A Note on an Expressiveness Hierarchy for Multi-exit Iteration

Luca Aceto* Wan Fokkink† Anna Ingólfsdóttir*‡

Abstract

Multi-exit iteration is a generalization of the standard binary Kleene star operation. The addition of this construct to Basic Process Algebra (BPA) yields a more expressive language than that obtained by augmenting BPA with the standard binary Kleene star. This note offers an expressiveness hierarchy, modulo bisimulation equivalence, for the family of multi-exit iteration operators proposed by Bergstra, Bethke and Ponse.

AMS Subject Classification (1991): 68Q15, 68Q70.
CR Subject Classification (1991): D.3.1, F.1.1, F.4.1.
Keywords and Phrases: Concurrency, process algebra, Basic Process Algebra (BPA), multi-exit iteration, bisimulation, expressiveness.

1 Introduction

One of the classic topics in the theory of computation is the study of the axiomatics and expressiveness of variations on regular expressions. This field of research has been active since Kleene's original paper [16], and some of its crowning achievements may be found in, e.g., [12, 17, 18]. The investigation of variations on the language of regular expressions from the perspective of process theory was begun by Milner in his seminal paper [20]—where an implicational proof system was proposed for bisimulation equivalence over regular events, and the problem of its completeness raised—, and has received new impulse after the publication of [6]. In *op. cit.* Bergstra, Bethke and Ponse have investigated the expressive power of variations on standard process description languages in which infinite behaviours are defined by means of Kleene's star operation [16] rather than by means of systems of recursion equations, and have proposed an axiom system for bisimulation equivalence over the language of Basic Process Algebra [8] with the original binary version of the Kleene star operation (BPA*). The completeness of the axiom system proposed in [6] was proven by Fokkink and Zantema in [15], and this result has been followed by a series of contributions in which several notions of process equivalence have been equationally axiomatized over languages incorporating variations on the Kleene star operation—see, e.g., the handbook chapter [7] for a survey.

The motivation for the use of Kleene's star operation for the description of processes with infinite behaviour is that algebraic operators are easier to manipulate in an axiomatic setting than systems of recursion equations or recursive terms. Interestingly, however, as already noted by Milner in [20, Sect. 6], not every process defined using finite-state systems of recursion equations can be described, up to bisimulation equivalence, using only regular expressions. The limited expressive power of the Kleene star operation in denoting finite automata modulo bisimulation equivalence is highlighted in [5], where it is shown that the process described by the following re-

*BRICS (Basic Research in Computer Science), Centre of the Danish National Research Foundation, Department of Computer Science, Aalborg University, Fr. Bajersvej 7E, 9220 Aalborg Ø, Denmark. Email: {luca,annai}@cs.auc.dk.

†CWI, Department of Software Engineering, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: wan@cwi.nl.

‡Iceland Genomics Corporation, Snorrabraut 60, 105 Reykjavík, Iceland.

cursion equation

$$X \stackrel{\text{def}}{=} a \cdot (a \cdot X + b) + a \qquad (1)$$

cannot be expressed in the language BPA* modulo bisimulation equivalence. This has led the authors of [5] to propose the use of k-exit iteration in order to increase the expressive power of super-languages of BPA incorporating Kleene star-like operations. This operation has later been generalized to multi-exit iteration in [1]. Multi-exit iteration is a generalization of the standard binary Kleene star operation that allows for the description of agents whose behaviour is specified, up to bisimulation equivalence, by the following defining equation:

$$\begin{aligned}(P_1, \ldots, P_m)^*(Q_1, \ldots, Q_n) = \qquad (2)\\ P_1(P_2, \ldots, P_m, P_1)^*(Q_2, \ldots, Q_n, Q_1) + Q_1 \ ,\end{aligned}$$

where m and n are positive integers, and the P_i and the Q_j are process terms. For example, the term $(a, a)^*(a, b)$ denotes, up to isomorphism, the finite automaton associated with the variable X in (1).

Despite its seemingly arbitrary nature, multi-exit iteration is a generalization of the standard Kleene star operation whose roots may be found in the time honoured theory of flowchart schema (see, e.g., the references [19, 23]), and in the study of looping constructs in programming languages. For example, the flow of control of a process described using multi-exit iteration bears strong similarity to that of a flowchart schema in normal form [13, 14]. (For the sake of completeness, we recall that a flowchart schema is said to be in normal form if it is a finite tree-like schema whose branches may only bend back to ancestor nodes.) The literature on looping constructs in programming languages is replete with results on operations with single entry points and multiple exits. Here we limit ourselves to mentioning the references [9, 10, 22]. In particular, Peterson et al. defined in [22] a well formed program to be one in which all control structures have a single entry point, and showed that such a program need only use IF THEN, DO FOREVER, and multi level loop EXIT statements as control structures.

Since the addition of multi-exit iteration to a process algebra like BPA produces a language with a countably infinite collection of basic operations, it is natural to wonder whether one may obtain an equally expressive language by restricting the number of behaviours that can be specified on the left- and/or the right-hand sides of a multi-exit iteration.

By analogy with the aforementioned result from [5], it was proved in [1] that, in the presence of a non-empty set of actions, the sequence of k-exit iteration operations induces a hierarchy of super-languages of BPA with a strictly increasing expressive power modulo bisimulation equivalence. In light of that result, increasing the maximum number of exits (viz. the Q_j in (2)) allowed in a multi-exit iteration increases the expressive power of the language modulo bisimulation equivalence. Our aim in this note is to show that increasing the maximum number of processes on the left-hand side of the star in a multi-exit iteration also increases the expressive power of the language modulo bisimulation equivalence (see Theorem 3.1). The lesson that can be drawn from these expressiveness results is that the adoption of an algebraic language, that uses multi-exit iteration in lieu of recursive terms to describe infinite behaviours modulo bisimulation equivalence, forces one to introduce an infinite family of Kleene star-like operations. This should be compared with the situation in the theory of regular languages, where the Kleene star operation suffices to describe all infinite regular languages.

2 Background

For the sake of completeness and readability, we begin by recalling the relevant notions from [1] that will be needed in this note. The interested reader is referred to *op. cit.* and [5] for motivation and further information.

We assume a non-empty alphabet A of atomic actions, with typical elements a, b. The language BPA$^{me*}(A)$ of terms over Basic Process Algebra (BPA) with multi-exit iteration is defined inductively as follows:

- each $a \in A$ is a term;

- $P + Q$ and $P \cdot Q$ are terms, if so are P and Q;

- $(P_1, ..., P_m)^*(Q_1, ..., Q_n)$ is a term, if so are $P_1, ..., P_m$ and $Q_1, ..., Q_n$ for some positive integers m and n.

We shall use P, Q, R (possibly subscripted and/or superscripted) to range over $\mathrm{BPA}^{me*}(A)$. In writing terms over the above syntax, we shall always assume that the operation $\cdot$ binds stronger than $+$. In the sequel the operation $\cdot$ will often be omitted, so PQ denotes $P \cdot Q$. We shall use the symbol $\equiv$ to stand for syntactic equality of terms. For a positive integer i and action a, we write a^i for the term obtained by concatenating i copies of action a. For every natural number n, we shall write $[n]$ in lieu of $\{1, \ldots, n\}$.

Apart from actions, the signature of the language $\mathrm{BPA}^{me*}(A)$ includes the binary operations of alternative composition $+$ and sequential composition $\cdot$ familiar from the theory of Basic Process Algebra [4, 8], and a variation on the original binary version of the Kleene star operation [16], that will be referred to as multi-exit iteration. For positive integers m and n, the process term $(P_1, ..., P_m)^*(Q_1, ..., Q_n)$ stands for an agent whose behaviour is specified by the defining equation (2). In order to simplify notation in the presentation of the operational semantics for $\mathrm{BPA}^{me*}(A)$, we shall use the notion of 'vectors of processes'. A vector of processes is a tuple $(P_1, ..., P_m)$, where $m \geq 0$. We shall use $\vec{Q}, \vec{S}$ to denote such vectors of processes. In multi-exit iteration, the expressions at the left- and right-hand sides of the star are non-empty vectors of processes. Enclosing parentheses will always be omitted from vectors of length one, i.e., (P) will be written P.

The operational semantics for the language $\mathrm{BPA}^{me*}(A)$ is given by the labelled transition system

$$\left(\mathrm{BPA}^{me*}(A), \left\{\stackrel{a}{\rightarrow} | \ a \in A\right\}, \left\{\stackrel{a}{\rightarrow}\checkmark \mid a \in A\right\}\right) \ ,$$

where the transition relations $\stackrel{a}{\rightarrow}$ and the unary predicates $\stackrel{a}{\rightarrow}\checkmark$ are, respectively, the least subsets of $\mathrm{BPA}^{me*}(A) \times \mathrm{BPA}^{me*}(A)$ and $\mathrm{BPA}^{me*}(A)$ satisfying the rules in Table 1. Intuitively, a transition $P \stackrel{a}{\rightarrow} Q$ means that the system represented by the term P can perform the action a, thereby evolving into Q. The special symbol $\checkmark$ stands for (successful) termination;

$$\frac{}{a \stackrel{a}{\rightarrow}\checkmark}$$

$$\frac{P \stackrel{a}{\rightarrow}\checkmark}{P + Q \stackrel{a}{\rightarrow}\checkmark} \qquad \frac{Q \stackrel{a}{\rightarrow}\checkmark}{P + Q \stackrel{a}{\rightarrow}\checkmark}$$

$$\frac{P \stackrel{a}{\rightarrow} P'}{P + Q \stackrel{a}{\rightarrow} P'} \qquad \frac{Q \stackrel{a}{\rightarrow} Q'}{P + Q \stackrel{a}{\rightarrow} Q'}$$

$$\frac{P \stackrel{a}{\rightarrow}\checkmark}{P \cdot Q \stackrel{a}{\rightarrow} Q} \qquad \frac{P \stackrel{a}{\rightarrow} P'}{P \cdot Q \stackrel{a}{\rightarrow} P' \cdot Q}$$

$$\frac{P \stackrel{a}{\rightarrow}\checkmark}{(P, \vec{Q})^*(R, \vec{S}) \stackrel{a}{\rightarrow} (\vec{Q}, P)^*(\vec{S}, R)}$$

$$\frac{P \stackrel{a}{\rightarrow} P'}{(P, \vec{Q})^*(R, \vec{S}) \stackrel{a}{\rightarrow} P' \cdot (\vec{Q}, P)^*(\vec{S}, R)}$$

$$\frac{R \stackrel{a}{\rightarrow}\checkmark}{(P, \vec{Q})^*(R, \vec{S}) \stackrel{a}{\rightarrow}\checkmark} \qquad \frac{R \stackrel{a}{\rightarrow} R'}{(P, \vec{Q})^*(R, \vec{S}) \stackrel{a}{\rightarrow} R'}$$

Table 1: Transition Rules

therefore the interpretation of the statement $P \stackrel{a}{\rightarrow}\checkmark$ is that the process term P can terminate by performing a. Note that, for every term P, there is some action a for which either $P \stackrel{a}{\rightarrow} P'$ holds for some P', or $P \stackrel{a}{\rightarrow}\checkmark$ does.

Definition 2.1 *The term P' is a* derivative *of P if P can evolve into P' by zero or more transitions. A derivative P' of P is* proper *if P can evolve into P' by performing at least one transition.*

Process terms are considered modulo bisimulation equivalence [21].

Definition 2.2 *Two process terms P and Q are* bisimilar*, denoted by $P \leftrightarrows Q$, if there exists a symmetric binary relation $\mathcal{B}$ on process terms which relates P and Q, such that:*

- *if $R \mathcal{B} S$ and $R \stackrel{a}{\rightarrow} R'$, then there is a transition $S \stackrel{a}{\rightarrow} S'$ such that $R' \mathcal{B} S'$,*

- *if $R \mathcal{B} S$ and $R \stackrel{a}{\rightarrow}\checkmark$, then $S \stackrel{a}{\rightarrow}\checkmark$.*

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

Note that if P is bisimilar to Q, then every (proper) derivative of P is bisimilar to some (proper) derivative of Q, and vice versa.

The transition rules in Table 1 are in the 'path' format of Baeten and Verhoef [3]. Hence, bisimulation equivalence is a congruence with respect to all the operations in the signature of $\mathrm{BPA}^{me*}(A)$.

Process terms in $\mathrm{BPA}^{me*}(A)$ are *normed*, which means that they are able to terminate by embarking in a finite sequence of transitions. We call such a sequence a termination trace. The *norm* of a process term P, denoted by $|P|$, is the length of its shortest termination trace; this notion stems from [2]. Note that bisimilar process terms have the same norm. The following lemma, which is due to Caucal [11], is typical for normed processes, and will be useful in the technical developments to follow.

Lemma 2.3 *Let $P, Q, R, S \in \mathrm{BPA}^{me*}(A)$ be such that $PQ \leftrightarrow RS$. If $|Q| = |S|$, then $P \leftrightarrow R$ and $Q \leftrightarrow S$.*

A technical tool we shall use below is a weight function g that associates a natural number to each process term. This is defined thus:

- $g(a)$ is 0,
- $g(P + Q)$ is $\max\{g(P), g(Q)\} + 1$,
- $g(PQ)$ is $\max\{g(P), g(Q)\}$, and
- $g\big((P_1, ..., P_m)^*(Q_1, ..., Q_n)\big)$ is

$$\max\{g(P_i), g(Q_j) + 1 \mid i \in [m], j \in [n]\} \ .$$

The basic property of this weight function that we shall need is expressed in the lemma below (cf. [1, Lemma 3.5]).

Lemma 2.4 *If P' is a derivative of P, then $g(P') \leq g(P)$. Moreover, if*

- *$P \equiv P_1 + P_2$ for some terms P_1 and P_2, and P' is a proper derivative of P, or*
- *$P \equiv (P_1, ..., P_m)^*(Q_1, ..., Q_n)$, for some terms P_i ($i \in [m]$) and Q_j ($j \in [n]$), and P' is a proper derivative of some Q_j,*

then $g(P') < g(P)$.

The weight function g is designed to ensure that, as stated formally in the above lemma, exiting from multi-exit iterations and resolving nondeterministic choices strictly decrease the weight of terms. This property will be used in our proof by infinite descent of Theorem 3.1 to rule out the possibility that a term of minimum weight that is bisimilar to $(a, a^2, \ldots, a^{k+1})^*a$ is either the sum of two terms or a multi-exit iteration with fewer than $k + 1$ terms on the left-hand side of the star.

3 An Expressiveness Hierarchy

As shown in [5], the addition of multi-exit iteration to BPA yields a language that, modulo bisimulation equivalence, is strictly more expressive than that obtained by augmenting BPA with the standard binary Kleene star. More precisely, it is proven *ibidem* that, in the presence of at least two actions, the process $(a, a)^*(a, b)$ cannot be expressed, modulo bisimulation equivalence, in ACP [4], and *a fortiori* in BPA, enriched with the binary Kleene star (cf. Lemma 3.2.3 in *op. cit.*).

Let us say that a term of the form $(P_1, \ldots, P_m)^*(Q_1, \ldots, Q_n)$ has n-exit iteration. By analogy with the aforementioned result from [5], it was proved in [1] that, in the presence of a *non-empty* set of actions, the sequence of k-exit iteration operations induces a hierarchy of super-languages of BPA with a strictly increasing expressive power modulo bisimulation equivalence. To this end, it was shown in *op. cit.* that, for every positive integer k, the process

$$a^*(a, a^2, \ldots, a^{k+1})$$

cannot be specified using h-exit iteration with $h \leq k$, modulo bisimulation equivalence. (Cf. Corollary 4.5 in [1].)

In light of the above result, increasing the maximum number of exits allowed in a multi-exit iteration

increases the expressive power of the language modulo bisimulation equivalence. Our aim in this note is to show that increasing the maximum number of processes on the left-hand side of the star in a multi-exit iteration also increases the expressive power of the language modulo bisimulation equivalence.

Notation 1 *For every positive integer k, we write* BPA^{k*} *for the set of terms in the language* $\mathrm{BPA}^{me*}(A)$ *that may use multi-exit iteration operations whose first argument is a non-empty vector of processes of length at most k.*

Our aim is to prove the following theorem:

Theorem 3.1 *For every positive integer k, the process $(a, a^2, \ldots, a^{k+1})^* a$ cannot be expressed in the language* BPA^{k*} *modulo bisimulation equivalence.*

The remainder of this note will be devoted to a proof of the above result. To this end, it is sufficient to establish the following special case of the statement of our main result.

Proposition 3.2 *For every positive integer k, the process $(a, a^2, \ldots, a^{k+1})^* a$ cannot be expressed, modulo bisimulation equivalence, as a term in the language* BPA^{k*} *of the form $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ with $|Q_j| = 1$, for every $j \in [m]$.*

Indeed, using the above result, we can prove Theorem 3.1 thus:

Proof of Theorem 3.1: Assume, towards a contradiction, that there is a term P in the language BPA^{k*} that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$. Assume, furthermore, that P is a process with this property with minimum weight $g(P)$. We proceed with the proof by analyzing the possible forms such a P may take.

It is easy to see that P can neither have the form a nor the form $P_1 P_2$ for some processes P_1 and P_2. Indeed, this follows because bisimilar processes have equal norm, but any process of the form $P_1 P_2$ has norm at least two and $(a, a^2, \ldots, a^{k+1})^* a$ has norm one.

We claim that P cannot have the form $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ either. To see this, note, first of all, that, by Proposition 3.2, P cannot have the form $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$, with $|Q_j| = 1$ for every $j \in [m]$. If there is some Q_j ($j \in [m]$) whose norm is greater than one, then this Q_j affords a transition $Q_j \stackrel{a}{\to} Q'_j$ for some process Q'_j. It follows that, for some positive integer ℓ,

$$(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \stackrel{a^\ell}{\to} Q'_j \ .$$

Since the terms $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ and $(a, a^2, \ldots, a^{k+1})^* a$ are bisimilar, there is a derivative R of the latter term such that

$$(a, a^2, \ldots, a^{k+1})^* a \stackrel{a^\ell}{\to} R \quad \text{and} \quad Q'_j \leftrightarrow R \ .$$

As $(a, a^2, \ldots, a^{k+1})^* a$ is easily seen to be a derivative of R, we have that Q'_j has a derivative that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$, and thus that Q_j has a *proper* derivative Q' that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$. By Lemma 2.4, the value of $g(Q')$ is strictly smaller than $g(P)$. This contradicts our assumption that P was a process with minimum weight in BPA^{k*} that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$.

From the above reasoning, it follows that P can only have the form $P_1 + P_2$ for some processes P_1 and P_2. Since

$$(a, a^2, \ldots, a^{k+1})^* a \stackrel{a^n}{\to} (a, a^2, \ldots, a^{k+1})^* a \ ,$$

where $n = (k+1)(k+2)/2$, and $P \equiv P_1 + P_2$ is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$, there is a process P' such that

$$P \stackrel{a^n}{\to} P' \qquad \text{and} \qquad P' \leftrightarrow (a, a^2, \ldots, a^{k+1})^* a \ .$$

By Lemma 2.4, since P' is a proper derivative of $P \equiv P_1 + P_2$, the value of $g(P')$ is strictly smaller than $g(P)$. This contradicts our assumption that P was a process with minimum weight in BPA^{k*} that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$.

It follows that no term in BPA^{k*} can be bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$, which was to be shown. □

To complete the proof, we are therefore left to show Proposition 3.2. This result is an immediate consequence of the second statement in the following lemma.

Lemma 3.3 *Assume that $Q_1, \ldots, Q_m$ are processes with norm one. Then the following statements hold:*

1. *For every positive integer i, if $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ is bisimilar to $(a^i, R_1, \ldots, R_n)^* a$ $(n \geq 0)$, then*
   - *$P_1 \leftrightarrow a^i$ and*
   - *$(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1)$ is bisimilar to $(R_1, \ldots, R_n, a^i)^* a$.*
2. *For every h, k such that $k \geq h \geq 1$, if $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \leftrightarrow (a, a^2, \ldots, a^k)^* a$, then $h = k$, and $P_i \leftrightarrow a^i$ for every $i \in [k]$.*

Proof: We prove the two statements separately.

- Proof of Statement 1. We consider two cases, depending on whether $i = 1$ or not. In both cases of the proof, we use the fact that, as $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \leftrightarrow (a^i, R_1, \ldots, R_n)^* a$ holds by assumption, P_1 can perform an a-labelled transition and no transition labelled with actions different from a.

  Assume that $i = 1$. Then P_1 has no transitions of the form $P_1 \stackrel{a}{\rightarrow} P_1'$. Indeed, if $P_1 \stackrel{a}{\rightarrow} P_1'$ holds, then so does

  $$(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \stackrel{a}{\rightarrow} P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \ . \quad (3)$$

  Since $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ is bisimilar to $(a, R_1, \ldots, R_n)^* a$ by assumption, there is a transition

  $$(a, R_1, \ldots, R_n)^* a \stackrel{a}{\rightarrow} R$$

  for some R such that

  $$P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \leftrightarrow R \ .$$

  The only candidate for this R is the term $(R_1, \ldots, R_n, a)^* a$. However, the term $(R_1, \ldots, R_n, a)^* a$ has norm one, whereas

  $$|P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1)| \geq 2 \ .$$

  It follows that

  $$P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \not\leftrightarrow (R_1, \ldots, R_n, a)^* a \ ,$$

  and thus that $P_1 \stackrel{a}{\rightarrow} \checkmark$ is the only transition afforded by P_1. We can now conclude that $P_1 \leftrightarrow a$ and

  $$(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \leftrightarrow (R_1, \ldots, R_n, a)^* a$$

  both hold, which was to be shown.

  Assume now that i is greater than 1. Reasoning as in the previous case, it is not hard to see that P_1 only affords transitions of the form $P_1 \stackrel{a}{\rightarrow} P_1'$. For every such transition, we have a transition of the form (3) out of $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$. These transitions can only be matched by the transition

  $$(a^i, R_1, \ldots, R_n)^* a \stackrel{a}{\rightarrow} a^{i-1}(R_1, \ldots, R_n, a^i)^* a$$

  from $(a^i, R_1, \ldots, R_n)^* a$. It follows that, for every term P_1' such that $P_1 \stackrel{a}{\rightarrow} P_1'$, it holds that

  $$P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \leftrightarrow a^{i-1}(R_1, \ldots, R_n, a^i)^* a \ .$$

  Since the norm of both of the terms $(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1)$ and $(R_1, \ldots, R_n, a_i)^* a$ is one by the proviso of the lemma, Lemma 2.3 yields that $P_1' \leftrightarrow a^{i-1}$ and

  $$(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \leftrightarrow (R_1, \ldots, R_n, a^i)^* a \ .$$

  To complete the proof for this case, note that since every term that can be reached from P_1 via an a-labelled transition is bisimilar to a^{i-1}, from our previous observations it follows that P_1 is bisimilar to a^i.

- Proof of Statement 2. Assume that $k \geq h \geq 1$ and

  $$(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \leftrightarrow (a, a^2, \ldots, a^k)^* a \ .$$

  Using statement 1 of the lemma repeatedly, we have that $P_i \leftrightarrow a^i$ for every $i \in [h]$, and

  $$(P_1, \ldots, P_h)^*(Q_{\ell+1}, \ldots, Q_m, Q_1, \ldots, Q_\ell) \leftrightarrow (a^{h+1}, \ldots, a^k, a_1, \ldots, a^h)^* a \ ,$$

where $\ell = h \mod m$.

If $h < k$, then statement 1 of the lemma would entail that

$$a \leftrightarrow P_1 \leftrightarrow a^{h+1} \ ,$$

which is impossible because $a \not\leftrightarrow a^{h+1}$, as $h \geq 1$. It follows that $h = k$ holds, and we are done.

This completes the proof of the lemma. □

Acknowledgements: The research reported in this note originated from a question posed to the authors by Kim G. Larsen. We thank Gerald Lüttgen, and an anonymous referee for suggestions that led to improvements in the paper. Luca Aceto's work was partly supported by an invited professorship at Reykjavík University, where the revision of this work was finalized. The work reported in this paper was partly carried out while Anna Ingólfsdóttir was at deCODE Genetics, Sturlugata 8, 101 Reykjavík, Iceland.

References

[1] L. Aceto and W. J. Fokkink, *An equational axiomatization for multi-exit iteration*, Information and Computation, 137 (1997), pp. 121–158.

[2] J. Baeten, J. Bergstra, and J. Klop, *Decidability of bisimulation equivalence for processes generating context-free languages*, J. Assoc. Comput. Mach., 40 (1993), pp. 653–682.

[3] J. 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.

[4] J. Baeten and W. Weijland, *Process Algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[5] J. Bergstra, I. Bethke, and A. Ponse, *Process algebra with iteration*, Report CS-R9314, Programming Research Group, University of Amsterdam, 1993.

[6] ——, *Process algebra with iteration and nesting*, Computer Journal, 37 (1994), pp. 243–258.

[7] J. Bergstra, W. J. Fokkink, and A. Ponse, *Process algebra with recursive operations*, in Handbook of Process Algebra, J. Bergstra, A. Ponse and S. Smolka, eds., Amsterdam, 2001, North-Holland, pp. 333–389.

[8] J. Bergstra and J. Klop, *Fixed point semantics in process algebras*, Report IW 206, Mathematisch Centrum, Amsterdam, 1982.

[9] G. von Bochmann, *Multiple exits from a loop without the GOTO*, Communications of the ACM, 16 (1973), pp. 443–444.

[10] P. A. Buhr, *A case for teaching multi-exit loops to beginning programmers*, SIGPLAN Notices 20 (1985), pp. 14–22.

[11] D. Caucal, *Graphes canoniques de graphes algébriques*, Theoretical Informatics and Applications, 24 (1990), pp. 339–352.

[12] J. H. Conway, *Regular Algebra and Finite Machines*, Mathematics Series, R. Brown and J. De Wet eds., Chapman and Hall, London, United Kingdom, 1971.

[13] D. Cooper, *Programs for mechanical program verification*, in Machine Intelligence, B. Meltzer and D. Michie, eds., vol. 6, Edinburgh, 1971, Edinburgh University Press, pp. 43–59.

[14] E. Engeler, *Structure and meanings of elementary programs*, in Symposium on Semantics of Algorithmic Languages, E. Engeler, ed., vol. 188 of Lecture Notes in Mathematics, Springer-Verlag, 1971, pp. 89–101.

[15] W. J. Fokkink and H. Zantema, *Basic process algebra with iteration: Completeness of its equational axioms*, Computer Journal, 37 (1994), pp. 259–267.

[16] S. Kleene, *Representation of events in nerve nets and finite automata*, in Automata Studies, C. Shannon and J. McCarthy, eds., Princeton University Press, 1956, pp. 3–41.

[17] D. Kozen, *A completeness theorem for Kleene algebras and the algebra of regular events*, Information and Computation, 110 (1994), pp. 366–390.

[18] D. Krob, *Complete systems of B-rational identities*, Theoretical Comput. Sci., 89 (1991), pp. 207–343.

[19] Z. Manna, *Mathematical Theory of Computation*, McGraw-Hill Book Co., 1974.

[20] R. Milner, *A complete inference system for a class of regular behaviours*, J. Comput. System Sci., 28 (1984), pp. 439–466.

[21] 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.

[22] W. W. Peterson, T. Kasami, and N. Tokura, *On the capabilities of while, repeat, and exit statements*, Communications of the ACM, 16 (1973), pp. 503–512.

[23] Y. Yanov, *The logical schemes of algorithms*, in Problems in Cybernetics, vol. 1, New York, 1960, Pergamon Press, pp. 82–140. English translation.