Axiomatizations for the Perpetual Loop in Process Algebra

Wan Fokkink

University of Wales Swansea
Department of Computer Science
Singleton Park, Swansea SA2 8PP, Wales
e-mail: w.j.fokkink@swan.ac.uk

Abstract. Milner proposed an axiomatization for the Kleene star in basic process algebra, in the presence of deadlock and empty process, modulo bisimulation equivalence. In this paper, Milner's axioms are adapted to no-exit iteration x^ω, which executes x infinitely many times in a row, and it is shown that this axiomatization is complete for no-exit iteration in basic process algebra with deadlock and empty process, modulo bisimulation.

1 Introduction

Kleene [15] defined a binary operator x^*y in the context of finite automata, which denotes the iterate of x on y. Intuitively, the expression x^*y can choose to execute either x, after which it evolves into x^*y again, or y, after which it terminates. A feature of the Kleene star is that on the one hand it can express recursion, while on the other hand it can be captured in equational laws. Hence, one does not need meta-principles such as the Recursive Specification Principle [10]. Kleene formulated several equations for his operator, notably the defining equation $x^*y = x(x^*y) + y$. In later years it became more fashionable to consider the unary version x^* of the Kleene star. In the presence of the empty process, the unary and the binary Kleene star are equally expressive.

Salomaa [22] presented a finite complete axiomatization for the Kleene star in language theory, modulo completed trace equivalence, which incorporates one conditional axiom, namely, if $x = y \cdot x + z$, and y cannot terminate immediately, then $x = y^*z$. Salomaa's completeness proof basically consists of two steps: first he shows that the solutions of a guarded recursive specification are all provably equal to the same term, and next he shows that if two terms are completed trace equivalent, then there exists a guarded recursive specification for which both terms are solutions.

Milner [17] was the first to study the (unary) Kleene star modulo bisimulation, and proposed an axiomatization for it, being an adaptation of Salomaa's axiom system. Milner [17, page 461] raised the question whether his axiomatization is complete for the Kleene star in process theory, and remarked that this question may be hard to answer: "The difficulty is that the method [...] of Salomaa's original completeness proof cannot be applied directly, since -in contrast with the case of languages- an arbitrary system of guarded equations [...] cannot in general be solved in star expressions".

In this paper the instantiation $x^*\delta$ of the binary Kleene star is studied, which carries two names: perpetual loop and no-exit iteration. Since the deadlock δ blocks the exits, this construct executes x an infinite number of times in a row. The perpetual loop is closely related to the Kleene star, and shares several of its characteristics. In this paper no-exit iteration, which is denoted by x^ω, is studied in Basic Process Algebra [9] with deadlock and empty process, denoted by $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$. No-exit iteration can be used to formally describe programs that repeat a certain procedure without end. A significant

advantage of iteration over recursion as a means to express infinite processes is that it does not involve a parametric process definition, because the development of process theory is easier if parametrization does not have to be taken as primitive (see e.g. Milner [18, page 212]). For example, iteration allows simpler axiomatizations than recursion, and it does not need a guardedness restriction to locate the class of meaningful terms. Therefore, the Kleene star is used for example in the specification and verification of Grid protocols [7], which describe parallel computations in a grid-like architecture, and in the ToolBus [8], which enables to link separate tools. In both cases, iteration is used almost exclusively in the form of the perpetual loop. No-exit iteration is also used in the educational vein [21], because it enables to specify and verify infinite processes in a simple and intuitive way.

The three axioms for the unary Kleene star in Milner's axiom system (being Kleene's defining equation, Salomaa's conditional axiom and an equation which describes the interplay of Kleene star and empty process) have obvious counterparts for no-exit iteration. It turns out that these three axioms, together with the standard axioms for $\mathrm{BPA}^{\omega}_{\delta\varepsilon}(A)$, make a complete axiomatization for $\mathrm{BPA}^{\omega}_{\delta\varepsilon}(A)$ modulo bisimulation. The completeness proof is based on a strategy that originates from [11]. It also uses new techniques, which will hopefully turn out to be applicable in a possible proof of Milner's conjecture (see Section 4 for a discussion on this topic). For a detailed presentation of the completeness proof for $\mathrm{BPA}^{\omega}_{\delta\varepsilon}(A)$, and for omitted proofs in this paper, the reader is referred to [12].

This paper focuses on the process algebra $\mathrm{BPA}^{\omega}_{\delta}(A)$, in which the empty process is not present. This setting allows a more concise presentation of the ideas that are used in the completeness proof for the perpetual loop in process algebra. We will see that Kleene's defining equation and Salomaa's conditional axiom for the perpetual loop, together with the standard axioms for $\mathrm{BPA}_{\delta}(A)$, are complete for $\mathrm{BPA}^{\omega}_{\delta}(A)$ modulo bisimulation.

Sewell [23] proved that there does not exist a complete finite equational axiomatization for the Kleene star in combination with deadlock modulo bisimulation, due to the fact that a^{ω} is bisimilar to $(a^n)^{\omega}$ for $n = 1, 2, ...$ Since these equivalences are also present in $\mathrm{BPA}^{\omega}_{\delta\varepsilon}$, Sewell's argument can be copied to conclude that there does not exist a complete finite equational axiomatization for $\mathrm{BPA}^{\omega}(A)$. Hence, the adaptation of Salomaa's conditional axiom for the perpetual loop is essential for the obtained completeness results.

The requirement 'y cannot terminate immediately' in Salomaa's conditional axiom can be defined inductively on the syntax. According to Kozen [16] this requirement is not algebraic, in the sense that it is not preserved under substitution of terms for actions. He proposed two alternative conditional axioms which do not have this drawback. These axioms, however, are not sound with respect to bisimulation equivalence.

Bergstra, Bethke and Ponse [6] suggested a finite equational axiomatization for BPA^{*}, i.e, for basic process algebra with the binary Kleene star without the special constants δ and ϵ, modulo bisimulation. Their conjecture that it is complete was solved by Fokkink and Zantema [14]. (In contrast with this result, Aceto, Fokkink and Ingólfsdóttir [3] showed that there does not exist a complete finite equational axiomatization for BPA^{*} modulo any process semantics in between ready simulation and completed traces.) In [11], a new proof for the completeness result from [14] was presented. This new proof technique was applied successfully not only in this paper, but also in a paper on a restricted version of iteration called prefix iteration, which is better suited for a setting with prefix multiplication or with communication [2], and in a paper on a more expressive variant of iteration called multi-exit iteration [1].

Acknowledgements. This research was initiated by a question from Alban Ponse. Luca Aceto, Jaco van de Pol, Alban Ponse and an anonymous referee provided useful comments, and Jan Bergstra is thanked for stimulating discussions.

2 The Perpetual Loop in Process Algebra

2.1 Syntax

We assume a non-empty alphabet A of atomic actions, with typical elements a, b, c. We also assume two special constants δ, which represents deadlock, and ε, which represents the empty process, and ξ ranges over $A \cup \{\delta, \varepsilon\}$. Furthermore, we have two binary operators: alternative composition $x + y$, which combines the behaviours of x and y, and sequential composition $x \cdot y$, which puts the behaviours of x and y in sequence. Finally, we have the unary operator x^ω, which executes x infinitely many times in a row. We will refer to this operator both as *perpetual loop* and as *no-exit iteration*. The language $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$, with typical elements $p, q, ..., w$, consists of all the terms that can be constructed from the atomic actions, the two special constants, the two binary composition operators, and the perpetual loop. That is, the BNF grammar for the collection of process terms is:

$$p \quad ::= \quad a \mid \delta \mid \varepsilon \mid p + p \mid p \cdot p \mid p^\omega.$$

$\mathrm{BPA}^\omega_\delta(A)$ is obtained by deleting the empty process ε, and $\mathrm{BPA}^\omega(A)$ is obtained by deleting the deadlock δ and the empty process ε from the syntax. The sequential composition operator will often be omitted, so pq denotes $p \cdot q$. As binding convention, alternative composition binds weaker than sequential composition and no-exit iteration.

Remark: The presence of the special constant δ in $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$ is redundant, because it can be expressed in $\mathrm{BPA}^\omega_\varepsilon(A)$ modulo bisimulation: ε^ω is bisimilar with δ, because both processes do not exhibit any behaviour. However, δ is maintained in the syntax as a standard abbreviation.

2.2 Operational Semantics

Table 1 presents an operational semantics for $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$ in Plotkin style [20], where $x \xrightarrow{a} x'$ represents that process x can evolve into process x' by the execution of action a, and $x \xrightarrow{a} \surd$ denotes that process x can terminate by the execution of action a, and the unary predicate $x \xrightarrow{\varepsilon} \surd$ denotes that process x can terminate immediately.

$$\varepsilon \xrightarrow{\varepsilon} \surd \qquad a \xrightarrow{a} \surd$$

$$\frac{x \xrightarrow{\xi} \surd}{x + y \xrightarrow{\xi} \surd \quad y + x \xrightarrow{\xi} \surd} \qquad \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x' \quad y + x \xrightarrow{a} x'}$$

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

$$\frac{x \xrightarrow{a} \surd}{x^\omega \xrightarrow{a} x^\omega} \qquad \frac{x \xrightarrow{a} x'}{x^\omega \xrightarrow{a} x' \cdot (x^\omega)}$$

Table 1. Transition rules for $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$

Definition 1. p' is a *derivative* of p if p can evolve into p' by zero or more transitions.
p' is a *proper* derivative of p if p can evolve into p' by one or more transitions.

Note that a process term can be a proper derivative of itself, for example, $a^*b \xrightarrow{a} a^*b$. In the sequel, p' and p'' will denote derivatives of process term p. The following lemma can easily be deduced, using structural induction.

Lemma 2. *Each process term in* $\mathrm{BPA}^{\omega}_{\delta\varepsilon}(A)$ *has only finitely many derivatives.*

Process terms are considered modulo bisimulation equivalence from Park [19]. Intuitively, two processes are bisimilar if they have the same branching structure.

Definition 3. Two processes p and q are *bisimilar*, denoted by $p \leftrightarrows q$, if there exists a symmetric binary relation B on processes which relates p and q, such that:

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

Bisimulation equivalence is a congruence with respect to all the operators, which means that if $p \leftrightarrows p'$ and $q \leftrightarrows q'$, then $p+q \leftrightarrows p'+q'$ and $pq \leftrightarrows p'q'$ and $p^{\omega} \leftrightarrows (p')^{\omega}$. Namely, the transition rules in Table 1 are in the 'path' format, which guarantees that the generated bisimulation equivalence is a congruence, see [5, 13].

2.3 Axiomatizations

Table 2 presents the standard axioms A1-9 for $\mathrm{BPA}_{\delta\varepsilon}(A)$. Furthermore, Table 3 contains the defining equation NEI1 together with the conditional axiom RSP^{ω} for the perpetual loop. The axiomatization A1-7+NEI1+RSP^{ω} is sound for $\mathrm{BPA}^{\omega}_{\delta}(A)$, i.e., if $p = q$ in $\mathrm{BPA}^{\omega}_{\delta}(A)$ is provable from these axioms, then $p \leftrightarrows q$. Since bisimulation equivalence is a congruence for $\mathrm{BPA}^{\omega}_{\delta}(A)$, soundness can be verified by checking this property for each axiom separately, which is left to the reader.

A1	$x + y = y + x$
A2	$(x + y) + z = x + (y + z)$
A3	$x + x = x$
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$
A6	$x + \delta = x$
A7	$\delta \cdot x = \delta$
A8	$x \cdot \varepsilon = x$
A9	$\varepsilon \cdot x = x$

Table 2. The axioms for $\mathrm{BPA}_{\delta\varepsilon}(A)$

NEI1	$x \cdot (x^{\omega}) = x^{\omega}$
RSP^{ω}	$x = y \cdot x \implies x = y^{\omega}$

Table 3. The axioms for the perpetual loop in the absence of ε

However, the axiom RSP^ω is not sound in the presence of the empty process. Namely, due to the axiom A9, $x = \varepsilon x$, it then implies $x = \varepsilon^\omega$, which is clearly unsound. Therefore, in Table 4 an adaptation $\mathrm{RSP}^\omega_\varepsilon$ is introduced, where the condition $y \not\downarrow$ expresses that y cannot terminate immediately. This condition, which is similar to the so-called guardedness restriction in the Recursive Specification Principle from Bergstra and Klop [10], can be defined inductively on the syntax:

$$\begin{array}{c} a \not\downarrow \\ \delta \not\downarrow \\ x \not\downarrow \wedge y \not\downarrow \Rightarrow (x+y) \not\downarrow \\ x \not\downarrow \vee y \not\downarrow \Rightarrow (x \cdot y) \not\downarrow \\ (x^\omega) \not\downarrow \end{array}$$

Table 4 contains the defining equation NEI1, and an extra equation NEI2 which describes the interplay of no-exit iteration with the empty process. The axiomatization A1-9+NEI1,2+$\mathrm{RSP}^\omega_\varepsilon$ is sound for $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$.

$$\begin{array}{ll} \text{NEI1} & x \cdot (x^\omega) = x^\omega \\ \text{NEI2} & (x+\varepsilon)^\omega = x^\omega \\ \mathrm{RSP}^\omega_\varepsilon & x = y \cdot x \wedge y \not\downarrow \implies x = y^\omega \end{array}$$

Table 4. The axioms for the perpetual loop in the presence of ε

The purpose of this paper is to present the following three completeness results.

Theorem 4. *The axiomatization* A1-9+NEI1,2+$\mathrm{RSP}^\omega_\varepsilon$ *is complete for* $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$ *with respect to bisimulation.*

That is, if $p \leftrightarrows q$ for process terms p and q in $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$, then $p = q$ can be derived from the axioms A1-9+NEI1,2+$\mathrm{RSP}^\omega_\varepsilon$.

Theorem 5. *The axiomatization* A1-7+NEI1+RSP^ω *is complete for* $\mathrm{BPA}^\omega_\delta(A)$ *with respect to bisimulation.*

Theorem 6. *The axiomatization* A1-5+NEI1+RSP^ω *is complete for* $\mathrm{BPA}^\omega(A)$ *with respect to bisimulation.*

This paper focuses on the completeness proof for $\mathrm{BPA}^\omega_\delta(A)$. The completeness proof for $\mathrm{BPA}^\omega(A)$ is closely related to the one for $\mathrm{BPA}^\omega_\delta(A)$ (missing only some minor cases for δ in the construction of basic terms in Lemma 17). The completeness proof for $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$ also uses the same proof strategy, but, due to the presence of the empty process, the technical details are considerably more complicated. The reader is referred to [12] for a detailed exposition on the completeness proof for $\mathrm{BPA}^\omega_{\delta\varepsilon}(A)$.

3 Proof of the Main Theorem

This section presents preliminaries that are needed in the proof of Theorem 5, together with the completeness proof itself. Many preliminary definitions in this section originate from [11]. For omitted proofs the reader is referred to [12].

3.1 Expansions

From now on, process terms in $\mathrm{BPA}^{\omega}_{\delta}(A)$ are considered modulo associativity and commutativity of the $+$, that is, modulo the axioms A1,2. We write $p =_{\mathrm{AC}} q$ if p and q can be equated by axioms A1,2. As usual, $\sum_{i=1}^{n} p_i$ represents the term $p_1 + \ldots + p_n$, and the p_i are called the summands of this term. The empty sum represents δ, where $\sum_{i\in\emptyset} p_i + q$ is not considered empty.

Definition 7. For each process term p, its collection of possible transitions is finite, say $\{p \xrightarrow{a_i} p_i \mid i = 1, ..., n\} \cup \{p \xrightarrow{b_j} \surd \mid j = 1, ..., m\}$. The *expansion* of p is

$$\sum_{i=1}^{n} a_i p_i + \sum_{j=1}^{m} b_j.$$

Lemma 8. *Each process term p in* $\mathrm{BPA}^{\omega}_{\delta}(A)$ *is provably equal to its expansion, using* A4-7+NEI1.

Proof: By structural induction with respect to p.

3.2 Normed Processes

The following terminology stems from [4].

Definition 9. A process term p is called *normed* if it can terminate in finitely many transitions, that is, $p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} p_n \xrightarrow{b} \surd$.

The class of normed processes in $\mathrm{BPA}^{\omega}_{\delta}(A)$ can be defined inductively as follows:

- $a \in A$ is normed;
- if p or q is normed, then $p + q$ is normed;
- if p and q are normed, then pq is normed.

Lemma 10. *If p is not normed, then $pq = p$ is provable using* A4,5,7+NEI1+RSP^{ω}.

Proof: By structural induction with respect to p.

3.3 An Ordering on Pairs of Terms

The following weight function on process terms in $\mathrm{BPA}^{\omega}_{\delta}(A)$, which represents the maximum nesting of ω's in a term, will be used to formulate an ordering on pairs of terms.

$$\begin{aligned} g(a) &= 0 \\ g(\delta) &= 0 \\ g(p+q) &= \max\{g(p), g(q)\} \\ g(pq) &= \max\{g(p), g(q)\} \\ g(p^{\omega}) &= g(p) + 1. \end{aligned}$$

Note that g-value is invariant under axioms A1,2. The following lemma can easily be deduced, using structural induction.

Lemma 11. *If p' is a derivative of p, then $g(p') \leq g(p)$.*

We consider pairs of process terms modulo commutativity. The ordering $<$ on pairs of process terms is defined as follows.

Definition 12. The ordering $<$ on pairs of terms is obtained by taking the transitive closure of the union of the three relations below.

1. $(r, s) < (p, q)$ if $g(r) < g(p)$ and $g(s) < g(p)$;
2. $(r, s) < (p, q)$ if $g(r) < g(p)$ and $g(s) \leq g(q)$;
3. $(p', q') < (p, q)$ if p' is a derivative of p, and not vice versa, and q' is a derivative of q.

The proof of the completeness theorem is based on induction with respect to this ordering, so we need to know that it is well-founded.

Lemma 13. *The ordering $<$ on pairs of process terms is well-founded modulo $=_{\mathrm{AC}}$.*

Proof: Omitted.

3.4 Basic Terms

We construct a set $\mathbb{B}$ of *basic* process terms, such that each process term is provably equal to a basic term, and the derivatives of basic terms are basic terms. We will prove the completeness theorem by showing that bisimilar basic terms are provably equal.

Definition 14. The set $\mathbb{B}$ of *basic* process terms is defined inductively as follows:

1. if $a_1, ..., a_n, b_1, ..., b_m \in A$ and $p_1, ..., p_n \in \mathbb{B}$, then $\sum_{i=1}^{n} a_i p_i + \sum_{j=1}^{m} b_j \in \mathbb{B}$;
2. if $p \in \mathbb{B}$ then $p^\omega \in \mathbb{B}$;
3. if $p \in \mathbb{B}$ and p' is a proper derivative of p, then $p'(p^\omega) \in \mathbb{B}$.

For notational convenience, we distinguish the following set $\mathbb{C}$ of cycles in $\mathbb{B}$.

Definition 15. $\mathbb{C} = \{p^\omega, p'(p^\omega) \mid p \in \mathbb{B},\ p' \text{ proper derivative of } p\}$.

The following facts for basic terms will be needed in the completeness proof.

Lemma 16. *1. If $p \in \mathbb{C}$ and $p \xrightarrow{a} p'$, then $p' \in \mathbb{C}$.*
2. If $p \in \mathbb{B}$ and $p \xrightarrow{a} p'$, then $p' \in \mathbb{B}$.
3. If $p \in \mathbb{B}$ and p is a proper derivative of itself, then $p \in \mathbb{C}$.

Lemma 17. *For each term p there exists a basic term q with $g(q) \leq g(p)$ such that $p = q$ is provable using* A4-7+NEI1+RSP$^\omega$.

3.5 The Auxiliary Function ϕ

Before starting with the completeness proof, first we need to develop some theory. The proposition that will be proved at the end of this section makes an important stepping stone to obtain the desired completeness result for $\mathrm{BPA}^\omega_\delta(A)$.

$p'(p^\omega) \leftrightarrows p''(p^\omega)$, with p' and p'' derivatives of p, does not imply $p' \leftrightarrows p''$. For example, clearly $a((aa)^\omega) \leftrightarrows aa((aa)^\omega)$, but $a \not\leftrightarrows aa$. In order to solve this ambiguity, we define an operator ϕ_p on basic terms, where intuitively the term $\phi_p(q)$, for $q \notin \mathbb{C}$, is obtained from the argument q as follows: all proper derivatives q' of q with $q'(p^\omega) \leftrightarrows p^\omega$ are removed in $\phi_p(q)$. We will see that if $p'(p^\omega) \leftrightarrows p''(p^\omega)$ then $\phi_p(p') \leftrightarrows \phi_p(p'')$.

Definition 18. Given $q \in \mathbb{B}$, the term $\phi_p(q)$ is defined as follows, using structural induction. We distinguish two cases: either $q \in \mathbb{C}$ or $q \notin \mathbb{C}$.

– Case 1: $q \in \mathbb{C}$. Then put

$$\phi_p(q) =_{\mathrm{AC}} q.$$

– Case 2: $q \notin \mathbb{C}$, so that

$$q =_{\mathrm{AC}} \sum_{i\in I} a_i q_i + \sum_{j\in J} b_j.$$

Then define $I_0 = \{i \in I \mid q_i(p^\omega) \not\leftrightarrow p^\omega\}$, and put

$$\phi_p(q) =_{\mathrm{AC}} \sum_{i\in I_0} a_i \phi_p(q_i) + \sum_{i\in I\setminus I_0} a_i + \sum_{j\in J} b_j.$$

Lemma 19. *For $q \in \mathbb{B}$ we have $g(\phi_p(q)) \leq g(q)$.*

Proof: By structural induction with respect to q.

The proofs of the next two technical lemmas are quite involved, and therefore omitted.

Lemma 20. *Assume that for some natural number N_0:*

A. for all terms u with $g(u) < N_0$ we have $p^\omega \not\leftrightarrow u$.

Let $q, r \in \mathbb{B}$ and $g(q + r) < N_0$. If $q(p^\omega) \leftrightarrow r(p^\omega)$ then

$$\phi_p(q) \leftrightarrow \phi_p(r).$$

Proof: Omitted.

Lemma 21. *Assume that for some natural number N_0:*

A. for all terms u with $g(u) < N_0$ we have $p \not\leftrightarrow u$;
B. for all pairs (u, v) of bisimilar terms with $g(u + v) < N_0$ we have $u = v$.

Let $p, q \in \mathbb{B}$ and $g(p + q) < N_0$. Then

$$q(\phi_p(p)^\omega) = \phi_p(q)(\phi_p(p)^\omega).$$

Proof: Omitted.

Proposition 22. *Assume that for some natural number N_0:*

A. for all terms u with $g(u) < N_0$ we have $p^\omega \not\leftrightarrow u$;
B. for all pairs (u, v) of bisimilar terms with $g(u + v) < N_0$ we have $u = v$.

Let $g(p + q + r) < N_0$ and $q(p^\omega) \leftrightarrow r(p^\omega)$. Then

$$q(p^\omega) = r(p^\omega).$$

Proof: By Lemma 17

$$p = s \tag{1}$$

with $s \in \mathbb{B}$ and $g(s) \leq g(p) < N_0$. Since conditions A and B hold, Lemma 21 can be applied to derive $s(\phi_s(s)^\omega) = \phi_s(s)(\phi_s(s)^\omega) = \phi_s(s)^\omega$. RSP$^\omega$ then yields

$$s^\omega = \phi_s(s)^\omega. \tag{2}$$

According to Lemma 17 there exist basic terms t and u with $g(t) \leq g(q) < N_0$ and $g(u) \leq g(r) < N_0$ and

$$q = t \tag{3}$$
$$r = u. \tag{4}$$

Since $t(s^\omega) \leftrightarrow q(p^\omega) \leftrightarrow r(p^\omega) \leftrightarrow u(s^\omega)$, and since $g(t + u) < N_0$ and requirement A of Lemma 20 is satisfied, it implies $\phi_s(t) \leftrightarrow \phi_s(u)$. Since $g(\phi_s(t) + \phi_s(u)) < N_0$ (Lemma 19), condition B yields

$$\phi_s(t) = \phi_s(u). \tag{5}$$

Hence,

$$\begin{aligned} q(p^\omega) &\overset{(1)(3)}{=} t(s^\omega) \overset{(2)}{=} t(\phi_s(s)^\omega) \overset{\text{Lem. } 21}{=} \phi_s(t)(\phi_s(s)^\omega) \\ &\overset{(5)}{=} \phi_s(u)(\phi_s(s)^\omega) \overset{\text{Lem. } 21}{=} u(\phi_s(s)^\omega) \overset{(2)}{=} u(s^\omega) \overset{(1)(4)}{=} r(p^\omega). \end{aligned}$$

□

3.6 Completeness Proof

Proof of Theorem 5: Assume $p, q \in \mathbb{B}$ with $p \leftrightarrows q$; we show that $p = q$ can be derived from A1-7+NEI1+RSP$^\omega$, by induction on the well-founded ordering $<$ on pairs of terms. So suppose that we have already dealt with pairs of bisimilar basic terms that are smaller than (p, q). By symmetry it is sufficient to consider two cases: either $p \notin \mathbb{C}$ or $p, q \in \mathbb{C}$.

- Case 1: $p \notin \mathbb{C}$.
  According to Lemma 8 p and q are provably equal to their expansions. Since $p \leftrightarrows q$, these expansions can be adapted, using axiom A3, to obtain:

$$p = \sum_{i=1}^{n} a_i p_i + \sum_{j=1}^{m} b_j, \qquad q = \sum_{i=1}^{n} a_i q_i + \sum_{j=1}^{m} b_j,$$

  where $p_i \leftrightarrows q_i$ for $i = 1, ..., n$. Since $p \notin \mathbb{C}$, Lemma 16.3 says that p is not a derivative of p_i for $i = 1, ..., n$. Since the p_i and the q_i for $i = 1, ..., n$ are derivatives of p and q respectively, it follows that $(p_i, q_i) < (p, q)$ for $i = 1, ..., n$ (by item 3 in Definition 12). So induction yields $p_i = q_i$ for $i = 1, ..., n$. Hence, $p = q$.
- Case 2: $p, q \in \mathbb{C}$.
  Since $p \in \mathbb{C}$, either $p =_{\rm AC} r^\omega = r(r^\omega)$ or $p =_{\rm AC} r'(r^\omega)$, where $r \in \mathbb{B}$ and r' is a proper derivative of r. In both cases $p = r'(r^\omega)$ with $r \in \mathbb{B}$ and r' a derivative (not necessarily proper) of r. Likewise, $q = s'(s^\omega)$ with $s \in \mathbb{B}$ and s' a derivative of s.
  By symmetry, it is sufficient to distinguish two cases: either r' is not normed, or both r' and s' are normed.
  - ⋆ Case 2.1: r' is not normed.
    Then by Lemma 10 $r'(r^\omega) = r'$. Since $g(r') \leq g(r) < g(p)$, item 2 in Definition 12 yields $(r', q) < (p, q)$. So, since $r' \leftrightarrows r'(r^\omega) \leftrightarrows q$, induction yields $r' = q$. Hence, $p = r'(r^\omega) = r' = q$.
  - ⋆ Case 2.2: Both r' and s' are normed.
    For convenience of notation put $N_0 = \max\{g(p), g(q)\}$. Again, we consider two cases: either there exists or there does not exist a term t with $g(t) < N_0$ and $p \leftrightarrows t$.
    - ∘ Case 2.2.1: There exists a term t with $g(t) < N_0$ and $p \leftrightarrows t$ (and so $q \leftrightarrows t$).
      Since by the assumption at case 2.2 r' is normed, and $r'(r^\omega) \leftrightarrows t$, there exists a derivative t' of t with $r^\omega \leftrightarrows t'$, and so $rt' \leftrightarrows t'$. Furthermore, Lemma 11 implies $g(t') \leq g(t) < N_0$, and so $g(rt' + t') < N_0$. So after using Lemma 17 to reduce rt' and t' to basic form, we can apply induction, by item 1 in Definition 12, to conclude $rt' = t'$. RSP$^\omega$ then yields $r^\omega = t'$, so $p = r't'$. By Lemma 17 $r't' = u$ with $u \in \mathbb{B}$ and $g(u) < N_0$. Thus, $p = u$. Likewise, $q = v$ for some basic term v with $g(v) < N_0$. Then $u \leftrightarrows p \leftrightarrows q \leftrightarrows v$, so since $g(u+v) < N_0$, induction yields $u = v$. Hence, $p = u = v = q$.
    - ∘ Case 2.2.2: For each term t, if $g(t) < N_0$ then $p \not\leftrightarrows t$ (and so $q \not\leftrightarrows t$).
      Since $p \leftrightarrows q$, the assumption of this case implies $g(p) = g(q)$.
      Note that the requirements A and B for Proposition 22 are satisfied, by the assumption at case 2.2.2 together with the induction hypothesis (item 1 of Definition 12). So we are allowed to apply Proposition 22 in this case.
      By the assumption at case 2.2 r' is normed, so since $r'(r^\omega) \leftrightarrows s'(s^\omega)$, there exists a derivative s'' of s such that $r^\omega \leftrightarrows s''(s^\omega)$. Likewise, $s^\omega \leftrightarrows r''(r^\omega)$ for some derivative r'' of r such that $r''(r^\omega) \leftrightarrows s^\omega$.
      Since $s''r''(r^\omega) \leftrightarrows s''(s^\omega) \leftrightarrows r^\omega \leftrightarrows r(r^\omega)$, and $g(s''r'' + r) < N_0$, Proposition 22 yields $s''r''(r^\omega) = r(r^\omega) \overset{\rm NEI1}{=} r^\omega$. RSP$^\omega$ then yields

$$r^\omega = (s''r'')^\omega. \tag{6}$$

Likewise,

$$s^\omega = (r''s'')^\omega. \tag{7}$$

Since $s''((r''s'')^\omega) \overset{\text{NEI1}}{=} s''((r''s'')((r''s'')^\omega)) \overset{\text{A5}}{=} (s''r'')(s''((r''s'')^\omega))$, RSP$^\omega$ yields

$$s''((r''s'')^\omega) = (s''r'')^\omega. \tag{8}$$

Since $r's''(s^\omega) \leftrightarrow r's''((r''s'')^\omega) \leftrightarrow r'((s''r'')^\omega) \leftrightarrow r'(r^\omega) \leftrightarrow s'(s^\omega)$, and $g(r's''+s') < N_0$, Proposition 22 yields

$$r's''(s^\omega) = s'(s^\omega). \tag{9}$$

So finally,

$$p =_{\text{AC}} r'(r^\omega) \overset{(6)}{=} r'((s''r'')^\omega) \overset{(8)}{=} r's''((r''s'')^\omega) \overset{(7)}{=} r's''(s^\omega) \overset{(9)}{=} s'(s^\omega) =_{\text{AC}} q. \qquad \square$$

3.7 An Example

We give an example as to how the construction in the completeness proof acts on particular pairs of bisimilar basic terms.

Example 1. $(a\delta + b)((c(a\delta + b))^\omega) \leftrightarrow (a\delta + bc)^\omega$.

This equivalence belongs with case 2.2.2. It can be derived as follows.

$$\begin{aligned}(a\delta + b)((c(a\delta + b))^\omega) &\overset{\text{NEI1}}{=} (a\delta + b)((c(a\delta + b))((c(a\delta + b))^\omega)) \\ &\overset{\text{A4,5}}{=} ((a\delta + b)c)((a\delta + b)((c(a\delta + b))^\omega)).\end{aligned}$$

Then RSP$^\omega$ yields

$$(a\delta + b)((c(a\delta + b))^\omega) = ((a\delta + b)c)^\omega. \tag{10}$$

So finally,

$$(a\delta + b)((c(a\delta + b))^\omega) \overset{(10)}{=} ((a\delta + b)c)^\omega \overset{\text{A4,5,7}}{=} (a\delta + bc)^\omega.$$

4 Conclusion

In this paper, Milner's axiomatization for iteration was restricted to the case of no-exit iteration, and it was proved that this yields a complete axiomatization for no-exit iteration in process algebra modulo bisimulation. The main new idea in the proof was to introduce a function ϕ which can help to minimize the argument p of a no-exit iteration term p^ω, in such a way that p does not contain any proper derivatives p' with $p'(p^\omega) \leftrightarrow p^\omega$. For example, using this function ϕ, the term $(aa)^\omega$ can be reduced to a^ω.

The completeness result in this paper may be a step forward to a positive answer to the question whether Milner's axiomatization is complete for iteration in process algebra modulo bisimulation. Namely, the main problem in solving this question is to deal with no-exit iteration terms p^ω where p is not minimal. Unfortunately, it is not obvious how to extend the definition of the function ϕ to all terms in process algebra with iteration. For example, consider the term

$$(a((a(ba + a))^* c))^\omega$$

where the argument $a((a(ba + a))^* c)$ of no-exit iteration is not minimal. Minimization of this argument would yield a so-called 'double-exit' term (with exits b and c), which cannot be expressed in process algebra with iteration modulo bisimulation (see [6, 1]). The only way to obtain a no-exit iteration term with a minimal argument in this particular case is to rewrite the term to

$$a((a(ba + a) + ca)^\omega)$$

A minimization strategy for all possible arguments of no-exit iteration would probably be the key to solving Milner's question.

References

1. L. Aceto and W.J. Fokkink. An equational axiomatization for multi-exit iteration. Report RS-96-22, BRICS, Aalborg University, 1996. Accepted for publication in *Information and Computation*.
2. L. Aceto, W.J. Fokkink, R.J. van Glabbeek, and A. Ingólfsdóttir. Axiomatizing prefix iteration with silent steps. *Information and Computation*, 127(1):26–40, 1996.
3. L. Aceto, W.J. Fokkink, and A. Ingólfsdóttir. A menagerie of non-finitely based process semantics over BPA*: from ready simulation to completed traces. Report RS-96-23, BRICS, Aalborg University, 1996.
4. J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the ACM*, 40(3):653–682, 1993.
5. J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In *Proceedings CONCUR'93*, LNCS 715, pp. 477–492. Springer, 1993.
6. J.A. Bergstra, I. Bethke, and A. Ponse. Process algebra with iteration and nesting. *The Computer Journal*, 37(4):243–258, 1994.
7. J.A. Bergstra, J.A. Hillebrand and A. Ponse. Grid protocols based on synchronous communication. *Science of Computer Programming*, 1997, To appear.
8. J.A. Bergstra and P. Klint. The discrete time toolbus. In *Proceedings AMAST'96*, LNCS 1101, pp. 286–305. Springer, 1996.
9. J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1/3):109–137, 1984.
10. J.A. Bergstra and J.W. Klop. Verification of an alternating bit protocol by means of process algebra. In *Proceedings Spring School on Mathematical Methods of Specification and Synthesis of Software Systems*, LNCS 215, pp. 9–23. Springer, 1985.
11. W.J. Fokkink. On the completeness of the equations for the Kleene star in bisimulation. In *Proceedings AMAST'96*, LNCS 1101, pp. 180–194. Springer, 1996.
12. W.J. Fokkink. An axiomatization for the terminal cycle. Logic Group Preprint Series 167, Utrecht University, 1996. Available at http://www.phil.ruu.nl.
13. W.J. Fokkink and R.J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation*, 126(1):1–10, 1996.
14. W.J. Fokkink and H. Zantema. Basic process algebra with iteration: completeness of its equational axioms. *The Computer Journal*, 37(4):259–267, 1994.
15. S.C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, pages 3–41. Princeton University Press, 1956.
16. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.
17. R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.
18. R. Milner. The polyadic π-calculus: a tutorial. In *Proceedings Marktoberdorf Summer School '91, Logic and Algebra of Specification*, NATO ASI Series F94, pp. 203–246. Springer, 1993.
19. D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings 5th GI Conference*, LNCS 104, pp. 167–183. Springer, 1981.
20. G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Aarhus University, 1981.
21. A. Ponse. Personal communication, March 1997. See also the handouts for the course "Concurrency and Distributed Systems", available at http://adam.wins.uva.nl/∼alban.
22. A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the ACM*, 13(1):158–169, 1966.
23. P.M. Sewell. Bisimulation is not finitely (first order) equationally axiomatisable. In *Proceedings LICS'94*, pp. 62–70. IEEE Computer Society Press, 1994.

This article was processed using the LaTeX macro package with LLNCS style