



## UvA-DARE (Digital Academic Repository)

### Process algebra, process scheduling, and mutual exclusion

Middelburg, C.A.

**DOI**

[10.48550/arXiv.2003.00473](https://doi.org/10.48550/arXiv.2003.00473)

**Publication date**

2020

**Document Version**

Submitted manuscript

**License**

Unspecified

[Link to publication](#)

**Citation for published version (APA):**

Middelburg, C. A. (2020). *Process algebra, process scheduling, and mutual exclusion*. (v1 ed.) ArXiv. <https://doi.org/10.48550/arXiv.2003.00473>

**General rights**

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

**Disclaimer/Complaints regulations**

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

# Process Algebra, Process Scheduling, and Mutual Exclusion

C.A. Middelburg

Informatics Institute, Faculty of Science, University of Amsterdam  
Science Park 904, 1098 XH Amsterdam, the Netherlands  
C.A.Middelburg@uva.nl

**Abstract.** In the case of multi-threading as found in contemporary programming languages, parallel processes are interleaved according to what is known as a process-scheduling policy in the field of operating systems. In a previous paper, we extend ACP with this form of interleaving. In the current paper, we do so with the variant of ACP known as  $ACP_\epsilon$ . The choice of  $ACP_\epsilon$  stems from the need to cover more process-scheduling policies. We show that a process-scheduling policy supporting mutual exclusion of critical subprocesses is now covered.

*Keywords:* process algebra, process scheduling, mutual exclusion, empty process, strategic interleaving, semaphore

*1998 ACM Computing Classification:* D.1.3, D.4.1, F.1.2

## 1 Introduction

In algebraic theories of processes, such as ACP [1], CCS [11], and CSP [9], processes are discrete behaviours that proceed by doing steps in a sequential fashion. In these theories, parallel composition of two processes is usually interpreted as arbitrary interleaving of the steps of the processes concerned. Arbitrary interleaving turns out to be appropriate for many applications and to facilitate formal algebraic reasoning. Our interest in process-scheduling policies originates from the feature of multi-threading found in contemporary programming languages such as Java [7] and C# [8]. Multi-threading gives rise to parallel composition of processes. However, the steps of the processes concerned are interleaved according to what is known as a process-scheduling policy in the field of operating systems. We use the term strategic interleaving for this more constrained form of interleaving and the term interleaving strategy instead of process-scheduling policy.

Nowadays, multi-threading is often used in the implementation of systems. Because of this, in many systems, for instance hardware/software systems, we

have to do with parallel processes that may best be considered to be interleaved in an arbitrary way as well as parallel processes that may best be considered to be interleaved according to some interleaving strategy. To our knowledge, there exists no work on strategic interleaving in the setting of a general algebraic theory of processes like ACP, CCS and CSP. This is what motivated us to do the work presented in [3], namely extending ACP such that it supports both arbitrary interleaving and strategic interleaving.

The extension of ACP presented in [3] is based on a generic interleaving strategy that can be instantiated with different specific interleaving strategies. The main reason for doing the work presented in the current paper is the finding that the generic interleaving strategy concerned cannot be instantiated with: (a) interleaving strategies where the data relevant to the process-scheduling decision making may be such that none of the processes concerned can be given a turn, (b) interleaving strategies where the data relevant to the process-scheduling decision making must be updated on successful termination of one of the processes concerned, and (c) interleaving strategies where the process-scheduling decision making may be adjusted by steps of the processes concerned that are solely intended to change the data relevant to the process-scheduling decision making. Another reason is the fact that it is not shown in [3] that the generic interleaving strategy can be instantiated with non-trivial specific interleaving strategies.

In this paper, we rectify the shortcomings of the generic interleaving strategy on which the extension of ACP presented in [3] is based by starting from  $ACP_\epsilon$ , i.e. ACP extended with a constant for a process that can only terminate successfully, and widening the generic interleaving strategy. Moreover, it is shown that the widened generic interleaving strategy can be instantiated with an interleaving strategy that supports mutual exclusion of critical subprocesses of the different processes being interleaved.

The relevance of developing theory about strategic interleaving lies in the fact that strategic interleaving is quite different from arbitrary interleaving. This is for example illustrated by the following easy to demonstrate phenomena: (a) sometimes a particular interleaving strategy leads to inactiveness whereas arbitrary interleaving does not lead to inactiveness and (b) whether the interleaving of certain processes leads to inactiveness depends on the interleaving strategy used.

The rest of this paper is organized as follows. First, we review  $ACP_\epsilon$  (Section 2.1) and guarded recursion in the setting of  $ACP_\epsilon$  (Section 2.2). Then, we extend  $ACP_\epsilon$  with strategic interleaving (Section 3.1) and present some properties concerning the connection between  $ACP_\epsilon$  and this extension (Section 3.2). After that, we show that the generic interleaving strategy on which the extension is based can be instantiated with the interleaving strategy mentioned above (Section 4). Finally, we make some concluding remarks (Section 5).

In [3], [10], and the current paper, different variants of ACP are extended with strategic interleaving. Because of this, there is some text overlap between these papers. The current paper can be looked upon as supplementary material to [3], but can be read independently.

## 2 ACP<sub>ε</sub> with Guarded Recursion

In this section, we give a survey of ACP<sub>ε</sub> (ACP with the empty process) and guarded recursion in the setting of ACP<sub>ε</sub>. For a more comprehensive treatment, the reader is referred to [1].

### 2.1 ACP<sub>ε</sub>

In ACP<sub>ε</sub>, it is assumed that a fixed but arbitrary set  $A$  of *actions*, with  $\delta, \epsilon \notin A$ , has been given. We write  $A_\delta$  for  $A \cup \{\delta\}$  and  $A_\epsilon$  for  $A \cup \{\epsilon\}$ . It is further assumed that a fixed but arbitrary commutative and associative *communication* function  $\gamma: A_\delta \times A_\delta \rightarrow A_\delta$ , with  $\gamma(\delta, a) = \delta$  for all  $a \in A_\delta$ , has been given. The function  $\gamma$  is regarded to give the result of synchronously performing any two actions for which this is possible, and to give  $\delta$  otherwise.

The signature of ACP<sub>ε</sub> consists of the following constants and operators:

- the *inaction* constant  $\delta$ ;
- the *empty process* constant  $\epsilon$ ;
- for each  $a \in A$ , the *action* constant  $a$ ;
- the binary *alternative composition* operator  $_ + _$ ;
- the binary *sequential composition* operator  $_ \cdot _$ ;
- the binary *parallel composition* operator  $_ \parallel _$ ;
- the binary *left merge* operator  $_ \parallel\!\!\! \parallel _$ ;
- the binary *communication merge* operator  $_ | _$ ;
- for each  $H \subseteq A$ , the unary *encapsulation* operator  $\partial_H$ .

We assume that there is a countably infinite set  $\mathcal{X}$  of variables which contains  $x, y$  and  $z$  with and without subscripts. Terms are built as usual. We use infix notation for the binary operators. The precedence conventions used with respect to the operators of ACP<sub>ε</sub> are as follows:  $+$  binds weaker than all others,  $\cdot$  binds stronger than all others, and the remaining operators bind equally strong.

The constants of ACP<sub>ε</sub> can be explained as follows ( $a \in A$ ):

- $\delta$  denotes the process that cannot do anything;
- $\epsilon$  denotes the process that can only terminate successfully;
- $a$  denotes the process that can first perform action  $a$  and after that terminate successfully.

Let  $t$  and  $t'$  be closed ACP<sub>ε</sub> terms denoting processes  $p$  and  $p'$ . Then the operators of ACP<sub>ε</sub> can be explained as follows:

- $t + t'$  denotes the process that can behave as  $p$  or as  $p'$ , but not both;
- $t \cdot t'$  denotes the process that can first behave as  $p$  and after that as  $p'$ ;
- $t \parallel t'$  denotes the process that can behave as  $p$  and  $p'$  in parallel;
- $t \parallel\!\!\! \parallel t'$  denotes the same process as  $t \parallel t'$ , except that it must start with performing an action of  $p$ ;
- $t | t'$  denotes the same process as  $t \parallel t'$ , except that it must start with performing an action of  $p$  and an action of  $p'$  synchronously;

**Table 1.** Axioms of  $ACP_\epsilon$ 

$x + y = y + x$	A1	$x \parallel y =$	
$(x + y) + z = x + (y + z)$	A2	$x \parallel y + y \parallel x + x \mid y + \partial_A(x) \cdot \partial_A(y)$	CM1T
$x + x = x$	A3	$\epsilon \parallel x = \delta$	CM2T
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4	$a \cdot x \parallel y = a \cdot (x \parallel y)$	CM3
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5	$(x + y) \parallel z = x \parallel z + y \parallel z$	CM4
$x + \delta = x$	A6	$\epsilon \mid x = \delta$	CM5T
$\delta \cdot x = \delta$	A7	$x \mid \epsilon = \delta$	CM6T
$x \cdot \epsilon = x$	A8	$a \cdot x \mid b \cdot y = \gamma(a, b) \cdot (x \parallel y)$	CM7
$\epsilon \cdot x = x$	A9	$(x + y) \mid z = x \mid z + y \mid z$	CM8
		$x \mid (y + z) = x \mid y + x \mid z$	CM9
$\partial_H(\epsilon) = \epsilon$	D0	$a \mid b = \gamma(a, b)$	CM10
$\partial_H(a) = a$ if $a \notin H$	D1	$\delta \mid x = \delta$	CM11
$\partial_H(a) = \delta$ if $a \in H$	D2	$x \mid \delta = \delta$	CM12
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3		
$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$	D4		

–  $\partial_H(t)$  denotes the process that can behave the same as  $p$ , except that actions from  $H$  are blocked.

The operators  $\parallel$  and  $\mid$  are of an auxiliary nature. They are needed to axiomatize  $ACP_\epsilon$ .

The axioms of  $ACP_\epsilon$  are the equations given in Table 1. In these equations,  $a$  and  $b$  stand for arbitrary constants of  $ACP_\epsilon$  that differ from  $\epsilon$ , and  $H$  stands for an arbitrary subset of  $A$ . Moreover,  $\gamma(a, b)$  stands for the action constant for the action  $\gamma(a, b)$ . In D1 and D2, side conditions restrict what  $a$  and  $H$  stand for.

In some presentations of  $ACP_\epsilon$ , e.g. in [1],  $\partial_A(x) \cdot \partial_A(y)$  is replaced by  $\surd(x) \cdot \surd(y)$  in CM1T. However,  $\partial_A$  and  $\surd$  have the same axioms. In other presentations of  $ACP_\epsilon$ ,  $\gamma(a, b)$  is frequently replaced by  $a \mid b$  in CM7. By CM10, which is more often called CF, this replacement give rise to an equivalent axiomatization. In other presentations of  $ACP_\epsilon$ , CM11 and CM12 are usually absent. These equations are not derivable from the other axioms, but all there closed substitution instances are derivable from the other axioms. Moreover, CM11 and CM12 hold in virtually all models of  $ACP_\epsilon$  that have been devised.

## 2.2 Guarded Recursion

A closed  $ACP_\epsilon$  term denotes a process with a finite upper bound to the number of actions that it can perform. Guarded recursion allows the description of processes without a finite upper bound to the number of actions that it can perform.

**Table 2.** Axioms for guarded recursion

$\langle X E \rangle = \langle t E \rangle$	if $X = t \in E$	RDP
$E \Rightarrow X = \langle X E \rangle$	if $X \in V(E)$	RSP

This section applies to both  $ACP_\epsilon$  and its extension  $ACP_\epsilon+SI$  introduced in Section 3. Therefore, in this section, let  $PA$  be  $ACP_\epsilon$  or  $ACP_\epsilon+SI$ .

Let  $t$  be a  $PA$  term containing a variable  $X$ . Then an occurrence of  $X$  in  $t$  is *guarded* if  $t$  has a subterm of the form  $a \cdot t'$  where  $a \in A$  and  $t'$  is a  $PA$  term containing this occurrence of  $X$ . A  $PA$  term  $t$  is a *guarded*  $PA$  term if all occurrences of variables in  $t$  are guarded.

A *guarded recursive specification* over  $PA$  is a set  $\{X_i = t_i \mid i \in I\}$ , where  $I$  is finite or countably infinite set, each  $X_i$  is a variable from  $\mathcal{X}$ , each  $t_i$  is either a guarded  $PA$  term in which variables other than the variables from  $\{X_i \mid i \in I\}$  do not occur or a  $PA$  term rewritable to such a term using the axioms of  $PA$  in either direction and/or the equations in  $\{X_j = t_j \mid j \in I \wedge i \neq j\}$  from left to right, and  $X_i \neq X_j$  for all  $i, j \in I$  with  $i \neq j$ .

We write  $V(E)$ , where  $E$  is a guarded recursive specification, for the set of all variables that occur in  $E$ .

A solution of a guarded recursive specification  $E$  in some model of  $PA$  is a set  $\{p_X \mid X \in V(E)\}$  of elements of the carrier of that model such that each equation in  $E$  holds if, for all  $X \in V(E)$ ,  $X$  is assigned  $p_X$ . We are only interested in models of  $PA$  in which guarded recursive specifications have unique solutions.

We extend  $PA$  with guarded recursion by adding constants for solutions of guarded recursive specifications over  $PA$  and axioms concerning these additional constants. For each guarded recursive specification  $E$  over  $PA$  and each  $X \in V(E)$ , we add a constant, denoted by  $\langle X|E \rangle$ , that stands for the unique solution of  $E$  for  $X$  to the constants of  $PA$ . We add the equation RDP and the conditional equation RSP given in Table 2 to the axioms of  $PA$ . In RDP and RSP,  $X$  stands for an arbitrary variable from  $\mathcal{X}$ ,  $t$  stands for an arbitrary  $PA$  term,  $E$  stands for an arbitrary guarded recursive specification over  $PA$ , and the notation  $\langle t|E \rangle$  is used for  $t$  with, for all  $X \in V(E)$ , all occurrences of  $X$  in  $t$  replaced by  $\langle X|E \rangle$ . Side conditions restrict what  $X$ ,  $t$  and  $E$  stand for. We write  $PA_{\text{rec}}$  for the resulting theory.

The equations  $\langle X|E \rangle = \langle t|E \rangle$  for a fixed  $E$  express that the constants  $\langle X|E \rangle$  make up a solution of  $E$  and the conditional equations  $E \Rightarrow X = \langle X|E \rangle$  express that this solution is the only one.

In extensions of  $ACP_\epsilon$  whose axioms include RSP, we have to deal with conditional equational formulas with a countably infinite number of premises. Therefore, infinitary conditional equational logic is used in deriving equations from the axioms of extensions of  $ACP_\epsilon$  whose axioms include RSP. A complete inference system for infinitary conditional equational logic can be found in, for example, [6]. It is noteworthy that derivations are allowed to be of countably infinite length in infinitary conditional equational logic.

### 3 Strategic Interleaving

In this section, we extend  $ACP_\epsilon$  with strategic interleaving, i.e. interleaving according to some interleaving strategy. Interleaving strategies are abstractions of scheduling algorithms. Interleaving according to some interleaving strategy represents what really happens in the case of multi-threading as found in contemporary programming languages.

#### 3.1 $ACP_\epsilon$ with Strategic Interleaving

In the extension of ACP with strategic interleaving presented below, it is expected that an interleaving strategy uses the interleaving history in one way or another to make process-scheduling decisions.

The set  $\mathcal{H}$  of *interleaving histories* is the subset of  $(\mathbb{N}_1 \times \mathbb{N}_1)^*$  that is inductively defined by the following rules:<sup>1</sup>

- $\langle \rangle \in \mathcal{H}$ ;
- if  $i \leq n$ , then  $(i, n) \in \mathcal{H}$ ;
- if  $h \frown (i, n) \in \mathcal{H}$ ,  $j \leq n$ , and  $n - 1 \leq m \leq n + 1$ , then  $h \frown (i, n) \frown (j, m) \in \mathcal{H}$ .

The intuition concerning interleaving histories is as follows: if the  $k$ th pair of an interleaving history is  $(i, n)$ , then the  $i$ th process got a turn in the  $k$ th interleaving step and after its turn there were  $n$  processes to be interleaved. The number of processes to be interleaved may increase due to process creation (introduced below) and decrease due to successful termination of processes.

The presented extension of  $ACP_\epsilon$  is called  $ACP_\epsilon$ +SI (ACP with Strategic Interleaving). It is based on a generic interleaving strategy that can be instantiated with different specific interleaving strategies that can be represented in the way that is explained below.

In  $ACP_\epsilon$ +SI, it is assumed that the following has been given:<sup>2</sup>

- a fixed but arbitrary set  $S$ ;
- a fixed but arbitrary partial function  $\sigma_n : \mathcal{H} \times S \rightarrow \{1, \dots, n\}$  for each  $n \in \mathbb{N}_1$ ;
- a fixed but arbitrary total function  $\vartheta_n : \mathcal{H} \times S \times \{1, \dots, n\} \times \mathbf{A}_\epsilon \rightarrow S$  for each  $n \in \mathbb{N}_1$ ;
- a fixed but arbitrary set  $C \subset \mathbf{A}$  such that, for each  $c \in C$ ,  $\bar{c} \in \mathbf{A} \setminus C$  and, for each  $a, b \in \mathbf{A}$ ,  $\gamma(a, b) \neq c$ ,  $\gamma(a, b) \neq \bar{c}$ ,  $\gamma(a, c) = \delta$ , and  $\gamma(a, \bar{c}) = \delta$ .

The elements of  $S$  are called *control states*,  $\sigma_n$  is called an *abstract scheduler* (for  $n$  processes),  $\vartheta_n$  is called a *control state transformer* (for  $n$  processes), and the elements of  $C$  are called *control actions*. The intuition concerning  $S$ ,  $\sigma_n$ ,  $\vartheta_n$ , and  $C$  is as follows:

- the control states from  $S$  encode data that are relevant to the interleaving strategy, but not derivable from the interleaving history;

<sup>1</sup> The sequence notation used in this paper is explained in Appendix B.

<sup>2</sup> We write  $f : A \rightarrow B$  to indicate that  $f$  is a partial function from  $A$  to  $B$ .

- if  $\sigma_n(h, s) = i$ , then the  $i$ th process gets the next turn after interleaving history  $h$  in control state  $s$ ;
- if  $\sigma_n(h, s)$  is undefined, then no process gets the next turn after interleaving history  $h$  in control state  $s$ ;
- if  $\vartheta_n(h, s, i, a) = s'$ , then  $s'$  is the control state that arises from the  $i$ th process doing  $a$  after interleaving history  $h$  in control state  $s$ ;
- if  $\vartheta_n(h, s, i, \epsilon) = s'$ , then  $s'$  is the control state that arises from the  $i$ th process terminating successfully after interleaving history  $h$  in control state  $s$ ;
- if  $a \in C$ , then  $a$  is an explicit means to bring about a control state change and  $\bar{a}$  is left as a trace after  $a$  has been dealt with.

Thus,  $S$ ,  $\langle \sigma_n \rangle_{n \in \mathbb{N}_1}$ ,  $\langle \vartheta_n \rangle_{n \in \mathbb{N}_1}$ , and  $C$  make up a way to represent an interleaving strategy. This way to represent an interleaving strategy is engrafted on [12].

The intuition concerning the actions  $\bar{a}$ , where  $a \in C$ , is as follows: when a process performs a control action  $a$ , it will interact with the scheduler at hand and the action resulting from this interaction is the action  $\bar{a}$ .

In ACP+SI, the extension of ACP with strategic interleaving from [3], abstract schedulers must be total functions  $\sigma_n : \mathcal{H} \times S \rightarrow \{1, \dots, n\}$ , control state transformers must be total functions  $\vartheta_n : \mathcal{H} \times S \times \{1, \dots, n\} \times \mathbf{A} \rightarrow S$ , and control actions are not distinguished from other actions. The widenings chosen in ACP $_\epsilon$ +SI rectify the shortcomings mentioned in Section 1. The widening with respect to the control state transformers would not be possible without the change from ACP to ACP $_\epsilon$ .

Consider the case where  $S$  is a singleton set, for each  $n \in \mathbb{N}_1$ ,  $\sigma_n$  is defined by

$$\begin{aligned} \sigma_n(\langle \rangle, s) &= 1, \\ \sigma_n(h \sim (j, n), s) &= (j \bmod n) + 1, \end{aligned}$$

for each  $n \in \mathbb{N}_1$ ,  $\vartheta_n$  is defined by

$$\vartheta_n(h, s, i, a) = s,$$

and  $C$  is the empty set. In this case, the interleaving strategy corresponds to the round-robin scheduling algorithm for the case where each process is given only one turn in a row. More advanced strategies can be obtained if the scheduling makes more advanced use of the interleaving history and the control state. An example is given in Section 4.

In ACP $_\epsilon$ +SI, it is also assumed that a fixed but arbitrary finite or countably infinite set  $D$  of *data* and a fixed but arbitrary function  $\phi : D \rightarrow P$ , where  $P$  is the set of all closed terms over the signature of ACP $_\epsilon$ +SI (given below), have been given and that, for each  $d \in D$ ,  $\text{cr}(d), \bar{\text{cr}}(d) \in \mathbf{A} \setminus (C \cup \{\bar{c} \mid c \in C\})$  and, for each  $a, b \in \mathbf{A}$ ,  $\gamma(a, b) \neq \text{cr}(d)$ ,  $\gamma(a, b) \neq \bar{\text{cr}}(d)$ ,  $\gamma(a, \text{cr}(d)) = \delta$ , and  $\gamma(a, \bar{\text{cr}}(d)) = \delta$ . The action  $\text{cr}(d)$  can be considered a process creation request and the action  $\bar{\text{cr}}(d)$  can be considered a process creation act. They stand for the request to start the process denoted by  $\phi(d)$  in parallel with the requesting process and the act of carrying out that request, respectively.



**Table 3.** Axioms for strategic interleaving

$\llbracket_{h,s}^n(x_1, \dots, x_n) = \delta$	if $\sigma_n(h, s)$ is undefined	SI0
$\llbracket_{h,s}^n(x_1, \dots, x_n) = \llbracket_{h,s}^{n, \sigma_n(h,s)}(x_1, \dots, x_n)$	if $\sigma_n(h, s)$ is defined	SI1
$\llbracket_{h,s}^{n,i}(x_1, \dots, x_{i-1}, \delta, x_{i+1}, \dots, x_n) = \delta$		SI2
$\llbracket_{h,s}^{1,i}(\epsilon) = \epsilon$		SI3T
$\llbracket_{h,s}^{n+1,i}(x_1, \dots, x_{i-1}, \epsilon, x_{i+1}, \dots, x_{n+1}) =$ $\llbracket_{h^{\curvearrowright(i,n)}, \vartheta_{n+1}(h,s,i,\epsilon)}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1})$		SI4T
$\llbracket_{h,s}^{n,i}(x_1, \dots, x_{i-1}, a \cdot x'_i, x_{i+1}, \dots, x_n) =$ $a \cdot \llbracket_{h^{\curvearrowright(i,n)}, \vartheta_n(h,s,i,a)}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)$	if $a \notin C$	SI5Ta
$\llbracket_{h,s}^{n,i}(x_1, \dots, x_{i-1}, \bar{a} \cdot x'_i, x_{i+1}, \dots, x_n) =$ $\bar{a} \cdot \llbracket_{h^{\curvearrowright(i,n)}, \vartheta_n(h,s,i,a)}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)$	if $a \in C$	SI5Tb
$\llbracket_{h,s}^{n,i}(x_1, \dots, x_{i-1}, \mathbf{cr}(d) \cdot x'_i, x_{i+1}, \dots, x_n) =$ $\overline{\mathbf{cr}}(d) \cdot \llbracket_{h^{\curvearrowright(i,n+1)}, \vartheta_n(h,s,i,\mathbf{cr}(d))}^{n+1}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n, \phi(d))$		SI7
$\llbracket_{h,s}^{n,i}(x_1, \dots, x_{i-1}, x'_i + x''_i, x_{i+1}, \dots, x_n) =$ $\llbracket_{h,s}^{n,i}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n) + \llbracket_{h,s}^{n,i}(x_1, \dots, x_{i-1}, x''_i, x_{i+1}, \dots, x_n)$		SI8

The signature of  $\text{ACP}_\epsilon + \text{SI}$  consists of the constants and operators from the signature of  $\text{ACP}$  and in addition the following operators:

- the  $n$ -ary *strategic interleaving* operator  $\llbracket_{h,s}^n$  for each  $n \in \mathbb{N}_1$ ,  $h \in \mathcal{H}$ , and  $s \in S$ ;
- the  $n$ -ary *positional strategic interleaving* operator  $\llbracket_{h,s}^{n,i}$  for each  $n, i \in \mathbb{N}_1$  with  $i \leq n$ ,  $h \in \mathcal{H}$ , and  $s \in S$ .

The strategic interleaving operators can be explained as follows:

- a closed term of the form  $\llbracket_{h,s}^n(t_1, \dots, t_n)$  denotes the process that results from interleaving of the  $n$  processes denoted by  $t_1, \dots, t_n$  after interleaving history  $h$  in control state  $s$ , according to the interleaving strategy represented by  $S$ ,  $\langle \sigma_n \rangle_{n \in \mathbb{N}_1}$ ,  $\langle \vartheta_n \rangle_{n \in \mathbb{N}_1}$ , and  $C$ .

The positional strategic interleaving operators are auxiliary operators used to axiomatize the strategic interleaving operators. The role of the positional strategic interleaving operators in the axiomatization is similar to the role of the left merge operator found in  $\text{ACP}_\epsilon$ .

The axioms of  $\text{ACP}_\epsilon + \text{SI}$  are the axioms of  $\text{ACP}_\epsilon$  and in addition the equations given in Table 3.<sup>3</sup> In the additional equations,  $n$  and  $i$  stand for arbitrary numbers from  $\mathbb{N}_1$  with  $i \leq n$ ,  $h$  stands for an arbitrary interleaving history from

<sup>3</sup> There is no axiom named SI6 because for common axioms of  $\text{ACP} + \text{SI}$  and  $\text{ACP}_\epsilon + \text{SI}$  the names introduced in [3] have been adopted.

**Table 4.** Alternative axioms for SI2

$\llbracket_{h,s}^{1,i}(\delta) = \delta$	SI2a
$\llbracket_{h,s}^{n+1,i}(x_1, \dots, x_{i-1}, \delta, x_{i+1}, \dots, x_{n+1}) =$	
$\llbracket_{h \curvearrowright (i,n), \vartheta_{n+1}(h,s,i,\delta)}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1}) \cdot \delta$	SI2b

$\mathcal{H}$ ,  $s$  stands for an arbitrary control state from  $S$ ,  $a$  stands for an arbitrary action constant that is not of the form  $\text{cr}(d)$  or  $\overline{\text{cr}}(d)$ , and  $d$  stands for an arbitrary datum  $d$  from  $D$ .

Axiom SI2 expresses that, in the event of inactiveness of the process whose turn it is, the whole becomes inactive immediately. A plausible alternative is that, in the event of inactiveness of the process whose turn it is, the whole becomes inactive only after all other processes have terminated or become inactive. In that case, the functions  $\vartheta_n : \mathcal{H} \times S \times \{1, \dots, n\} \times \mathbf{A}_\epsilon \rightarrow S$  must be extended to functions  $\vartheta_n : \mathcal{H} \times S \times \{1, \dots, n\} \times (\mathbf{A}_\epsilon \cup \{\delta\}) \rightarrow S$  and axiom SI2 must be replaced by the axioms in Table 4.

In  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}$ , i.e.  $\text{ACP}_\epsilon + \text{SI}$  extended with guarded recursion in the way described in Section 2.2, the processes that can be created are restricted to the ones denotable by a closed  $\text{ACP}_\epsilon + \text{SI}$  term. This restriction stems from the requirement that  $\phi$  is a function from  $D$  to the set of all closed  $\text{ACP}_\epsilon + \text{SI}$  terms. The restriction can be removed by relaxing this requirement to the requirement that  $\phi$  is a function from  $D$  to the set of all closed  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}$  terms. We write  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}^+$  for the theory resulting from this relaxation. In other words,  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}^+$  differs from  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}$  in that it is assumed that a fixed but arbitrary function  $\phi : D \rightarrow P$ , where  $P$  is the set of all closed terms over the signature of  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}$ , has been given.

It is customary to associate transition systems with closed terms of the language of an ACP-like algebraic theory of processes by means of structural operational semantics and to use this to construct a model in which closed terms are identified if their associated transition systems are bisimilar. The structural operational semantics of  $\text{ACP}_\epsilon$  can be found in [1]. The additional transition rules for the strategic interleaving operators and the positional strategic interleaving operators are given in Appendix A.

### 3.2 Properties about the Connection between $\text{ACP}_\epsilon$ and $\text{ACP}_\epsilon + \text{SI}$

In this section, we present some theorems concerning the connection between  $\text{ACP}_\epsilon$  and  $\text{ACP}_\epsilon + \text{SI}$ . Each of the theorems refers to more than one process algebra. It is implicit that the same set  $\mathbf{A}$  of actions and the same communication function  $\gamma$  are assumed in the process algebras referred to.

Each guarded recursive specification over  $\text{ACP}_\epsilon + \text{SI}$  can be reduced to a guarded recursive specification over  $\text{ACP}_\epsilon$ .

**Theorem 1 (Reduction).** *For each guarded recursive specification  $E$  over  $\text{ACP}_\epsilon + \text{SI}$  and each  $X \in V(E)$ , there exists a guarded recursive specification*

$E'$  over  $\text{ACP}_\epsilon$  such that  $\langle X|E \rangle = \langle X|E' \rangle$  is derivable from the axioms of  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}$ .

Each closed  $\text{ACP}_\epsilon + \text{SI}$  term is derivably equal to a closed  $\text{ACP}_\epsilon$  term.

**Theorem 2 (Elimination).**

1. For each closed  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}^+$  term  $t$ , there exists a closed  $(\text{ACP}_\epsilon)_{\text{rec}}$  term  $t'$  such that  $t = t'$  is derivable from the axioms of  $(\text{ACP}_\epsilon + \text{SI})_{\text{rec}}^+$ .
2. For each closed  $\text{ACP}_\epsilon + \text{SI}$  term  $t$ , there exists a closed  $\text{ACP}_\epsilon$  term  $t'$  such that  $t = t'$  is derivable from the axioms of  $\text{ACP}_\epsilon + \text{SI}$ .

Each equation between closed  $\text{ACP}_\epsilon$  terms that is derivable in  $\text{ACP}_\epsilon + \text{SI}$  is also derivable in  $\text{ACP}_\epsilon$ .

**Theorem 3 (Conservative extension).** For each two closed  $\text{ACP}_\epsilon$  terms  $t$  and  $t'$ ,  $t = t'$  is derivable from the axioms of  $\text{ACP}_\epsilon + \text{SI}$  only if  $t = t'$  is derivable from the axioms of  $\text{ACP}_\epsilon$ .

The following theorem concerns the expansion of minimal models of  $\text{ACP}_\epsilon$  to models of  $\text{ACP}_\epsilon + \text{SI}$ .

**Theorem 4 (Unique expansion).** Each minimal model of  $\text{ACP}_\epsilon$  has a unique expansion to a model of  $\text{ACP}_\epsilon + \text{SI}$ .

The proofs of Theorems 1, 2.1, 3, and 4 go along the same line as the proofs of the corresponding theorems in [3]. Theorem 2.2 is a corollary of the proof of Theorem 2.1. Theorem 1 would not go through if guarded recursive specifications were required to be finite.

## 4 An Example

In this section, we instantiate the generic interleaving strategy on which  $\text{ACP} + \text{SI}$  is based with a specific interleaving strategy. The interleaving strategy concerned corresponds to the round-robin scheduling algorithm, where each of the processes being interleaved is given a fixed number  $k$  of consecutive turns, adapted to mutual exclusion of critical subprocesses of the different processes being interleaved. Mutual exclusion of certain subprocesses is the condition that they are not interleaved and critical subprocesses are subprocesses that possibly interfere with each other when this condition is not met. The adopted mechanism for mutual exclusion is essentially a binary semaphore mechanism [5,4,2]. Below binary semaphores are simply called *semaphores*.

In this section, it is assumed that a fixed but arbitrary natural number  $k \in \mathbb{N}_1$  has been given. We use  $k$  as the number of consecutive turns that each process being interleaved gets.

Moreover, it is assumed that a finite set  $R$  of semaphores has been given. We instantiate the set  $C$  of control actions as follows:

$$C = \{\text{wait}(r) \mid r \in R\} \cup \{\text{signal}(r) \mid r \in R\} ,$$

hereby taking for granted that  $C$  satisfies the necessary conditions. The **wait** and **signal** actions correspond to the **P** and **V** operations from [5].

We instantiate the set  $S$  of control states as follows:

$$S = \bigcup_{R' \subseteq R} (R' \rightarrow \mathbb{N}_1^*) .$$

The intuition concerning the connection between control states  $s \in S$  and the semaphore mechanism as introduced in [5] is as follows:

- $r \notin \text{dom}(s)$  indicates that semaphore  $r$  has the value 1;
- $r \in \text{dom}(s)$  indicates that semaphore  $r$  has the value 0;
- $r \in \text{dom}(s)$  and  $s(r) = \langle \rangle$  indicates that no process is suspended on semaphore  $r$ ;
- if  $r \in \text{dom}(s)$  and  $s(r) \neq \langle \rangle$ , then  $s(r)$  represents a first-in, first-out queue of processes suspended on  $r$ .

As a preparation for the instantiation of the abstract schedulers  $\sigma_n$  and control state transformers  $\vartheta_n$ , we define some auxiliary functions.

We define a total function  $\text{turns} : \mathcal{H} \times \mathbb{N}_1 \rightarrow \mathbb{N}$  recursively as follows:

$$\begin{aligned} \text{turns}(\langle \rangle, i) &= 0 , \\ \text{turns}(h \curvearrowright (j, n), i) &= 0 && \text{if } i \neq j , \\ \text{turns}(h \curvearrowright (j, n), i) &= \text{turns}(h, i) + 1 && \text{if } i = j . \end{aligned}$$

If  $\text{turns}(h, i) = l$  and  $l > 0$ , then the interleaving history  $h$  ends with  $l$  consecutive turns of the  $i$ th process being interleaved. If  $\text{turns}(h, i) = 0$ , then the interleaving history  $h$  does not end with turns of the  $i$ th process being interleaved.

For each  $n \in \mathbb{N}_1$ , we define a total function  $\text{next}_n : \mathcal{H} \times \mathbb{N} \rightarrow \{1, \dots, n\}$  by cases as follows:

$$\begin{aligned} \text{next}_n(\langle \rangle, i) &= i + 1 , \\ \text{next}_n(h \curvearrowright (j, n), i) &= j && \text{if } \text{turns}(h, j) < k , \\ \text{next}_n(h \curvearrowright (j, n), i) &= ((i + j) \bmod n) + 1 && \text{if } \text{turns}(h, j) \geq k . \end{aligned}$$

If  $\text{next}_n(h, i) = j$ , then the  $j$ th process being interleaved is the process that should get the  $(i+1)$ th next turn after interleaving history  $h$  according to the round-robin scheduling algorithm, where each of the processes being interleaved is given  $k$  consecutive turns.

We define a total function  $\text{waiting} : S \rightarrow \mathcal{P}(\mathbb{N}_1)$  as follows:

$$\text{waiting}(s) = \bigcup_{r \in \text{dom}(s)} \text{elems}(s(r)) .$$

If  $\text{waiting}(s) = I$ , then  $i \in I$  iff the  $i$ th process being interleaved is suspended on one or more semaphores in control state  $s$ .

For each  $n \in \mathbb{N}_1$ , we define a partial function  $\text{sched}_n : \mathcal{H} \times S \times \mathbb{N} \rightarrow \{1, \dots, n\}$  recursively as follows:

$$\begin{aligned} \text{sched}_n(h, s, i) &= \text{next}_n(h, i) && \text{if } i < k \cdot n \wedge \text{next}_n(h, i) \notin \text{waiting}(s) , \\ \text{sched}_n(h, s, i) &= \text{sched}_n(h, s, i + 1) && \text{if } i < k \cdot n \wedge \text{next}_n(h, i) \in \text{waiting}(s) . \end{aligned}$$

The function  $sched_n$  is like the function  $next_n$ , but skips the processes that are suspended on one or more semaphores according to the control state  $s$ . Notice that  $sched_n(h, s, i)$  is undefined if  $waiting(s) = \{1, \dots, n\}$ . In this case, none of the processes being interleaved can be given a turn and the whole becomes inactive.

We define a total function  $remove_n: S \times \{1, \dots, n\} \rightarrow S$  recursively as follows:<sup>4</sup>

$$\begin{aligned} remove_n([], i) &= [] , \\ remove_n(s \dagger [r \mapsto q], i) &= remove_n(s, i) \dagger [r \mapsto remove'_n(q, i)] , \end{aligned}$$

where the total function  $remove'_n: \mathbb{N}_1^* \times \{1, \dots, n\} \rightarrow \mathbb{N}_1^*$  is recursively defined as follows:

$$\begin{aligned} remove'_n(\langle \rangle, i) &= \langle \rangle , \\ remove'_n(j \frown q, i) &= j \frown remove'_n(q, i) && \text{if } j < i , \\ remove'_n(j \frown q, i) &= remove'_n(q, i) && \text{if } j = i , \\ remove'_n(j \frown q, i) &= (j - 1) \frown remove'_n(q) && \text{if } j > i . \end{aligned}$$

If  $remove_n(s, i) = s'$ , then  $s'$  is  $s$  adapted to the successful termination of the  $i$ th process of the processes being interleaved.

For each  $n \in \mathbb{N}_1$ , we instantiate the abstract scheduler  $\sigma_n$  and control state transformer  $\vartheta_n$  as follows:

$$\begin{aligned} \sigma_n(h, s) &= sched_n(h, s, 0) , \\ \vartheta_n(\langle \rangle, s, i, a) &= [] && \text{if } a \notin C , \\ \vartheta_n(h \frown (j, n), s, i, a) &= s && \text{if } a \notin C , \\ \vartheta_n(\langle \rangle, s, i, \text{wait}(r)) &= [r \mapsto \langle \rangle] , \\ \vartheta_n(h \frown (j, n), s, i, \text{wait}(r)) &= s \dagger [r \mapsto \langle \rangle] && \text{if } r \notin \text{dom}(s) , \\ \vartheta_n(h \frown (j, n), s, i, \text{wait}(r)) &= s \dagger [r \mapsto s(r) \frown i] && \text{if } r \in \text{dom}(s) , \\ \vartheta_n(\langle \rangle, s, i, \text{signal}(r)) &= [] , \\ \vartheta_n(h \frown (j, n), s, i, \text{signal}(r)) &= s && \text{if } r \notin \text{dom}(s) , \\ \vartheta_n(h \frown (j, n), s, i, \text{signal}(r)) &= s \Leftarrow \{r\} && \text{if } r \in \text{dom}(s) \wedge s(r) = \langle \rangle , \\ \vartheta_n(h \frown (j, n), s, i, \text{signal}(r)) &= s \dagger [r \mapsto \text{tl}(s(r))] && \text{if } r \in \text{dom}(s) \wedge s(r) \neq \langle \rangle , \\ \vartheta_n(h, s, i, \epsilon) &= remove_n(s, i) . \end{aligned}$$

The following clarifies the connection between the instantiated control state transformers  $\vartheta_n$  and the semaphore mechanism as introduced in [5]:

- $s = []$  indicates that all semaphores have value 1;
- if  $r \notin \text{dom}(s)$ , then the transition from  $s$  to  $s \dagger [r \mapsto \langle \rangle]$  indicates that the value of semaphore  $r$  changes from 1 to 0;

<sup>4</sup> The special function notation used below is explained in Appendix B.

- if  $r \in \text{dom}(s)$ , then the transition from  $s$  to  $s \dagger [r \mapsto s(r) \sim i]$  indicates that the  $i$ th process being interleaved is added to the queue of suspended processes;
- if  $r \notin \text{dom}(s)$ , then the transition from  $s$  to  $s$  indicates that the value of semaphore  $r$  remains 1;
- if  $r \in \text{dom}(s)$  and  $s(r) = \langle \rangle$ , then the transition from  $s$  to  $s \triangleleft \{r\}$  indicates that the value of semaphore  $r$  changes from 0 to 1;
- if  $r \in \text{dom}(s)$  and  $s(r) \neq \langle \rangle$ , then the transition from  $s$  to  $s \dagger [r \mapsto \text{tl}(s(r))]$  indicates that the first process in the queue of suspended processes is removed from that queue.

All respects in which the generic interleaving strategy of ACP+SI are widened appear to be indispensable for the instantiation presented in this section.

## 5 Concluding Remarks

In a previous paper, we have extended the algebraic theory of processes known as ACP with strategic interleaving, i.e. interleaving according to some process-scheduling policy. The extension concerned is based on a generic interleaving strategy that can be instantiated with different specific interleaving strategies. In the current paper, we have extended the variant of ACP known as  $\text{ACP}_\epsilon$  with strategic interleaving and widened the generic interleaving strategy in three respects. For the widening in one of these respects, the setting of ACP is unfit. We have instantiated the widened generic interleaving strategy with a specific interleaving strategy that supports mutual exclusion of critical subprocesses of the different processes being interleaved. This instantiation provides evidence of the desirability of the widening of the generic interleaving strategy.

## A Structural Operational Semantics of $\text{ACP}_\epsilon + \text{SI}$

It is customary to associate transition systems with closed terms of the language of an ACP-like algebraic theory about processes by means of structural operational semantics and to use this to construct a model in which closed terms are identified if their associated transition systems are bisimilar. The structural operational semantics of  $\text{ACP}_\epsilon$  can be found in [1]. The additional transition rules for the strategic interleaving operators and the positional strategic interleaving operators are given in Table 5. In this table,  $n$  and  $i$  stand for arbitrary numbers from  $\mathbb{N}_1$  with  $i \leq n$ ,  $h$  stands for an arbitrary interleaving history from  $\mathcal{H}$ ,  $s$  stands for an arbitrary control state from  $S$ ,  $a$  stands for an arbitrary action constant that is not of the form  $\text{cr}(d)$  or  $\overline{\text{cr}}(d)$ ,  $\alpha$  stands for an arbitrary action constant, and  $d$  stands for an arbitrary datum  $d$  from  $D$ . The intuition concerning  $\downarrow$  and  $\xrightarrow{a}$  is as follows:

- $t \downarrow$  indicates that  $t$  is capable of terminating successfully;
- $t \xrightarrow{a} t'$  indicates that  $t$  is capable of performing action  $a$  and then proceeding as  $t'$ .

**Table 5.** Transition rules for strategic interleaving

---

$\frac{x \downarrow}{\Downarrow_{h,s}^1(x) \downarrow} \sigma_1(h, s) = 1$	
$\frac{x_i \downarrow, \Downarrow_{h \curvearrowright (i,n), \vartheta_{n+1}(h,s,i,\epsilon)}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1}) \downarrow}{\Downarrow_{h,s}^{n+1}(x_1, \dots, x_{n+1}) \downarrow} \sigma_n(h, s) = i$	
$\frac{x_i \xrightarrow{a} x'_i}{\Downarrow_{h,s}^n(x_1, \dots, x_n) \xrightarrow{a} \Downarrow_{h \curvearrowright (i,n), \vartheta_n(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)} \sigma_n(h, s) = i, a \notin C$	
$\frac{x_i \xrightarrow{a} x'_i}{\Downarrow_{h,s}^n(x_1, \dots, x_n) \xrightarrow{\bar{a}} \Downarrow_{h \curvearrowright (i,n), \vartheta_n(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)} \sigma_n(h, s) = i, a \in C$	
$\frac{x_i \xrightarrow{\text{cr}(d)} x'_i}{\Downarrow_{h,s}^n(x_1, \dots, x_n) \xrightarrow{\overline{\text{cr}(d)}} \Downarrow_{h \curvearrowright (i,n+1), \vartheta_n(h,s,i,\text{cr}(d))}^{n+1}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n, \phi(d))} \sigma_n(h, s) = i$	
$\frac{x_i \downarrow, \Downarrow_{h \curvearrowright (i,n), \vartheta_{n+1}(h,s,i,\epsilon)}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1}) \xrightarrow{\alpha} x'}{\Downarrow_{h,s}^{n+1}(x_1, \dots, x_{n+1}) \xrightarrow{\alpha} x'} \sigma_n(h, s) = i$	
$\frac{x \downarrow}{\Downarrow_{h,s}^{1,i}(x) \downarrow}$	
$\frac{x_i \downarrow, \Downarrow_{h \curvearrowright (i,n), \vartheta_{n+1}(h,s,i,\epsilon)}^{n,i}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1}) \downarrow}{\Downarrow_{h,s}^{n+1,i}(x_1, \dots, x_{n+1}) \downarrow}$	
$\frac{x_i \xrightarrow{a} x'_i}{\Downarrow_{h,s}^{n,i}(x_1, \dots, x_n) \xrightarrow{a} \Downarrow_{h \curvearrowright (i,n), \vartheta_n(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)} a \notin C$	
$\frac{x_i \xrightarrow{a} x'_i}{\Downarrow_{h,s}^{n,i}(x_1, \dots, x_n) \xrightarrow{\bar{a}} \Downarrow_{h \curvearrowright (i,n), \vartheta_n(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)} a \in C$	
$\frac{x_i \xrightarrow{\text{cr}(d)} x'_i}{\Downarrow_{h,s}^{n,i}(x_1, \dots, x_n) \xrightarrow{\overline{\text{cr}(d)}} \Downarrow_{h \curvearrowright (i,n+1), \vartheta_n(h,s,i,\text{cr}(d))}^{n+1}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n, \phi(d))}$	
$\frac{x_i \downarrow, \Downarrow_{h \curvearrowright (i,n), \vartheta_{n+1}(h,s,i,\epsilon)}^{n,i}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1}) \xrightarrow{\alpha} x'}{\Downarrow_{h,s}^{n+1,i}(x_1, \dots, x_{n+1}) \xrightarrow{\alpha} x'}$	

---

The transition rules for the strategic interleaving operators are similar to the transition rules for the positional strategic interleaving operators. However, each transition rule for the strategic interleaving operators has the side-condition  $\sigma_n(h, s) = i$ .

## B Sequence Notation and Function Notation

We use the following sequence notation:

- $\langle \rangle$  for the empty sequence;
- $d$  for the sequence having  $d$  as sole element;
- $u \sim v$  for the concatenation of sequences  $u$  and  $v$ ;

- $\text{hd}(u)$  for the first element of non-empty sequence  $u$ ;
- $\text{tl}(u)$  for the subsequence of non-empty sequence  $u$  whose first element is the second element of  $u$  and whose last element is the last element of  $u$ ;
- $\text{elems}(u)$  is the set of all elements of sequence  $u$ .

We use the following special function notation:

- $[\ ]$  for the empty function;
- $[d \mapsto e]$  for the function  $f$  with  $\text{dom}(f) = \{d\}$  such that  $f(d) = e$ ;
- $f \dagger g$  for the function  $h$  with  $\text{dom}(h) = \text{dom}(f) \cup \text{dom}(g)$  such that for all  $d \in \text{dom}(h)$ ,  $h(d) = f(d)$  if  $d \notin \text{dom}(g)$  and  $h(d) = g(d)$  otherwise;
- $f \triangleleft S$  for the function  $g$  with  $\text{dom}(g) = \text{dom}(f) \setminus S$  such that for all  $d \in \text{dom}(g)$ ,  $g(d) = f(d)$ .

## References

1. Baeten, J.C.M., Weijland, W.P.: Process Algebra, Cambridge Tracts in Theoretical Computer Science, vol. 18. Cambridge University Press, Cambridge (1990)
2. Ben-Ari, M.: Principles of Concurrent and Distributed Programming. Pearson, Harlow, second edn. (2006)
3. Bergstra, J.A., Middelburg, C.A.: Process algebra with strategic interleaving. *Theory of Computing Systems* 63(3), 488–505 (2019)
4. Brinch Hansen, P.: Operating System Principles. Prentice-Hall, Englewood Cliffs, NJ (1973)
5. Dijkstra, E.W.: Cooperating sequential processes. In: Genuys, F. (ed.) *Programming Languages*. pp. 43–112. Academic Press (1968)
6. van Glabbeek, R.J., Vaandrager, F.W.: Modular specification of process algebras. *Theoretical Computer Science* 113(2), 293–348 (1993)
7. Gosling, J., Joy, B., Steele, G., Bracha, G.: *The Java Language Specification*. Addison-Wesley, Reading, MA, second edn. (2000)
8. Hejlsberg, A., Wiltamuth, S., Golde, P.: *C# Language Specification*. Addison-Wesley, Reading, MA (2003)
9. Hoare, C.A.R.: *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs (1985)
10. Middelburg, C.A.: Probabilistic process algebra and strategic interleaving. [arXiv:1912.10041v2](https://arxiv.org/abs/1912.10041v2) [cs.LG] (December 2019)
11. Milner, R.: *Communication and Concurrency*. Prentice-Hall, Englewood Cliffs (1989)
12. Sabelfeld, A., Sands, D.: Probabilistic noninterference for multi-threaded programs. In: *Computer Security Foundations Workshop 2000*. pp. 200–214. IEEE Computer Society Press (2000)