and Bergstra in [7] relies on the use of four auxiliary operators—two per mode transfer operator. Although the use of auxiliary operators in the axiomatization of behavioral equivalences over process description languages has been well established since Bergstra and Klop's axiomatization of parallel composition using the left and communication merge operators [13], to our mind, a result like the aforementioned one always begs the question whether the use of auxiliary operators is necessary to obtain a finite axiomatization of bisimulation equivalence.

For the case of parallel composition, Moller showed in [27, 28] that strong bisimulation equivalence is not finitely based over CCS [25] and PA [13] without the left merge operator. (The process algebra PA [13] contains a parallel composition operator based on pure interleaving without communication, and the left merge operator.) Thus auxiliary operators are necessary to obtain a finite axiomatization of parallel composition. But, is the use of auxiliary operators necessary to give a finite axiomatization of bisimulation equivalence over the language BPA enriched with the mode transfer operators studied by Baeten and Bergstra in [7]?

We address the above natural question in this paper. In particular, we focus on BPA enriched with the interrupt operator. Intuitively, "p interrupted by q" describes a process that normally behaves like p. However, at each point of the computation before p terminates, q can interrupt it, and begin its execution. If this happens, p resumes its computation upon termination of q.

We show that, in the presence of two distinct actions, bisimulation equivalence is *not* finitely based over BPA with the interrupt operator. Moreover, we prove that the collection of closed equations over this language is also not finitely based. This result provides evidence that the use of auxiliary operators in the technical developments presented in [7] is indeed necessary in order to obtain a finite axiomatization of bisimulation equivalence.

Our main result adds the interrupt operator to the list of operators whose addition to a process algebra spoils finite axiomatizability modulo bisimulation equivalence; see, e.g., [4, 3, 14, 16, 20, 30, 31] for other examples of non-finite axiomatizability results over process algebras, and some of their precursors in the setting of formal language theory. Of special relevance for concurrency theory are the aforementioned results of Moller's to the effect that the process algebras CCS and PA without the auxiliary left merge operator from [12] do not have a finite equational axiomatization modulo bisimulation equivalence [27, 28]. Recently, in collaboration with Luttik, the first three authors have shown in [5] that the process algebra obtained by adding Hennessy's merge operator from [22] to CCS does not have a finite equational axiomatization modulo bisimulation equivalence. This result is in sharp contrast with a theorem established by Fokkink and Luttik in [18] to the effect that the process algebra PA [13] affords an ω-complete axiomatization that is finite if so is the underlying set of actions. Aceto, Ésik and Ingolfsdottir proved in [2] that there is no finite equational axiomatization that is ω-complete for the max-plus algebra of the natural numbers, a result whose process

algebraic implications are discussed in [1]. Fokkink and Nain have shown in [19] that no congruence over the language BCCSP, a basic formalism to express finite process behaviour, that is included in possible worlds equivalence, and includes ready trace equivalence, affords a finite ω-complete equational axiomatization.

The paper is organized as follows. We begin by presenting the language BPA with the interrupt operator, its operational semantics and preliminaries on equational logic in Section 2. There we also show that the interrupt operator is not definable in BPA modulo bisimilarity. The general structure of the proof of our main result, to the effect that bisimilarity is not finitely based over the language we consider in this paper, is presented in Section 3. In that section, we also show how to reduce the proof of our main result to that of a technical statement describing a key property of closed instantiations of sound equations that is preserved under equational derivations (Proposition 3). We conclude the paper by showing in Section 4 that, in sharp contrast to the main result of the paper, the use of auxiliary operators is *not* necessary in order to obtain a finite axiomatization of bisimulation equivalence over closed terms in the language obtained by enriching BPA with the disrupt operator from [7].

2 Preliminaries

We begin by introducing the basic definitions and results on which the technical developments to follow are based. The interested reader is referred to [7, 12] for more information.

2.1 The Language $\mathrm{BPA}_{\mathbf{int}}$

We assume a non-empty alphabet A of atomic actions, with typical elements a, b. The language for processes we shall consider in this paper, henceforth referred to as $\mathrm{BPA}_{\mathrm{int}}$, is obtained by adding the interrupt operator from [7] to Bergstra and Klop's BPA [12]. This language is given by the following grammar:

$$t ::= x \mid a \mid t \cdot t \mid t + t \mid t \rhd t \ ,$$

where x is a variable drawn from a countably infinite set V and a is an action. In the above grammar, we use the symbol $\rhd$ for the *interrupt operator*. We shall use the meta-variables t, u, v, w to range over process terms, and write $var(t)$ for the collection of variables occurring in the term t. The *size* of a term is the number of operator symbols in it. A process term is *closed* if it does not contain any variables. Closed terms will be typically denoted by p, q, r, s. As usual, we shall often write tu in lieu of $t \cdot u$, and we assume that $\cdot$ binds stronger than $+$.

A (closed) substitution is a mapping from process variables to (closed) $\mathrm{BPA}_{\mathrm{int}}$ 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. In what follows, we shall use the notation $\sigma[x \mapsto p]$, where σ is a closed substitution and p is a closed $\mathrm{BPA}_{\mathrm{int}}$ term, to stand for the substitution mapping x to p, and acting like σ on all of the other variables in V.

In the remainder of this paper, we let a^1 denote a, and a^{m+1} denote $a(a^m)$. Moreover, we consider terms modulo associativity and commutativity of $+$. In other words, we do not distinguish $t + u$ and $u + t$, nor $(t + u) + v$ and $t + (u + v)$. This is justified because $+$ is associative and commutative with respect to the notion of equivalence we shall consider over BPA_{int}. (See axioms A1, A2 in Table 3.) In what follows, the symbol $=$ will denote equality modulo associativity and commutativity of $+$.

We say that a term t *has $+$ as head operator* if $t = t_1 + t_2$ for some terms t_1 and t_2. For example, $a + b$ has $+$ as head operator, but $(a + b)a$ does not.

For $k \geq 1$, we use a *summation* $\sum_{i\in\{1,\ldots,k\}} t_i$ to denote $t_1 + \cdots + t_k$. It is easy to see that every BPA_{int} term t has the form $\sum_{i\in I} t_i$, for some finite, non-empty index set I, and terms t_i ($i \in I$) that do not have $+$ as head operator. The terms t_i ($i \in I$) will be referred to as the *(syntactic) summands* of t. For example, the term $(a + b)a$ has only itself as (syntactic) summand.

The operational semantics for the language BPA_{int} is given by the labelled transition system

$$\left(\text{BPA}_{\text{int}}, \left\{\xrightarrow{a} \mid a \in A\right\}, \left\{\xrightarrow{a}\checkmark \mid a \in A\right\}\right) \ ,$$

where the transition relations $\xrightarrow{a}$ and the unary predicates $\xrightarrow{a}\checkmark$ are, respectively, the least subsets of $\text{BPA}_{\text{int}} \times \text{BPA}_{\text{int}}$ and BPA_{int} satisfying the rules in Table 1. Intuitively, a transition $t \xrightarrow{a} u$ means that the system represented by the term t can perform the action a, thereby evolving into u. The special symbol $\checkmark$ stands for (successful) termination; therefore the interpretation of the statement $t \xrightarrow{a}\checkmark$ is that the process term t can terminate by performing a. Note that, for every closed term p, there is some action a for which either $p \xrightarrow{a} p'$ holds for some p', or $p \xrightarrow{a}\checkmark$ does.

For terms t, u, and action a, we say that u is an *a-derivative* of t if $t \xrightarrow{a} u$.

$$\frac{}{a \xrightarrow{a}\checkmark} \qquad \frac{t \xrightarrow{a}\checkmark}{t + u \xrightarrow{a}\checkmark} \qquad \frac{u \xrightarrow{a}\checkmark}{t + u \xrightarrow{a}\checkmark} \qquad \frac{t \xrightarrow{a} t'}{t + u \xrightarrow{a} t'} \qquad \frac{u \xrightarrow{a} u'}{t + u \xrightarrow{a} u'}$$

$$\frac{t \xrightarrow{a}\checkmark}{t \cdot u \xrightarrow{a} u} \qquad \frac{t \xrightarrow{a} t'}{t \cdot u \xrightarrow{a} t' \cdot u}$$

$$\frac{t \xrightarrow{a}\checkmark}{t \rhd u \xrightarrow{a}\checkmark} \qquad \frac{t \xrightarrow{a} t'}{t \rhd u \xrightarrow{a} t' \rhd u} \qquad \frac{u \xrightarrow{a}\checkmark}{t \rhd u \xrightarrow{a} t} \qquad \frac{u \xrightarrow{a} u'}{t \rhd u \xrightarrow{a} u' \cdot t}$$

Table 1. Transition Rules for BPA_{int}

The transition relations $\xrightarrow{a}$ naturally compose to determine the possible effects that performing a sequence of actions may have on a BPA_{int} term.

Definition 1. *For a sequence of actions $a_1 \cdots a_k$ ($k \geq 0$), and BPA_{int} terms t, t', we write $t \xrightarrow{a_1\cdots a_k} t'$ iff there exists a sequence of transitions*

$$t = t_0 \xrightarrow{a_1} t_1 \xrightarrow{a_2} \cdots \xrightarrow{a_k} t_k = t' \ .$$

Similarly, we say that $a_1 \cdots a_k$ ($k \geq 1$) is a termination trace of a BPA$_{int}$ *term t iff there exists a term t' such that*

$$t \stackrel{a_1 \cdots a_{k-1}}{\longrightarrow} t' \stackrel{a_k}{\rightarrow} \checkmark \ .$$

If $t \stackrel{a_1 \cdots a_k}{\longrightarrow} t'$ holds for some BPA$_{int}$ *term t', or $a_1 \cdots a_k$ is a termination trace of t, then $a_1 \cdots a_k$ is a trace of t.*

The depth of a term t, written $depth(t)$, is the length of the longest trace it affords.

The norm of a term t, denoted by $norm(t)$, is the length of its shortest termination trace; this notion stems from [9].

The depth and the norm of closed terms can also be characterized inductively thus:

$$\begin{aligned} depth(a) &= 1 \\ depth(p + q) &= \max\{depth(p), depth(q)\} \\ depth(pq) &= depth(p) + depth(q) \\ depth(p \rhd q) &= depth(p) + depth(q) \end{aligned}$$

$$\begin{aligned} norm(a) &= 1 \\ norm(p + q) &= \min\{norm(p), norm(q)\} \\ norm(pq) &= norm(p) + norm(q) \\ norm(p \rhd q) &= norm(p) \ . \end{aligned}$$

Note that the depth and the norm of each closed BPA$_{\text{int}}$ term are positive.

In what follows, we shall sometimes need to consider the possible origins of a transition of the form $\sigma(t) \stackrel{a}{\rightarrow} p$, for some action a, closed substitution σ, BPA$_{\text{int}}$ term t and closed term p. Naturally enough, we expect that $\sigma(t)$ affords that transition if $t \stackrel{a}{\rightarrow} t'$, for some t' such that $p = \sigma(t')$. However, the above transition may also derive from the initial behaviour of some closed term $\sigma(x)$, provided that the collection of initial moves of $\sigma(t)$ depends, in some formal sense, on that of the closed term substituted for the variable x. Similarly, we shall sometimes need to consider the possible origins of a transition of the form $\sigma(t) \stackrel{a}{\rightarrow} \checkmark$, for some action a, closed substitution σ and BPA$_{\text{int}}$ term t.

To fully describe these situations, we introduce the auxiliary notion of confi guration of a BPA$_{\text{int}}$ term. To this end, we assume a set of symbols

$$V_d = \{x_d \mid x \in V\}$$

disjoint from V. Intuitively, the symbol x_d (read "during x") will be used to denote that the closed term substituted for variable x has begun executing, but has not yet terminated.

Definition 2. *The collection of* BPA$_{int}$ *confi gurations is given by the following grammar:*

$$c ::= t \mid x_d \mid c \cdot t \mid c \rhd t \ ,$$

where t is a BPA$_{int}$ *term, and $x_d \in V_d$.*

For example, the confi guration $x_d \cdot (a \rhd x)$ is meant to describe a state of the computation of some term in which the (closed term substituted for the) occurrence of variable x

on the left-hand side of the $\cdot$ operator has begun its execution (and has not terminated), but the one on the right-hand side has not. Note that each configuration contains at most one occurrence of an $x_d \in V_d$.

We shall consider the symbols x_d as variables, and use the notation $\sigma[x_d \mapsto p]$, where σ is a closed substitution and p is a closed $\mathrm{BPA}_{\mathrm{int}}$ term, to stand for the substitution mapping x_d to p, and acting like σ on all of the other variables.

$$\frac{}{x \xrightarrow{x_s} x_d} \qquad \frac{}{x \xrightarrow{x} \surd}$$

$$\frac{t \xrightarrow{x} t'}{t+u \xrightarrow{x} t'} \qquad \frac{t \xrightarrow{x_s} c}{t+u \xrightarrow{x_s} c} \qquad \frac{t \xrightarrow{x} \surd}{t+u \xrightarrow{x} \surd}$$

$$\frac{u \xrightarrow{x} u'}{t+u \xrightarrow{x} u'} \qquad \frac{u \xrightarrow{x_s} c}{t+u \xrightarrow{x_s} c} \qquad \frac{u \xrightarrow{x} \surd}{t+u \xrightarrow{x} \surd}$$

$$\frac{t \xrightarrow{x} t'}{tu \xrightarrow{x} t'u} \qquad \frac{t \xrightarrow{x_s} c}{tu \xrightarrow{x_s} cu} \qquad \frac{t \xrightarrow{x} \surd}{tu \xrightarrow{x} u}$$

$$\frac{t \xrightarrow{x} t'}{t \rhd u \xrightarrow{x} t' \rhd u} \qquad \frac{t \xrightarrow{x_s} c}{t \rhd u \xrightarrow{x_s} c \rhd u} \qquad \frac{t \xrightarrow{x} \surd}{t \rhd u \xrightarrow{x} \surd}$$

$$\frac{u \xrightarrow{x} u'}{t \rhd u \xrightarrow{x} u't} \qquad \frac{u \xrightarrow{x_s} c}{t \rhd u \xrightarrow{x_s} ct} \qquad \frac{u \xrightarrow{x} \surd}{t \rhd u \xrightarrow{x} t}$$

Table 2. SOS Rules for the Auxiliary Transitions $\xrightarrow{x}$, $\xrightarrow{x_s}$ and $\xrightarrow{x}\surd$ ($x \in V$)

The way in which the initial behaviour of a term may depend on that of the variables that occur in it is formally described by three auxiliary transition relations whose elements have the following forms:

- $t \xrightarrow{x_s} c$ (read "t can start executing x and become c in doing so"), where t is a term, x is a variable, and c is a configuration,
- $t \xrightarrow{x} t'$, where t and t' are terms and x is a variable, or
- $t \xrightarrow{x} \surd$, where t is a term.

The first of these types of transitions will be used to account for those transitions of the form $\sigma(t) \xrightarrow{a} p$ that are due to a-labelled transitions of the closed term $\sigma(x)$ that do not lead to its termination. The second will describe the origin of transitions of the form $\sigma(t) \xrightarrow{a} \sigma(t')$ that are due to a-labelled transitions of the closed term $\sigma(x)$ that lead to its termination. Finally, transitions of the third kind will allow us to describe the origin of termination transitions of the form $\sigma(t) \xrightarrow{a} \surd$ that are due to a-labelled termination transitions of the closed term $\sigma(x)$.

The SOS rules defining these transitions are given in Table 2. In those rules, the meta-variables t, u, t' and u' denote $\mathrm{BPA}_{\mathrm{int}}$ terms, and c ranges over the collection of configurations that contain one occurrence of a symbol of the form x_d. The attentive reader might have already noticed that the left-hand sides of the rules in Table 2 are

always BPA_{int} terms, and therefore that no transitions are possible from configurations that contain one occurrence of a symbol of the form x_d. This is in line with our aim in defining the auxiliary transition relations $\xrightarrow{x}$, $\xrightarrow{x_s}$ and $\xrightarrow{x}\surd$ ($x \in V$), viz. to describe the possible origins of the *initial* transitions of a term of the form $\sigma(t)$, with t a BPA_{int} term and σ a closed substitution.

Lemma 1. *For each* BPA_{int} *term* t*, configuration* c *and variable* x*, if* $t \xrightarrow{x_s} c$*, then* x_d *occurs in* c*. Moreover, if* $c = x_d$ *then* x *is a summand of* t*.*

The precise connection between the transitions of a term $\sigma(t)$ and those of t is expressed by the following lemma.

Lemma 2 (Operational Correspondence). *Assume that* t *is a* BPA_{int} *term,* σ *is a closed substitution and* a *is an action. Then the following statements hold:*

1. *If* $t \xrightarrow{a}\surd$*, then* $\sigma(t) \xrightarrow{a}\surd$*.*
2. *If* $t \xrightarrow{x}\surd$ *and* $\sigma(x) \xrightarrow{a}\surd$*, then* $\sigma(t) \xrightarrow{a}\surd$*.*
3. *If* $t \xrightarrow{x} t'$ *and* $\sigma(x) \xrightarrow{a}\surd$*, then* $\sigma(t) \xrightarrow{a} \sigma(t')$*.*
4. *Assume that* $t \xrightarrow{x_s} c$ *and* $\sigma(x) \xrightarrow{a} p$*, for some closed term* p*. Then*
$$\sigma(t) \xrightarrow{a} \sigma[x_d \mapsto p](c) \ .$$
5. *If* $t \xrightarrow{a} t'$*, then* $\sigma(t) \xrightarrow{a} \sigma(t')$*.*
6. *Assume that* $\sigma(t) \xrightarrow{a}\surd$*. Then either* $t \xrightarrow{a}\surd$ *or there is a variable* x *such that* $t \xrightarrow{x}\surd$ *and* $\sigma(x) \xrightarrow{a}\surd$*.*
7. *Assume that* $\sigma(t) \xrightarrow{a} p$*, for some closed term* p*. Then one of the following possibilities applies:*
   - $t \xrightarrow{x} t'$*,* $\sigma(x) \xrightarrow{a}\surd$ *and* $p = \sigma(t')$*, for some term* t' *and variable* x*,*
   - $t \xrightarrow{a} t'$ *for some term* t' *such that* $p = \sigma(t')$*, or*
   - $t \xrightarrow{x_s} c$ *and* $\sigma(x) \xrightarrow{a} q$*, for some variable* x*, configuration* c *and closed term* q *such that* $\sigma[x_d \mapsto q](c) = p$*.*

In this paper, we consider the language BPA_{int} modulo bisimulation equivalence [29].

Definition 3. *Two closed* BPA_{int} *terms* p *and* q *are bisimilar, denoted by* $p \leftrightarrows q$*, if there exists a symmetric binary relation* $\mathcal{B}$ *over closed* BPA_{int} *terms which relates* p *and* q*, such that:*

- *if* $r\ \mathcal{B}\ s$ *and* $r \xrightarrow{a} r'$*, then there is a transition* $s \xrightarrow{a} s'$ *such that* $r'\ \mathcal{B}\ s'$*;*
- *if* $r\ \mathcal{B}\ s$ *and* $r \xrightarrow{a}\surd$*, then* $s \xrightarrow{a}\surd$*.*

Such a relation $\mathcal{B}$ *will be called a bisimulation. The relation* $\leftrightarrows$ *will be referred to as bisimulation equivalence or bisimilarity.*

It is well known that $\leftrightarrows$ is an equivalence relation [29]. Moreover, the transition rules in Table 1 are in the 'path' format of Baeten and Verhoef [10]. Hence, bisimulation equivalence is a congruence with respect to all the operators in the signature of BPA_{int}.

Note that bisimilar closed BPA_{int} terms afford the same finite non-empty collection of (termination) traces, and therefore have the same norm and depth.

Bisimulation equivalence is extended to arbitrary BPA_{int} terms thus:

Definition 4. *Let t, u be* BPA$_{int}$ *terms. Then $t \leftrightarrow u$ iff $\sigma(t) \leftrightarrow \sigma(u)$ for every closed substitution σ.*

For instance, we have that

$$x \rhd y \leftrightarrow (x \rhd y) + yx$$

because, as our readers can easily check, the terms $p \rhd q$ and $(p \rhd q) + qp$ have the same set of initial "capabilities", i.e.,

$$p \rhd q \xrightarrow{a} r \text{ iff } (p \rhd q) + qp \xrightarrow{a} r \ , \text{ for each } a \text{ and } r, \text{ and}$$
$$p \rhd q \xrightarrow{a} \checkmark \text{ iff } (p \rhd q) + qp \xrightarrow{a} \checkmark, \text{ for each } a \ .$$

It is natural to expect that the interrupt operator cannot be defined in the language BPA modulo bisimulation equivalence. This expectation is confirmed by the following simple, but instructive, result:

Proposition 1. *There is no* BPA$_{int}$ *term t such that t does not contain occurrences of the interrupt operator, and $t \leftrightarrow x \rhd y$.*

Proof. Assume, towards a contradiction, that t is a BPA$_{\text{int}}$ term such that t does not contain occurrences of the interrupt operator, and $t \leftrightarrow x \rhd y$.

Consider the closed substitution σ_a mapping each variable to a. Since

$$\sigma_a(t) \leftrightarrow a \rhd a \text{ and } a \rhd a \xrightarrow{a} \checkmark \ ,$$

we have that $\sigma_a(t) \xrightarrow{a} \checkmark$. Lemma 2(6) yields that either $t \xrightarrow{a} \checkmark$ or there is a variable z such that $t \xrightarrow{z} \checkmark$ and $\sigma_a(z) \xrightarrow{a} \checkmark$. We shall now argue that both of these possibilities imply that $t \not\leftrightarrow x \rhd y$, contradicting our assumption.

Indeed, using the former possibility and Lemma 2(1), we may infer that

$$\sigma_a[x \mapsto a^2](t) \xrightarrow{a} \checkmark \ .$$

This implies that $t \not\leftrightarrow x \rhd y$, because $a^2 \rhd a$ does not have termination traces of length 1.

Assume now that there is a variable z such that $t \xrightarrow{z} \checkmark$ and $\sigma_a(z) \xrightarrow{a} \checkmark$. It is not hard to see that $t \leftrightarrow z + u$ for some term u, since t does not contain occurrences of the interrupt operator and $t \xrightarrow{z} \checkmark$. We claim that

$$\sigma_a[x \mapsto a^2](t) \not\leftrightarrow a^2 \rhd a \ .$$

If $z \neq x$, our claim follows, because, reasoning as above,

$$\sigma_a[x \mapsto a^2](t) \leftrightarrow a + \sigma_a[x \mapsto a^2](u) \xrightarrow{a} \checkmark$$

whereas $a^2 \rhd a$ does not have termination traces of length 1.

If $t \leftrightarrow x + u$, then $\sigma_a[x \mapsto a^2](t) \xrightarrow{a} p$ for some $p \leftrightarrow a$. On the other hand, the two a-derivatives of $a^2 \rhd a$, namely $a \rhd a$ and a^2, have depth 2, and thus neither of them is bisimilar to a. □

2.2 Equational Logic

An *axiom system* is a collection of equations $t \approx u$ over the language $\mathrm{BPA}_{\mathrm{int}}$. 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 $\mathrm{BPA}_{\mathrm{int}}$ contexts):

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

$$\frac{t \approx u \;\; t' \approx u'}{t + t' \approx u + u'} \quad \frac{t \approx u \;\; t' \approx u'}{tt' \approx uu'} \quad \frac{t \approx u \;\; t' \approx u'}{t \rhd t' \approx u \rhd u'} \;.$$

Without loss of generality one may assume that substitutions happen first in equational proofs, 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 applications of symmetry happen first in equational proofs. In the remainder of this paper, we shall tacitly assume that our equational axiom systems are closed with respect to symmetry.

It is well-known (see, e.g., Sect. 2 in [21]) 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 5. *An equation $t \approx u$ over the language BPA_{int} is sound with respect to $\underline{\leftrightarrow}$ iff $t \mathrel{\underline{\leftrightarrow}} u$. An axiom system is sound with respect to $\underline{\leftrightarrow}$ iff so is each of its equations.*

An example of a collection of equations over the language $\mathrm{BPA}_{\mathrm{int}}$ that are sound with respect to $\underline{\leftrightarrow}$ is given in Table 3. Those equations stem from [12]. Equations dealing with the interrupt operator using two auxiliary operators are offered in [7].

A1	$x + y \approx y + x$
A2	$(x + y) + z \approx x + (y + z)$
A3	$x + x \approx x$
A4	$(x + y)z \approx (xz) + (yz)$
A5	$(xy)z \approx x(yz)$

Table 3. Some Axioms for $\mathrm{BPA}_{\mathrm{int}}$

3 Bisimilarity is not Finitely Based over $\mathrm{BPA}_{\mathbf{int}}$

Our order of business in the remainder of this paper will be to show the following theorem:

Theorem 1. *Bisimilarity is not finitely based over the language* BPA_{int}*—that is, there is no finite axiom system that is sound with respect to* $\underline{\leftrightarrow}$*, and proves all of the equations* $t \approx u$ *such that* $t \underline{\leftrightarrow} u$*. Moreover, the same holds true if we restrict ourselves to the collection of closed equations over* BPA_{int} *that hold modulo* $\underline{\leftrightarrow}$*.*

The above theorem is an immediate corollary of the following result:

Theorem 2. *Let* E *be a finite collection of equations over the language* BPA_{int} *that is sound modulo* $\underline{\leftrightarrow}$*. Let* $n > 2$ *be larger than the size of each term in the equations in* E*. Then* E *does not prove the equation* e_n*, where the family of equations* e_n *(*$n \geq 1$*) is defined thus:*

$$e_n : \left(\sum_{i=1}^{n} p_i\right) \rhd a \approx b + \sum_{i=2}^{n} b((b^{i-1} + b) \rhd a) + a \sum_{i=1}^{n} p_i \ . \tag{1}$$

In the above family, $p_1 = b$ *and* $p_i = b(b^{i-1} + b)$ *for* $i > 1$*.*

Observe that, for each $n \geq 1$, the closed equation e_n is sound modulo bisimilarity. Indeed, the left-hand and right-hand sides of the equation have isomorphic labelled transitions systems. Therefore, as claimed above, Theorem 1 is an immediate consequence of Theorem 2. In the remainder of this study, we shall offer a proof of Theorem 2. In order to prove this theorem, it will be suffi cient to establish the following technical result:

Proposition 2. *Let* E *be a finite axiom system over the language* BPA_{int} *that is sound modulo bisimilarity. Let* $n > 2$ *be larger than the size of each term in the equations in* E*. Assume, furthermore, that*

- $E \vdash p \approx q$*,*
- $p \underline{\leftrightarrow} \left(\sum_{i=1}^{n} p_i\right) \rhd a$ *and*
- p *has a summand bisimilar to* $\left(\sum_{i=1}^{n} p_i\right) \rhd a$*.*

Then q *has a summand bisimilar to* $\left(\sum_{i=1}^{n} p_i\right) \rhd a$*.*

Indeed, assuming Proposition 2, we can prove Theorem 2, and therefore Theorem 1, as follows.

Proof of Theorem 2: Assume that E is a fi nite axiom system over the language BPA_{int} that is sound modulo bisimilarity. Pick $n > 2$ and larger than the size of the terms in the equations in E. Assume that, for some closed term q,

$$E \vdash \left(\sum_{i=1}^{n} p_i\right) \rhd a \approx q \ .$$

Using Proposition 2, we have that q has a summand bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$. Note now that the summands of the right-hand side of equation e_n, viz.

$$b + \sum_{i=2}^{n} b((b^{i-1} + b) \rhd a) + a \sum_{i=1}^{n} p_i \ ,$$

are the terms

– b,
– $b((b^{i-1}+b) \rhd a)$, for some $2 \leq i \leq n$, and
– $a \sum_{i=1}^{n} p_i$.

Unlike $\left(\sum_{i=1}^{n} p_i\right) \rhd a$, none of these terms can initially perform both an a and a b action. It follows that no summand of the right-hand side of equation e_n is bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$, and thus that

$$q \neq b + \sum_{i=2}^{n} b((b^{i-1}+b) \rhd a) + a \sum_{i=1}^{n} p_i \ .$$

We may therefore conclude that the axiom system E does not prove equation e_n, which was to be shown. □

Our order of business will now be to provide a proof of Proposition 2. Our proof of that result will be proof-theoretic in nature, and will proceed by induction on the depth of equational derivations from a finite axiom system E. The crux in such an induction proof is given by the following proposition, to the effect that the statement of Proposition 2 holds for closed instantiations of axioms in E.

Proposition 3. *Let $t \approx u$ be an equation over the language* BPA_{int} *that holds modulo bisimilarity. Let σ be a closed substitution, $p = \sigma(t)$ and $q = \sigma(u)$. Assume that*

– *$n > 2$ and the size of t is smaller than n,*
– *$p \leftrightarrow \left(\sum_{i=1}^{n} p_i\right) \rhd a$ and*
– *p has a summand bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$.*

Then q has a summand bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$.

Indeed, let us assume for the moment that the above result holds. Using it, we can prove Proposition 2 thus:

Proof of Proposition 2: Assume that E is a finite axiom system over the language BPA_{int} that is sound with respect to bisimulation equivalence, and that the following hold, for some closed terms p and q and positive integer $n > 2$ that is larger than the size of each term in the equations in E:

1. $E \vdash p \approx q$,
2. $p \leftrightarrow \left(\sum_{i=1}^{n} p_i\right) \rhd a$, and
3. p has a summand bisimilar to $(\sum_{i=1}^{n} p_i) \rhd a$.

We prove that q also has a summand bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$ by induction on the depth of the closed proof of the equation $p \approx q$ from E. Recall that, without loss of generality, we may assume that applications of symmetry happen first in equational proofs (that is, E is closed with respect to symmetry).

We proceed by a case analysis on the last rule used in the proof of $p \approx q$ from E. The case of reflexivity is trivial, and that of transitivity follows immediately by using the inductive hypothesis twice. Below we only consider the other possibilities.

– CASE $E \vdash p \approx q$, BECAUSE $\sigma(t) = p$ AND $\sigma(u) = q$ FOR SOME EQUATION $(t \approx u) \in E$ AND CLOSED SUBSTITUTION σ. Since $n > 2$ is larger than the size of each term mentioned in equations in E, the claim follows by Proposition 3.
– CASE $E \vdash p \approx q$, BECAUSE $p = p' + p''$ AND $q = q' + q''$ FOR SOME p', q', p'', q'' SUCH THAT $E \vdash p' \approx q'$ AND $E \vdash p'' \approx q''$. Since p has a summand bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$, we have that so does either p' or p''. Assume, without loss of generality, that p' has a summand bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$. Since p is bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$, so is p'. The inductive hypothesis now yields that q' has a summand bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$. Hence, q has a summand bisimilar to $\left(\sum_{i=1}^{n} p_i\right) \rhd a$, which was to be shown.
– CASE $E \vdash p \approx q$, BECAUSE $p = p'p''$ AND $q = q'q''$ FOR SOME p', q', p'', q'' SUCH THAT $E \vdash p' \approx q'$ AND $E \vdash p'' \approx q''$. This case is vacuous. In fact, $norm(p) = 1$ by our assumption that $p \leftrightarrows \left(\sum_{i=1}^{n} p_i\right) \rhd a$, whereas the norm of a closed term of the form $p'p''$ is at least 2.
– CASE $E \vdash p \approx q$, BECAUSE $p = p' \rhd p''$ AND $q = q' \rhd q''$ FOR SOME p', q', p'', q'' SUCH THAT $E \vdash p' \approx q'$ AND $E \vdash p'' \approx q''$. The claim is immediate because p and q are their only summands, and E is sound modulo bisimilarity.

This completes the proof. □

In light of our previous discussion, all that we are left to do to complete our proof of Theorem 1 is to show Proposition 3. The rather lengthy proof of that result may be found in [6].

4 BPA with the Disrupt Operator

As mentioned in Sect. 1, in their paper [7], Baeten and Bergstra have given a finite axiomatization of bisimilarity over BPA_δ (the extension of BPA with a constant δ to describe "deadlock") enriched with two mode transfer operators, viz. the *disrupt* and *interrupt* operators, using auxiliary operators. The main result in this paper (Theorem 1) shows that the use of auxiliary operators is indeed necessary in order to obtain a finite axiomatization of bisimulation equivalence over the language $\mathrm{BPA}_{\mathrm{int}}$, and that this holds true even if we restrict ourselves to axiomatizing the collection of closed equations over this language.

A natural question to ask at this point is whether this negative result applies also to the language $\mathrm{BPA}_{\mathrm{dis}}$ obtained by enriching BPA with the disrupt operator. Intuitively, "p disrupted by q"—which we shall write $p \blacktriangleright q$ in what follows—describes a process that normally behaves like p. However, at each point of the computation before p terminates, q can begin its execution. If this happens, q takes over, and p never resumes its computation. This intuition is captured formally by the following transition rules:

$$\frac{t \xrightarrow{a} \surd}{t \blacktriangleright u \xrightarrow{a} \surd} \qquad \frac{t \xrightarrow{a} t'}{t \blacktriangleright u \xrightarrow{a} t' \blacktriangleright u} \qquad \frac{u \xrightarrow{a} \surd}{t \blacktriangleright u \xrightarrow{a} \surd} \qquad \frac{u \xrightarrow{a} u'}{t \blacktriangleright u \xrightarrow{a} u'}$$

It is not hard to see that the following equations are sound modulo bisimilarity over the language $\mathrm{BPA}_{\mathrm{dis}}$:

$$\begin{aligned} a \blacktriangleright x &\approx a + x \\ ax \blacktriangleright y &\approx a(x \blacktriangleright y) + y \quad \text{and} \\ (x + y) \blacktriangleright z &\approx (x \blacktriangleright z) + (y \blacktriangleright z) \ . \end{aligned}$$

The last of these equations is particularly important, at least as far as obtaining a finite equational axiomatization of bisimilarity over the collection of closed terms in the language $\mathrm{BPA}_{\mathrm{dis}}$ is concerned. (The interested reader may have already noticed that its soundness modulo bisimulation equivalence depends crucially on the fact that transitions due to moves of the second argument of a disrupt discard the first argument.) Indeed, its repeated use in conjunction with the first two laws allows us to eliminate occurrences of the disrupt operator from closed terms. This effectively reduces the problem of finitely axiomatizing bisimilarity over the collection of closed terms in the language $\mathrm{BPA}_{\mathrm{dis}}$ to that of offering a finite axiomatization of bisimilarity over closed BPA terms. As shown by Bergstra and Klop in [12], the five equations in Table 3 suffice to axiomatize bisimilarity over the language BPA.

In sharp contrast to Theorem 1, we therefore have that:

Theorem 3. *The collection of closed equations over* BPA_{dis} *that hold modulo* $\underline{\leftrightarrow}$ *is axiomatized by the five equations in Table 3 together with the aforementioned three equations for the disrupt operator, and is therefore finitely based.*

It follows that the use of auxiliary operators is *not* necessary in order to obtain a finite axiomatization of bisimulation equivalence over closed terms in the language $\mathrm{BPA}_{\mathrm{dis}}$.

The axiomatization of bisimilarity over closed terms in the language $\mathrm{BPA}_{\mathrm{dis}}$ offered in the theorem above is not ω-complete. For example, the reader can easily check that the disrupt operator is associative modulo bisimilarity, i.e., that the equation

$$(x \blacktriangleright y) \blacktriangleright z \approx x \blacktriangleright (y \blacktriangleright z)$$

holds modulo $\underline{\leftrightarrow}$. This equation is not provable using the equations mentioned in Theorem 3. However, we conjecture that bisimilarity also affords a finite ω-complete axiomatization over $\mathrm{BPA}_{\mathrm{dis}}$.

References

1. L. ACETO, Z. ÉSIK, AND A. INGOLFSDOTTIR, *On the two-variable fragment of the equational theory of the max-sum algebra of the natural numbers*, in Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science, STACS 2000 (Lille), H. Reichel and S. Tison, eds., vol. 1770 of Lecture Notes in Computer Science, Springer-Verlag, Feb. 2000, pp. 267–278.
2. ———, *The max-plus algebra of the natural numbers has no finite equational basis*, Theoretical Comput. Sci., 293 (2003), pp. 169–188.
3. L. ACETO, W. FOKKINK, R. VAN GLABBEEK, AND A. INGOLFSDOTTIR, *Nested semantics over finite trees are equationally hard*, Information and Computation, 191 (2004), pp. 203–232.

4. L. ACETO, W. FOKKINK, AND A. INGOLFSDOTTIR, *A menagerie of non-finitely based process semantics over BPA*—from ready simulation to completed traces*, Mathematical Structures in Computer Science, 8 (1998), pp. 193–230.
5. L. ACETO, W. FOKKINK, A. INGOLFSDOTTIR, AND B. LUTTIK, *CCS with Hennessy's merge has no finite equational axiomatization*, Theoretical Comput. Sci., 330 (2005), pp. 377–405.
6. L. ACETO, W. FOKKINK, A. INGOLFSDOTTIR, AND S. NAIN, *Bisimilarity is not Finitely Based over BPA with Interrupt*, Research report RS-04-24, BRICS, Oct. 2004. Available from `http://www.brics.dk/RS/04/24/index.html`.
7. J. C. BAETEN AND J. BERGSTRA, *Mode transfer in process algebra*, Report CSR 00–01, Technische Universiteit Eindhoven, 2000. This paper is an expanded and revised version of [11].
8. J. C. BAETEN, J. BERGSTRA, AND J. W. KLOP, *Syntax and defining equations for an interrupt mechanism in process algebra*, Fundamenta Informaticae, IX (1986), pp. 127–168.
9. ———, *Decidability of bisimulation equivalence for processes generating context-free languages*, J. Assoc. Comput. Mach., 40 (1993), pp. 653–682.
10. J. C. BAETEN AND C. VERHOEF, *A congruence theorem for structured operational semantics*, in Proceedings CONCUR 93, Hildesheim, Germany, E. Best, ed., vol. 715 of Lecture Notes in Computer Science, Springer-Verlag, 1993, pp. 477–492.
11. J. BERGSTRA, *A mode transfer operator in process algebra*, Report P8808, Programming Research Group, University of Amsterdam, 1988.
12. J. BERGSTRA AND J. W. KLOP, *Fixed point semantics in process algebras*, Report IW 206, Mathematisch Centrum, Amsterdam, 1982.
13. ———, *Process algebra for synchronous communication*, Information and Control, 60 (1984), pp. 109–137.
14. S. BLOM, W. FOKKINK, AND S. NAIN, *On the axiomatizability of ready traces, ready simulation and failure traces*, in Proceedings 30th Colloquium on Automata, Languages and Programming—ICALP'03, Eindhoven, J. C. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, eds., vol. 2719 of Lecture Notes in Computer Science, Springer-Verlag, 2003, pp. 109–118.
15. E. BRINKSMA, *A tutorial on LOTOS*, in Proceedings of the IFIP Workshop on Protocol Specification, Testing and Verification, M. Diaz, ed., North-Holland, 1986, pp. 73–84.
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. A. DSOUZA AND B. BLOOM, *On the expressive power of CCS*, in Foundations of Software Technology and Theoretical Computer Science (Bangalore, 1995), P. S. Thiagarajan, ed., vol. 1026 of Lecture Notes in Computer Science, Springer-Verlag, 1995, pp. 309–323.
18. W. FOKKINK AND B. LUTTIK, *An omega-complete equational specification of interleaving*, in Proceedings 27th Colloquium on Automata, Languages and Programming—ICALP'00, Geneva, U. Montanari, J. Rolinn, and E. Welzl, eds., vol. 1853 of Lecture Notes in Computer Science, Springer-Verlag, July 2000, pp. 729–743.
19. W. FOKKINK AND S. NAIN, *On finite alphabets and infinite bases: From ready pairs to possible worlds*, in Proceedings of Foundations of Software Science and Computation Structures, 7th International Conference, FOSSACS 2004, I. Walukiewicz, ed., vol. 2897, Springer-Verlag, 2004, pp. 182–194.
20. J. L. GISCHER, *The equational theory of pomsets*, Theoretical Comput. Sci., 61 (1988), pp. 199–224.
21. J. F. GROOTE, *A new strategy for proving ω–completeness with applications in process algebra*, in Proceedings CONCUR 90, Amsterdam, J. C. Baeten and J. W. Klop, eds., vol. 458 of Lecture Notes in Computer Science, Springer-Verlag, 1990, pp. 314–331.

22. M. HENNESSY, *Axiomatising finite concurrent processes*, SIAM J. Comput., 17 (1988), pp. 997–1017.
23. ISO, *Information processing systems – open systems interconnection – LOTOS – a formal description technique based on the temporal ordering of observational behaviour* ISO/TC97/SC21/N DIS8807, 1987.
24. S. MAUW, *PSF – A Process Specification Formalism*, PhD thesis, University of Amsterdam, Dec. 1991.
25. R. MILNER, *Communication and Concurrency*, Prentice-Hall International, Englewood Cliffs, 1989.
26. R. MILNER, M. TOFTE, R. HARPER, AND D. MACQUEEN, *The Definition of Standard ML (Revised)*, MIT Press, 1997.
27. F. MOLLER, *The importance of the left merge operator in process algebras*, in Proceedings 17^{th} ICALP, Warwick, M. Paterson, ed., vol. 443 of Lecture Notes in Computer Science, Springer-Verlag, July 1990, pp. 752–764.
28. ———, *The nonexistence of finite axiomatisations for CCS congruences*, in Proceedings 5^{th} Annual Symposium on Logic in Computer Science, Philadelphia, USA, IEEE Computer Society Press, 1990, pp. 142–153.
29. D. PARK, *Concurrency and automata on infinite sequences*, in 5^{th} GI Conference, Karlsruhe, Germany, P. Deussen, ed., vol. 104 of Lecture Notes in Computer Science, Springer-Verlag, 1981, pp. 167–183.
30. V. REDKO, *On defining relations for the algebra of regular events*, Ukrainskii Matematicheskii Zhurnal, 16 (1964), pp. 120–126. In Russian.
31. P. SEWELL, *Nonaxiomatisability of equivalences over finite state processes*, Annals of Pure and Applied Logic, 90 (1997), pp. 163–191.