



UvA-DARE (Digital Academic Repository)

Process Algebra with Strategic Interleaving

Bergstra, J.A.; Middelburg, C.A.

DOI

[10.1007/s00224-018-9873-2](https://doi.org/10.1007/s00224-018-9873-2)

Publication date

2019

Document Version

Final published version

Published in

Theory of Computing Systems

License

CC BY

[Link to publication](#)

Citation for published version (APA):

Bergstra, J. A., & Middelburg, C. A. (2019). Process Algebra with Strategic Interleaving. *Theory of Computing Systems*, 63(3), 488–505 . <https://doi.org/10.1007/s00224-018-9873-2>


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, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

Process Algebra with Strategic Interleaving

J. A. Bergstra¹ · C. A. Middelburg¹ 

Published online: 8 August 2018
© The Author(s) 2018

Abstract In process algebras such as ACP (Algebra of Communicating Processes), parallel processes are considered to be interleaved in an arbitrary way. In the case of multi-threading as found in contemporary programming languages, parallel processes are actually interleaved according to some interleaving strategy. An interleaving strategy is what is called a process-scheduling policy in the field of operating systems. In many systems, for instance hardware/software systems, we have to do with both parallel processes that may best be considered to be interleaved in an arbitrary way and parallel processes that may best be considered to be interleaved according to some interleaving strategy. Therefore, we extend ACP in this paper with the latter form of interleaving. The established properties of the extension concerned include an elimination property, a conservative extension property, and a unique expansion property.

Keywords Process algebra · Arbitrary interleaving · Strategic interleaving · Abstract scheduler · Interleaving history

✉ C. A. Middelburg
C.A.Middelburg@uva.nl

J. A. Bergstra
J.A.Bergstra@uva.nl

¹ Informatics Institute, Faculty of Science, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands

1 Introduction

In algebraic theories of processes, such as ACP [5, 7], CCS [18, 21] and CSP [12, 19], processes are discrete behaviours that proceed by doing steps in a sequential fashion. The parallel composition of two processes is usually considered to incorporate all conceivable interleavings of their steps. In each interleaving, the steps of both processes occur in some order where each time one step is taken from either of the processes. According to many, this interpretation of parallel composition, called arbitrary interleaving, is a plausible, general, if not idealized interpretation. Underlying the usual justification of this claim is the assumption that at most one step is done at each point in time. However, others contend that interpretations in which this simplifying assumption is fulfilled are not faithful. Be that as it may, arbitrary interleaving turns out to be appropriate for many applications and to facilitate formal algebraic reasoning.

Multi-threading as found in programming languages such as Java [16] and C# [17], gives rise to parallel composition of processes. In the case of multi-threading, however, the steps of the processes concerned are interleaved according to a process-scheduling policy. We use the term strategic interleaving for this more constrained form of interleaving; and we further use the term interleaving strategy instead of process-scheduling policy. Arbitrary interleaving and strategic interleaving are quite different. The following points illustrate this: (a) whether the interleaving of certain processes leads to inactiveness depends on the interleaving strategy used; (b) sometimes inactiveness occurs with a particular interleaving strategy whereas arbitrary interleaving would not lead to inactiveness and vice versa.

In previous work, we studied strategic interleaving in the setting of thread algebra, which is built on a specialized algebraic theory of processes devoted to the behaviours produced by instruction sequences under execution (see e.g. [8–10]). We have, for instance, given demonstrations of points (a) and (b) above in this setting. 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. This is what motivated us to do the work presented in this paper, namely extending ACP such that it supports both arbitrary interleaving and strategic interleaving.

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. In the work presented in this paper, we consider strategic interleaving where process creation is taken into account. The approach to process creation followed in this paper originates from the one first followed in [6] to extend ACP with process creation and later followed in [2, 3, 11] to extend different timed versions of ACP with process creation. The only other approach that we know of is the approach, based on [1], that has for instance been followed in [4, 14]. However, with that approach, it is most unlikely that data about the creation of processes can be made available for the decision making concerning the strategic interleaving of processes.

The extension of ACP presented in this paper covers a generic interleaving strategy that can be instantiated with different specific interleaving strategies. We found

two plausible ways to deal with inactiveness of a process whose steps are being interleaved with steps of other processes in the case of strategic interleaving. This gives rise to two plausible extensions of ACP. We will treat only one of them in detail.

The rest of this paper is organized as follows. In Section 2, we review ACP (Section 2.1), guarded recursion in the setting of ACP (Section 2.2), and some relevant results about the latter (Section 2.3). In Section 3, we extend ACP with strategic interleaving (Section 3.1) and establish some important properties of the extension (Section 3.2). In Section 4, we make some concluding remarks.

2 ACP with Guarded Recursion

In this section, we give a survey of ACP (Algebra of Communicating Processes), guarded recursion in the setting of ACP, and some relevant results about the extension of ACP with guarded recursion. For a comprehensive overview, the reader is referred to [5, 13].

2.1 ACP

In ACP, it is assumed that a fixed but arbitrary set A of *actions*, with $\delta \notin A$, has been given. We write A_δ for $A \cup \{\delta\}$. 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 γ is regarded to give the result of synchronously performing any two actions for which this is possible, and to give δ otherwise.

The signature of ACP consists of the following constants and operators:

- for each $a \in A$, the *action* constant a ;
- the *inaction* constant δ ;
- the binary *alternative composition* operator $_+_-$;
- the binary *sequential composition* operator $_ \cdot _$;
- the binary *parallel composition* operator $_ \| _$;
- the binary *left merge* operator $_ \parallel _$;
- the binary *communication merge* operator $_ | _$;
- for each $H \subseteq A$, the unary *encapsulation* operator ∂_H .

We assume that there are infinitely many variables, including x, y, z . Terms are built as usual. We use infix notation for the binary operators. The precedence conventions used with respect to the operators of ACP are as follows: $+$ binds weaker than all others, \cdot binds stronger than all others, and the remaining operators bind equally strong.

The constants and operators of ACP can be explained as follows:

- the constant a denotes the process that is only capable of first performing action a and next terminating successfully;
- the constant δ denotes the process that is not capable of doing anything;

- a closed term of the form $t+t'$ denotes the process that behaves either as the process denoted by t or as the process denoted by t' , but not both;
- a closed term of the form $t \cdot t'$ denotes the process that first behaves as the process denoted by t and on successful termination of that process it next behaves as the process denoted by t' ;
- a closed term of the form $t \parallel t'$ denotes the process that behaves as the process that proceeds with the processes denoted by t and t' in parallel;
- a closed term of the form $t \parallel\!\!| t'$ denotes the process that behaves the same as the process denoted by $t \parallel t'$, except that it starts with performing an action of the process denoted by t ;
- a closed term of the form $t|t'$ denotes the process that behaves the same as the process denoted by $t \parallel t'$, except that it starts with performing an action of the process denoted by t and an action of the process denoted by t' synchronously;
- a closed term of the form $\partial_H(t)$ denotes the process that behaves the same as the process denoted by t , except that actions from H are blocked.

The operators \parallel and $|$ are of an auxiliary nature. They are needed to axiomatize ACP.

The axioms of ACP are the equations given in Table 1. In these equations, a, b and c stand for arbitrary constants of ACP, 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 other presentations of ACP, $\gamma(a, b)$ is regularly replaced by $a|b$ in CM5–CM7. By CM12, which is more often called CF, these replacements give rise to an equivalent axiomatization. In other presentations of ACP, CM10 and CM11 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, CM10 and CM11 hold in virtually all models of ACP that have been devised.

Table 1 Axioms of ACP

$x + y = y + x$	A1	$x \parallel y = x \parallel\!\! y + y \parallel\!\! x + x \parallel\!\! y$	CM1
$(x + y) + z = x + (y + z)$	A2	$a \parallel\!\! x = a \cdot x$	CM2
$x + x = x$	A3	$a \cdot x \parallel\!\! y = a \cdot (x \parallel\!\! y)$	CM3
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4	$(x + y) \parallel\!\! z = x \parallel\!\! z + y \parallel\!\! z$	CM4
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5	$a \cdot x \mid b = \gamma(a, b) \cdot x$	CM5
$x + \delta = x$	A6	$a \mid b \cdot x = \gamma(a, b) \cdot x$	CM6
$\delta \cdot x = \delta$	A7	$a \cdot x \mid b \cdot y = \gamma(a, b) \cdot (x \parallel\!\! y)$	CM7
		$(x + y) \mid z = x \mid z + y \mid z$	CM8
$\partial_H(a) = a$ if $a \notin H$	D1	$x \mid (y + z) = x \mid y + x \mid z$	CM9
$\partial_H(a) = \delta$ if $a \in H$	D2	$\delta \mid x = \delta$	CM10
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3	$x \mid \delta = \delta$	CM11
$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$	D4	$a \mid b = \gamma(a, b)$	CM12

Table 2 Axioms for guarded recursion

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

In the sequel, we will use the sum notation $\sum_{i < n} t_i$. For each $i \in \mathbb{N}$, let t_i be a term of ACP or an extension of ACP. Then $\sum_{i < 0} t_i = \delta$ and, for each $n \in \mathbb{N}_1$,¹ the term $\sum_{i < n} t_i$ is defined by induction on n as follows: $\sum_{i < 1} t_i = t_0$ and $\sum_{i < n+1} t_i = \sum_{i < n} t_i + t_n$.

2.2 Guarded Recursion

A closed ACP 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.

Let T be ACP or a concrete extensions of ACP,² and let t be a T 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 T term containing this occurrence of X .

Let T be ACP or a concrete extension of ACP. Then a T term t is a *guarded T term* if all occurrences of variables in t are guarded.

Let T be ACP or a concrete extension of ACP. Then a *guarded recursive specification* over T is a finite or countably infinite set of recursion equations $E = \{X = t_X \mid X \in V\}$, where V is a set of variables and each t_X is either a guarded T term in which variables other than the variables from V do not occur or a T term rewritable to such a term using the axioms of T in either direction and/or the equations in E , except the equation $X = t_X$, from left to right. We write $V(E)$ for the set of all variables that occur in E . A solution of E in some model of T is a set $\{P_X \mid X \in V(E)\}$ of elements of the carrier of that model such that the equations of E hold if, for all $X \in V(E)$, X is assigned P_X . We are only interested models of ACP and concrete extensions of ACP in which guarded recursive specifications have unique solutions.

Let T be ACP or a concrete extension of ACP. We extend T with guarded recursion by adding constants for solutions of guarded recursive specifications over T and axioms concerning these additional constants. For each guarded recursive specification E over T and each $X \in V(E)$, we add a constant standing for the unique solution of E for X to the constants of T . The constant standing for the unique solution of E for X is denoted by $\langle X|E \rangle$. We use the following notation. Let t be a T term and E be a guarded recursive specification. Then we write $\langle t|E \rangle$ for t with, for all $X \in V(E)$, all occurrences of X in t replaced by $\langle X|E \rangle$. We add the equation RDP and the conditional equation RSP given in Table 2 to the axioms of T . In RDP and RSP, X stands

¹We write \mathbb{N}_1 for the set $\{n \in \mathbb{N} \mid n \geq 1\}$ of positive natural numbers.

²A concrete extension of ACP is an extension of ACP that does not offer the possibility of abstraction from certain actions. All extensions of ACP introduced in this paper are concrete extensions.

for an arbitrary variable, t_X stands for an arbitrary T term, and E stands for an arbitrary guarded recursive specification over T . Side conditions restrict what X , t_X and E stand for. We write T_{rec} for the resulting theory.

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

In extensions of ACP 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 whose axioms include RSP. A complete inference system for infinitary conditional equational logic can be found in, for example, [15]. It is noteworthy that derivations are allowed to be of countably infinite length in infinitary conditional equational logic.

2.3 Some Results about Guarded Recursion

This section is concerned with legitimate ways of manipulating guarded recursive specifications and with guarded terms of a special form.

Let T be ACP or a concrete extension of ACP. Then, each guarded recursive specification over T can be manipulated in several ways that are justified by RDP and RSP.

Proposition 1 (Manipulation) *Let T be ACP or a concrete extension of ACP. Then, for all guarded recursive specifications E over T , for all $X \in \mathbf{V}(E)$:*

- (1) *if $Y = t_Y \in E$ and $t_Y = t'_Y$ is derivable from the axioms of T , then $\langle X|E \rangle = \langle X|(E \setminus \{Y = t_Y\}) \cup \{Y = t'_Y\}$ is derivable from the axioms of T , RDP and RSP;*
- (2) *if $Y = t_Y \in E$, $Z = t_Z \in E$, and t'_Y is t_Y with some occurrence of Z in t_Y replaced by t_Z , then $\langle X|E \rangle = \langle X|(E \setminus \{Y = t_Y\}) \cup \{Y = t'_Y\}$ is derivable from the axioms of T , RDP and RSP;*
- (3) *if $Y \notin \mathbf{V}(E)$ and t_Y is a guarded T term in which variables other than the variables from $\mathbf{V}(E)$ do not occur, then $\langle X|E \rangle = \langle X|E \cup \{Y = t_Y\}$ is derivable from the axioms of T , RDP and RSP.*

Proof In case (1), first we apply RDP for each recursion equation in E , next we apply $t_Y = t'_Y$ to $\langle Y|E \rangle = \langle t_Y|E \rangle$, and finally we apply RSP to the resulting set of equations. In case (2), first we apply RDP for each recursion equation in E , next we apply $\langle Z|E \rangle = \langle t_Z|E \rangle$ to $\langle Y|E \rangle = \langle t_Y|E \rangle$, and finally we apply RSP to the resulting set of equations. In case (3), we first apply RDP for each recursion equation in $E \cup \{Y = t_Y\}$ and then we apply RSP to the resulting set of equations.³ \square

Proposition 1 will be used in the proof of Theorem 1 in Section 3.2.

³Further details on cases (1) and (2) can be found in the proof of Theorem 4.3.2 from [13].

Let T be ACP or a concrete extension of ACP. Then the set HNF of *head normal forms* of T is inductively defined by the following rules:

- $\delta \in HNF$;
- if $a \in \mathbf{A}$, then $a \in HNF$;
- if $a \in \mathbf{A}$ and t is a T term, then $a \cdot t \in HNF$;
- if $t, t' \in HNF$, then $t+t' \in HNF$.

Each head normal form of T is derivably equal to a head normal form of the form $\sum_{i < n} a_i \cdot t_i + \sum_{j < m} b_j$, where $n, m \in \mathbb{N}$, for all $i \in \mathbb{N}$ with $i < n$, $a_i \in \mathbf{A}$ and t_i is a T term, and, for all $j \in \mathbb{N}$ with $j < m$, $b_j \in \mathbf{A}$.

It is well-known that each guarded ACP_{rec} term is derivably equal to a head normal form of ACP_{rec} (see also Lemma 2.4.7 in [5]).

Proposition 2 (Head normal form) *For each guarded ACP_{rec} term t , there exists a head normal form t' of ACP such that $t = t'$ is derivable from the axioms of ACP_{rec} .*

Proof The proof is analogous to the proof of Proposition 3 in Section 3.2. □

3 Strategic Interleaving

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

3.1 ACP 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:

- $\langle \rangle \in \mathcal{H}$;
- 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 $h \frown (i, n) \frown (j, m) \in \mathcal{H}$.⁴

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

⁴We write $\langle \rangle$ for the empty sequence, d for the sequence having d as sole element, and $\alpha \frown \alpha'$ for the concatenation of sequences α and α' . We assume that the usual identities, such as $\langle \rangle \frown \alpha = \alpha$ and $(\alpha \frown \alpha') \frown \alpha'' = \alpha \frown (\alpha' \frown \alpha'')$, hold.

be interleaved may increase due to process creation (introduced below) and decrease due to successful termination of processes.

The presented extension of ACP is called ACP+SI (ACP with Strategic Interleaving). It covers 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+SI, it is assumed that the following has been given:

- a fixed but arbitrary set S ;
- for each $n \in \mathbb{N}_1$, a fixed but arbitrary function $\sigma_n: \mathcal{H} \times S \rightarrow \{1, \dots, n\}$;
- for each $n \in \mathbb{N}_1$, a fixed but arbitrary function $\vartheta_n: \mathcal{H} \times S \times \{1, \dots, n\} \times \mathbf{A} \rightarrow S$.

The elements of S are called *control states*, σ_n is called an *abstract scheduler* (for n processes), and ϑ_n is called a *control state transformer* (for n processes). The intuition concerning S , σ_n , and ϑ_n is as follows:

- the control states from S encode data that are relevant to the interleaving strategy, but not derivable from the interleaving history;
- if $\sigma_n(h, s) = i$, then the i th 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 .

Thus, S , $\langle \sigma_n \rangle_{n \in \mathbb{N}_1}$, and $\langle \vartheta_n \rangle_{n \in \mathbb{N}_1}$ make up a way to represent an interleaving strategy. This way to represent an interleaving strategy is engrafted on [22].

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

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

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

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

In this case, the interleaving strategy corresponds to the round-robin scheduling algorithm. More advanced strategies can be obtained if the scheduling makes more advanced use of the interleaving history and the control state. The interleaving history may, for example, be used to factor the individual lifetimes of the processes to be interleaved and their creation hierarchy into the process-scheduling decision making. Individual properties of the processes to be interleaved that depend on the actions performed by them can be taken into account by making use of the control state. The control state may, for example, be used to factor the processes being interleaved that currently wait to acquire a lock from a process that manages a shared resource into the process-scheduling decision making.⁵

In ACP+SI, it is also assumed that a fixed but arbitrary 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

⁵In [8], various examples of interleaving strategies are given in the setting of the relatively unknown thread algebra. The representation of the more serious of these examples in the current setting demands nontrivial use of the control state.

signature of ACP+SI (given below), have been given and that, for each $d \in D$ and $a, b \in A$, $cr(d), \bar{cr}(d) \in A$, $\gamma(cr(d), a) = \delta$, and $\gamma(a, b) \neq cr(d)$. The action $cr(d)$ can be considered a process creation request and the action $\bar{cr}(d)$ can be considered a process creation act. They represent 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.

The signature of ACP+SI consists of the constants and operators from the signature of ACP and in addition the following operators:

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

The strategic interleaving operators can be explained as follows:

- a closed term of the form $\parallel_{h,s}^n(t_1, \dots, t_n)$ 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 $S, \langle \sigma_n \rangle_{n \in \mathbb{N}_1}$, and $\langle \vartheta_n \rangle_{n \in \mathbb{N}_1}$.

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 ACP.

The axioms of ACP+SI are the axioms of ACP and in addition the equations given in Table 3. 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 \mathcal{H} , s stands for an arbitrary control state from S , a stands for an arbitrary action constant that is not of the form $cr(d)$ or $\bar{cr}(d)$, and d stands for an arbitrary datum d from D .

Table 3 Axioms for strategic interleaving

$\parallel_{h,s}^n(x_1, \dots, x_n) = \parallel_{h,s}^{n, \sigma_n(h,s)}(x_1, \dots, x_n)$	SI1
$\parallel_{h,s}^{n,i}(x_1, \dots, x_{i-1}, \delta, x_{i+1}, \dots, x_n) = \delta$	SI2
$\parallel_{h,s}^{1,i}(a) = a$	SI3
$\parallel_{h,s}^{n+1,i}(x_1, \dots, x_{i-1}, a, x_{i+1}, \dots, x_{n+1}) =$ $a \cdot \parallel_{h \circlearrowleft (i,n), \vartheta_{n+1}(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1})$	SI4
$\parallel_{h,s}^{n,i}(x_1, \dots, x_{i-1}, a \cdot x'_i, x_{i+1}, \dots, x_n) =$ $a \cdot \parallel_{h \circlearrowleft (i,n), \vartheta_n(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)$	SI5
$\parallel_{h,s}^{n,i}(x_1, \dots, x_{i-1}, cr(d), x_{i+1}, \dots, x_n) =$ $\bar{cr}(d) \cdot \parallel_{h \circlearrowleft (i,n), \vartheta_n(h,s,i,cr(d))}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n, \phi(d))$	SI6
$\parallel_{h,s}^{n,i}(x_1, \dots, x_{i-1}, cr(d) \cdot x'_i, x_{i+1}, \dots, x_n) =$ $\bar{cr}(d) \cdot \parallel_{h \circlearrowleft (i,n+1), \vartheta_n(h,s,i,cr(d))}^{n+1}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n, \phi(d))$	SI7
$\parallel_{h,s}^{n,i}(x_1, \dots, x_{i-1}, x'_i + x''_i, x_{i+1}, \dots, x_n) =$ $\parallel_{h,s}^{n,i}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n) + \parallel_{h,s}^{n,i}(x_1, \dots, x_{i-1}, x''_i, x_{i+1}, \dots, x_n)$	SI8

Table 4 Alternative axioms for SI2

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

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 $\vartheta_n: \mathcal{H} \times S \times \{1, \dots, n\} \times A \rightarrow S$ must be extended to functions $\vartheta_n: \mathcal{H} \times S \times \{1, \dots, n\} \times (A \cup \{\delta\}) \rightarrow S$ and axiom SI2 must be replaced by the axioms in Table 4.

In $(ACP+SI)_{rec}$, i.e. $ACP+SI$ extended with guarded recursion in the way described in Section 2, the processes that can be created are restricted to the ones denotable by a closed $ACP+SI$ term. This restriction stems from the requirement that ϕ is a function from D to the set of all closed $ACP+SI$ terms. The restriction can be removed by relaxing this requirement to the requirement that ϕ is a function from D to the set of all closed $(ACP+SI)_{rec}$ terms. We write $(ACP+SI)_{rec}^+$ for the theory resulting from this relaxation. In other words, $(ACP+SI)_{rec}^+$ differs from $(ACP+SI)_{rec}$ 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 $(ACP+SI)_{rec}$, has been given.

It is customary to associate transition systems with closed terms of the language of an ACP -like 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 ACP can be found in [5, 13]. The additional transition rules for the strategic interleaving operators and the positional strategic interleaving operators are given in Appendix A.

3.2 Basic Properties of ACP with Strategic Interleaving

In this section, the subject of concern is the connection between ACP and $ACP+SI$. The main results are an elimination result and a conservative extension result. We begin with establishing some results that will be used in the proof of those main results.

Each guarded $ACP+SI$ term is derivably equal to a head normal form of $ACP+SI$.

Proposition 3 (Head normal form) *For each guarded $ACP+SI$ term t , there exists a head normal form t' of $ACP+SI$ such that $t = t'$ is derivable from the axioms of $ACP+SI$.*

Proof The proof is straightforward by induction on the structure of t . The case where t is of the form δ and the case where t is of the form a ($a \in A$) are trivial. The case where t is of the form $t_1 \cdot t_2$ follows immediately from the induction hypothesis and the claim that, for all head normal forms t_1 and t_2 of $ACP+SI$, there exists a head

normal form t' of ACP+SI such that $t_1 \cdot t_2 = t'$ is derivable from the axioms of ACP+SI. This claim is easily proved by induction on the structure of t_1 . The case where t is of the form $t_1 + t_2$ follows immediately from the induction hypothesis. The cases where t is of one of the forms $t_1 \parallel t_2$, $t_1 | t_2$, $\partial_H(t_1)$ or $\parallel_{h,s}^{n,i}(t_1, \dots, t_n)$ are proved along the same lines as the case where t is of the form $t_1 \cdot t_2$. In the case that t is of the form $t_1 | t_2$, each of the cases to be considered in the inductive proof of the claim demands a proof by induction on the structure of t_2 . In the case that t is of the form $\parallel_{h,s}^{n,i}(t_1, \dots, t_n)$, the claim is of course proved by induction on the structure of t_i instead of t_1 . The case that t is of the form $t_1 \parallel t_2$ follows immediately from the case that t is of the form $t_1 | t_2$ and the case that t is of the form $t_1 | t_2$. The case that t is of the form $\parallel_{h,s}^{n,i}(t_1, \dots, t_n)$ follows immediately from the case that t is of the form $\parallel_{h,s}^{n,i}(t_1, \dots, t_n)$. Because t is a guarded ACP+SI term, the case where t is a variable cannot occur. □

Each of the four theorems to come refer to several process algebras. It is implicit that the same set A of actions and the same communication function γ are assumed in the process algebras referred to.

Each guarded recursive specification over ACP+SI can be reduced to a guarded recursive specification over ACP.

Theorem 1 (Reduction) *For each guarded recursive specification E over ACP+SI and each $X \in V(E)$, there exists a guarded recursive specification E' over ACP such that $\langle X|E \rangle = \langle X|E' \rangle$ is derivable from the axioms of $(ACP+SI)_{rec}^+$.*

Proof Let E be a guarded recursive specification over ACP+SI. Assume that, for each equation $X = t_X$ from E , t_X is a guarded ACP+SI term. It follows from Proposition 1 that this assumption does not lead to loss of generality.

Let $X = t_X$ be an equation from E . Now, by Proposition 3, there exist $n, m \in \mathbb{N}$ such that, for each $i \in \mathbb{N}$ with $i < n$ and $j \in \mathbb{N}$ with $j < m$, there exist an $a_i \in A$, an ACP+SI term t_i , and a $b_j \in A$ such that $t_X = \sum_{i < n} a_i \cdot t_i + \sum_{j < m} b_j$ is derivable from the axioms of ACP+SI. For each $i \in \mathbb{N}$ with $i < n$, let t'_i be t_i with, for each equation $Y = t_Y$ from E , each unguarded occurrence of Y in t_i replaced by the guarded ACP+SI term t_Y . For each $i \in \mathbb{N}$ with $i < n$, by its construction, the term t'_i is a guarded ACP+SI terms in which variables other than the ones from $V(E)$ do not occur. Now, by Proposition 1, the equation $X_i = t'_i$, where X_i is a fresh variable, can be added to E for each $i \in \mathbb{N}$ with $i < n$ and the equation $X = t_X$ can be replaced by the equation $X = \sum_{i < n} a_i \cdot X_i + \sum_{j < m} b_j$ in E . The other equations from E can be replaced by a set of equations in the same way as the equation $X = t_X$.

The set of equations so obtained can be manipulated following the same procedure as in the case of E , but the manipulation can be restricted to the added equations. Repeating this procedure, perhaps countably infinitely many times, we obtain a guarded recursive specification E' over ACP for which $\langle X|E \rangle = \langle X|E' \rangle$ is derivable from the axioms of $(ACP+SI)_{rec}^+$. □

The next three theorems will be proved by means of term rewriting systems. In Appendix B, basic definitions and results regarding term rewriting systems are

collected. This appendix also serves to fix the terminology on term rewriting systems used in the proofs of the next three theorems.

Each closed $(ACP+SI)_{rec}^+$ term is derivably equal to a closed ACP_{rec} term.

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

Proof We prove this by means of a term rewriting system that takes equational axioms of $(ACP+SI)_{rec}^+$ and equations derivable from the axioms of $(ACP+SI)_{rec}^+$ as rewrite rules. Thus, the proof boils down to showing that (a) the term rewriting system concerned has the property that each $(ACP+SI)_{rec}^+$ term has a unique normal form modulo axioms A1 and A2 and (b) each closed $(ACP+SI)_{rec}^+$ term that is a normal form modulo axioms A1 and A2 is a closed ACP_{rec} term. Henceforth, we will write AC for the set of equations that consists of axioms A1 and A2.

Let R be a set of equations that contains for each guarded recursive specification E over $ACP+SI$ and $X \in V(E)$ an equation $\langle X|E \rangle = \langle X|E' \rangle$, where E' is a guarded recursive specification over ACP , that is derivable from the axioms of $(ACP+SI)_{rec}^+$. Such a set R exists by Theorem 1. Consider the term rewriting system $\mathcal{R}((ACP+SI)_{rec}^+)$ that consists of the axioms of $(ACP+SI)_{rec}^+$, with the exception of A1, A2, RDP, and RSP, and the equations from R taken as rewrite rules.

We show that $\mathcal{R}((ACP+SI)_{rec}^+)$ has the property that each $(ACP+SI)_{rec}^+$ term has a unique normal form modulo AC by proving that $\mathcal{R}((ACP+SI)_{rec}^+)$ is terminating modulo AC and confluent modulo AC.

First, we show that $\mathcal{R}((ACP+SI)_{rec}^+)$ is terminating modulo AC. This can be proved by the reduction ordering $>$ induced by the extended integer polynomials $\theta(t)$ associated with $(ACP+SI)_{rec}^+$ terms t as follows:⁶

$$\begin{aligned} \theta(X) &= \underline{X}, & \theta(t_1 \parallel t_2) &= 3 \cdot (\theta(t_1) \cdot \theta(t_2))^2 + 1, \\ \theta(a) &= 2, & \theta(t_1 \ll t_2) &= (\theta(t_1) \cdot \theta(t_2))^2, \\ \theta(\delta) &= 2, & \theta(t_1 | t_2) &= (\theta(t_1) \cdot \theta(t_2))^2, \\ \theta(\text{cr}(d)) &= \theta(\phi(d))^2 + 1, & \theta(\partial_H(t)) &= 2^{\theta(t)}, \\ \theta(t_1 + t_2) &= \theta(t_1) + \theta(t_2), & \theta(\|_{h,s}^n(t_1, \dots, t_n)) &= (\theta(t_1) \cdot \dots \cdot \theta(t_n))^2 + 1, \\ \theta(t_1 \cdot t_2) &= \theta(t_1)^2 \cdot \theta(t_2), & \theta(\ll_{h,s}^{n,i}(t_1, \dots, t_n)) &= (\theta(t_1) \cdot \dots \cdot \theta(t_n))^2, \end{aligned}$$

$$\theta(\langle X|E \rangle) = \begin{cases} 2 & \text{if } E \text{ is a guarded recursive specification over } ACP \\ 3 & \text{otherwise,} \end{cases}$$

where it is assumed that, for each variable X over processes, \underline{X} is a variable over integers. The following is easy to see: (a) $t > t'$ for all rewrite rules $t = t'$ of $\mathcal{R}((ACP+SI)_{rec}^+)$ and (b) $t > t'$ implies $s > s'$ for all $(ACP+SI)_{rec}^+$ terms s and s' for which $t = s$ and $t' = s'$ are derivable from AC.⁷ Hence, $\mathcal{R}((ACP+SI)_{rec}^+)$ is terminating modulo AC.

⁶Here, extended polynomials differ from polynomials in that both variables and expressions of the form 2^X , where X is a variable, are allowed where only variables are allowed in polynomials.

⁷We do not have that $t > t'$ for all rewrite rules $t = s$ if SI2 is replaced by SI2a and SI2b (see Table 4).

Next, we show that $\mathcal{R}((ACP+SI)_{rec}^+)$ is confluent modulo AC. It follows from Theorems 5 and 16 in [20] and the fact that $\mathcal{R}((ACP+SI)_{rec}^+)$ is terminating modulo AC that $\mathcal{R}((ACP+SI)_{rec}^+)$ is confluent modulo AC if it does not give rise to critical pairs modulo AC that are not convergent. It is easy to see that all critical pairs modulo AC arise from overlappings of (a) A3 on A4, CM4, CM8, CM9, D3, and SI8, (b) A6 on A4, CM4, CM8, CM9, D3, and SI8, (c) A7 on CM3, CM5, CM6, CM7, D4, and SI5, (d) CM10 on CM9, and (e) CM11 on CM8. It is straightforward to check that all critical pairs concerned are convergent. Hence, $\mathcal{R}((ACP+SI)_{rec}^+)$ is confluent modulo AC.

Above, we have shown that $\mathcal{R}((ACP+SI)_{rec}^+)$ is terminating modulo AC and confluent modulo AC and by this that it has the property that each $(ACP+SI)_{rec}^+$ term has a unique normal form modulo AC. It remains to be shown that each closed $(ACP+SI)_{rec}^+$ term that is a normal form modulo AC is a closed ACP_{rec} term. It is not hard to see that, for each closed $(ACP+SI)_{rec}^+$ term in which other operators than $+$ and \cdot occur, a reduction step modulo AC is still possible in $\mathcal{R}((ACP+SI)_{rec}^+)$. Because a reduction step modulo AC is impossible for a normal form modulo AC, no other operators than $+$ or \cdot can occur in a closed $(ACP+SI)_{rec}^+$ term that is a normal form modulo AC. Hence, each closed $(ACP+SI)_{rec}^+$ term that is a normal form modulo AC is a closed ACP_{rec} term. \square

Each equation between closed ACP terms that is derivable in $ACP+SI$ is also derivable in ACP .

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

Proof We prove this by means of a restriction of the term rewriting system from the proof of Theorem 2. Consider the term rewriting system $\mathcal{R}(ACP+SI)$ that consists of the axioms of $ACP+SI$, with the exception of A1 and A2. $\mathcal{R}(ACP+SI)$ is $\mathcal{R}((ACP+SI)_{rec}^+)$ restricted to $ACP+SI$ terms. Just like $\mathcal{R}((ACP+SI)_{rec}^+)$, $\mathcal{R}(ACP+SI)$ is terminating modulo AC and confluent modulo AC. The proofs of these properties for $\mathcal{R}((ACP+SI)_{rec}^+)$ carry over to $\mathcal{R}(ACP+SI)$.

Let t and t' be two closed ACP terms such that $t = t'$ is derivable from the axioms of $ACP+SI$. Reduce t and t' to normal forms s and s' , respectively, by means of the term rewriting system $\mathcal{R}(ACP+SI)$. By Theorem 5 in [20], being confluent modulo AC is equivalent to being Church-Rosser modulo AC for a term rewriting system that is terminating modulo AC. This means that t and t' have the same normal form modulo AC. In other words, $s = s'$ is derivable from axioms A1 and A2. Because (a) no other operators than $+$ and \cdot occur in t and t' and (b) no rewrite rule introduces one or more of the other operators if one or more of the other operators was not already in its left-hand side, each rewrite rule applied in the reduction from t to s or the reduction from t' to s' is one of the axioms of ACP . Therefore, each rewrite rule involved in the reduction from t to s or the reduction from t' to s' is an axiom of ACP . Hence, the reduction from t to s shows that $t = s$ is derivable from the axioms of

ACP and the reduction from t' to s' shows that $t' = s'$ is derivable from the axioms of ACP. From this and the fact that $s = s'$ is derivable from axioms A1 and A2, it follows $t = t'$ is derivable from the axioms of ACP. \square

The following theorem concerns the expansion of minimal models of ACP to models of ACP+SI.

Theorem 4 (Unique expansion) *Each minimal model of ACP has a unique expansion to a model of ACP+SI.*

Proof We write $f^{\mathcal{A}}$, where \mathcal{A} is a model of ACP or ACP+SI and f is a constant or operator from the signature of \mathcal{A} , for the interpretation of f in \mathcal{A} . We write $t^{\mathcal{A}}$, where \mathcal{A} is a model of ACP or ACP+SI and t is a closed term over the signature of \mathcal{A} , for the interpretation of t in \mathcal{A} .

Let \mathcal{A} be a minimal model of ACP. Let CT be a function from the carrier of \mathcal{A} to the set of all closed ACP terms such that, for each element p of the carrier of \mathcal{A} , $CT(p)^{\mathcal{A}} = p$. Because \mathcal{A} is a minimal model of ACP, $CT(p)$ is a total function. We write \underline{p} , where p is an element of the carrier of \mathcal{A} , for $CT(p)$. Let NF be a function from the set of all closed ACP+SI terms to the set of all closed ACP terms such that, for each closed ACP+SI term t , $NF(t)$ is one of the normal forms that t can be reduced to by means of the term rewriting system $\mathcal{R}(\text{ACP+SI})$ from the proof of Theorem 3.

We start with constructing an expansion of \mathcal{A} with interpretations of the additional operators of ACP+SI. Let \mathcal{B} be the expansion of \mathcal{A} with interpretations of the additional operators of ACP+SI where these interpretations are defined as follows:

$$\begin{aligned} \parallel_{h,s}^n{}^{\mathcal{B}}(p_1, \dots, p_n) &= NF(\parallel_{h,s}^n(\underline{p}_1, \dots, \underline{p}_n))^{\mathcal{A}}, \\ \perp_{h,s}^{n,i}{}^{\mathcal{B}}(p_1, \dots, p_n) &= NF(\perp_{h,s}^{n,i}(\underline{p}_1, \dots, \underline{p}_n))^{\mathcal{A}}, \end{aligned}$$

for all p_1, \dots, p_n from the carrier of \mathcal{A} .

We proceed with proving that \mathcal{B} is a model of ACP+SI. By Theorem 3, it is sufficient to prove that \mathcal{B} satisfies axioms SI1–SI8. By its construction, \mathcal{B} is a minimal algebra and consequently it is sufficient to prove that \mathcal{B} satisfies all closed substitution instances of SI1–SI8. We use the following three claims to prove this:

- for all closed substitution instances $t = t'$ of SI1–SI8, $t^{\mathcal{B}} = NF(t)^{\mathcal{A}}$;
- for all closed substitution instances $t = t'$ of SI1–SI8, $t'^{\mathcal{B}} = NF(t')^{\mathcal{A}}$;
- for all closed substitution instances $t = t'$ of SI1–SI8, $NF(t)^{\mathcal{A}} = NF(t')^{\mathcal{A}}$.

The first claim follows easily from the definitions of the interpretations of the additional operators of ACP+SI given above. The second claim follows easily from these definitions and the proof of the first claim. Because $\mathcal{R}(\text{ACP+SI})$ is Church-Rosser modulo AC (see the proof of Theorem 3), we have that $NF(t) = NF(t')$ is derivable from axioms A1 and A2. From this, the third claim follows immediately. It is an immediate consequence of the three claims that \mathcal{B} satisfies all closed substitution instances of SI1–SI8.

We still have to prove that \mathcal{B} is the only expansion of \mathcal{A} to a model of ACP+SI. We can prove this by contradiction. Assume that \mathcal{C} is an expansion of \mathcal{A} to a model of

ACP+SI that differs from \mathcal{B} . Then at least one of the additional operators of ACP+SI has different interpretations in \mathcal{B} and \mathcal{C} . By the definitions of the interpretations of the additional operators of ACP+SI in \mathcal{B} , this means that there exists a closed ACP+SI term t such that $t^{\mathcal{C}} \neq NF(t)^{\mathcal{A}}$. Moreover, because $t = NF(t)$ is derivable from the axioms of ACP+SI, $t^{\mathcal{C}} = NF(t)^{\mathcal{C}}$. Hence, $NF(t)^{\mathcal{C}} \neq NF(t)^{\mathcal{A}}$. Because $NF(t)$ is a closed ACP term, this contradicts the fact that \mathcal{C} is an expansion of \mathcal{A} . \square

4 Concluding Remarks

We have extended the algebraic theory of processes known as ACP with the form of interleaving that underlies multi-threading as found in contemporary programming languages. We have also established some basic properties of the resulting theory. It remains an open question whether strategic interleaving is definable in an established extension of ACP.

Acknowledgements We thank an anonymous referee for carefully reading a preliminary version of this paper, for pointing out an error in one of the proofs, and for suggesting improvements of the presentation.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix A: Structural Operational Semantics of ACP+SI

It is customary to associate transition systems with closed terms of the language of an ACP-like 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 ACP can be found in [5, 13]. The additional transition rules for the strategic interleaving operators and the positional strategic interleaving operators are given in Table 5. In this table,

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

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

Appendix B: Term Rewriting Systems

In this appendix, basic definitions and results regarding term rewriting systems are collected. This appendix also serves to fix the terminology on term rewriting systems used in the proofs that make use of term rewriting systems.

Table 5 Transition rules for strategic interleaving

$\frac{x \xrightarrow{\alpha} \surd}{\ _{h,s}^1(x) \xrightarrow{\alpha} \surd}$
$\frac{x_i \xrightarrow{\alpha} \surd \quad i = \sigma_n(h, s)}{\ _{h,s}^{n+1}(x_1, \dots, x_{n+1}) \xrightarrow{\alpha} \ _{h \curvearrowright (i,n), \vartheta_{n+1}(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1})}$
$\frac{x_i \xrightarrow{\alpha} x'_i \quad i = \sigma_n(h, s)}{\ _{h,s}^n(x_1, \dots, x_n) \xrightarrow{\alpha} \ _{h \curvearrowright (i,n), \vartheta_n(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)}$
$\frac{x_i \xrightarrow{\text{cr}(d)} \surd \quad i = \sigma_n(h, s)}{\ _{h,s}^n(x_1, \dots, x_n) \xrightarrow{\text{cr}(d)} \ _{h \curvearrowright (i,n), \vartheta_n(h,s,i,\text{cr}(d))}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n, \phi(d))}$
$\frac{x_i \xrightarrow{\text{cr}(d)} x'_i \quad i = \sigma_n(h, s)}{\ _{h,s}^n(x_1, \dots, x_n) \xrightarrow{\text{cr}(d)} \ _{h \curvearrowright (i,n+1), \vartheta_n(h,s,i,\text{cr}(d))}^{n+1}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n, \phi(d))}$
$\frac{x \xrightarrow{\alpha} \surd}{\ _{h,s}^{1,i}(x) \xrightarrow{\alpha} \surd}$
$\frac{x_i \xrightarrow{\alpha} \surd}{\ _{h,s}^{n+1,i}(x_1, \dots, x_{n+1}) \xrightarrow{\alpha} \ _{h \curvearrowright (i,n), \vartheta_{n+1}(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{n+1})}$
$\frac{x_i \xrightarrow{\alpha} x'_i}{\ _{h,s}^{n,i}(x_1, \dots, x_n) \xrightarrow{\alpha} \ _{h \curvearrowright (i,n), \vartheta_n(h,s,i,a)}^n(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)}$
$\frac{x_i \xrightarrow{\text{cr}(d)} \surd}{\ _{h,s}^{n,i}(x_1, \dots, x_n) \xrightarrow{\text{cr}(d)} \ _{h \curvearrowright (i,n), \vartheta_n(h,s,i,\text{cr}(d))}^n(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n, \phi(d))}$
$\frac{x_i \xrightarrow{\text{cr}(d)} x'_i}{\ _{h,s}^{n,i}(x_1, \dots, x_n) \xrightarrow{\text{cr}(d)} \ _{h \curvearrowright (i,n+1), \vartheta_n(h,s,i,\text{cr}(d))}^{n+1}(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n, \phi(d))}$

We assume that a set of constants, a set of operators with fixed arities, and a set of variables have been given; and we consider an arbitrary term rewriting system \mathcal{R} for terms that can be built from the constants, operators, and variables in these sets.

A *rewrite rule* is a pair of terms $t \rightarrow s$, where t is not a variable and each variable occurring in s occurs in t as well. A *term rewriting system* is a set of rewrite rules.

A *reduction step* of \mathcal{R} is a pair $t \rightarrow s$ such that for some substitution instance $t' \rightarrow s'$ of a rewrite rule of \mathcal{R} , t' is a subterm of t , and s is t with t' replaced by s' . Here, t' is called the *redex* of the reduction step. A *reduction* of \mathcal{R} is a pair $t \rightarrow s$ such that either $t \equiv s$ or there exists a finite sequence $t_1 \rightarrow t_2, \dots, t_n \rightarrow t_{n+1}$ of consecutive reduction steps of \mathcal{R} such that $t \equiv t_1$ and $s \equiv t_{n+1}$.

A term t is a *normal form* of \mathcal{R} if there does not exist a term s such that $t \rightarrow s$ is a reduction step of \mathcal{R} . A term t has a *normal form* in \mathcal{R} if there exists a reduction $t \rightarrow s$ of \mathcal{R} and s is a normal form of \mathcal{R} . \mathcal{R} is *terminating* on term t if there does not exist an infinite sequence $t \rightarrow t_1, t_1 \rightarrow t_2, t_2 \rightarrow t_3, \dots$ of consecutive reduction steps of \mathcal{R} . \mathcal{R} is *terminating* if \mathcal{R} is terminating on all terms. \mathcal{R} is *confluent* if for all reductions $t \rightarrow s_1$ and $t \rightarrow s_2$ of \mathcal{R} there exist reductions $s_1 \rightarrow s$ and $s_2 \rightarrow s$ of \mathcal{R} . If \mathcal{R} is terminating and confluent, then each term has a unique normal form in \mathcal{R} .

A *reduction ordering* for \mathcal{R} is a well-founded ordering on terms that is closed under substitutions and contexts. \mathcal{R} is terminating if and only if there exists a reduction ordering $>$ for \mathcal{R} such that $t > s$ for each rewrite rule $t \rightarrow s$ of \mathcal{R} .

A *unifier* of two terms s and t is a substitution σ such that $\sigma(s) \equiv \sigma(t)$. A *critical pair* of \mathcal{R} is a pair (t_1, t_2) of terms for which there exist rewrite rules $s \rightarrow s'$ and $t \rightarrow t'$ of \mathcal{R} and a ‘most general unifier’ σ of s and a non-variable subterm of t such that $t_1 \equiv \sigma(t'')$ and $t_2 \equiv \sigma(t')$, where t'' is t with $\sigma(s)$ replaced by $\sigma(s')$.⁸ A critical pair (t_1, t_2) of \mathcal{R} is *convergent* if there exist reductions $t_1 \rightarrow s$ and $t_2 \rightarrow s$ of \mathcal{R} . If \mathcal{R} is terminating, then \mathcal{R} is confluent if and only if all critical pairs of \mathcal{R} are convergent.

Henceforth, we consider an arbitrary set E of equations between terms.

A *reduction step modulo E* of \mathcal{R} is a pair $t \rightarrow_E s$ such that there exists a reduction step $t' \rightarrow s'$ of \mathcal{R} such that $t = t'$ and $s = s'$ are derivable from E . A *reduction modulo E* of \mathcal{R} is pair $t \rightarrow_E s$ such that either $t = s$ is derivable from E or there exists a finite sequence $t_1 \rightarrow_E t_2, \dots, t_n \rightarrow_E t_{n+1}$ of consecutive reduction steps modulo E of \mathcal{R} such that $t \equiv t_1$ and $s \equiv t_{n+1}$.

A term t is a *normal form modulo E* of \mathcal{R} if there does not exist a term s such that $t \rightarrow_E s$ is a reduction step modulo E of \mathcal{R} . A term t has a *normal form modulo E* in \mathcal{R} if there exists a reduction modulo E $t \rightarrow_E s$ of \mathcal{R} and s is a normal form modulo E of \mathcal{R} . \mathcal{R} is *terminating modulo E* on term t if there does not exist an infinite sequence $t \rightarrow_E t_1, t_1 \rightarrow_E t_2, t_2 \rightarrow_E t_3, \dots$ of consecutive reduction steps modulo E of \mathcal{R} . \mathcal{R} is *terminating modulo E* if \mathcal{R} is terminating modulo E on all terms. \mathcal{R} is *confluent modulo E* if for all reductions modulo E $t \rightarrow_E s_1$ and $t \rightarrow_E s_2$ of \mathcal{R} there exist reductions modulo E $s_1 \rightarrow_E s$ and $s_2 \rightarrow_E s$ of \mathcal{R} . If \mathcal{R} is terminating modulo E and confluent modulo E , then each term has a unique normal form modulo E in \mathcal{R} .

A reduction ordering $>$ for \mathcal{R} is *E -compatible* if $t > s$ implies $t' > s'$ for all terms t' and s' for which $t = t'$ and $s = s'$ are derivable from E . \mathcal{R} is *terminating modulo E* if and only if there exists an E -compatible reduction ordering $>$ for \mathcal{R} such that $t > s$ for each rewrite rule $t \rightarrow s$ of \mathcal{R} .

A *unifier modulo E* of two terms s and t is a substitution σ such that $\sigma(s) = \sigma(t)$ is derivable from E . A *critical pair modulo E* of \mathcal{R} is a pair (t_1, t_2) of terms for which there exist rewrite rules $s \rightarrow s'$ and $t \rightarrow t'$ of \mathcal{R} and a substitution σ from a ‘complete set of unifiers modulo E' ’ of s and a non-variable subterm of t such that $t_1 \equiv \sigma(t'')$ and $t_2 \equiv \sigma(t')$, where t'' is t with $\sigma(s)$ replaced by $\sigma(s')$.⁸ If \mathcal{R} is terminating modulo E , then \mathcal{R} is confluent modulo E if and only if all critical pairs modulo E of \mathcal{R} are convergent.

An *E -equality step* is a pair $t \vdash_E s$ such that, for some substitution instance $t' = s'$ of an equation from E , either t' is a subterm of t and s is t with t' replaced by s' or s' is a subterm of t and s is t with s' replaced by t' . An $\mathcal{R} \cup E$ -equality step is a pair $t \vdash_E s$ such that $t \rightarrow s$ is a reduction step of \mathcal{R} or $s \rightarrow t$ is a reduction step of \mathcal{R} or $t \vdash_E s$ is an E -equality step. An $\mathcal{R} \cup E$ -equality is a pair $t \vdash_E^* s$ such that either $t \equiv s$ or there exists a finite sequence $t_1 \vdash_E t_2, \dots, t_n \vdash_E t_{n+1}$ of consecutive $\mathcal{R} \cup E$ -equality steps such that $t \equiv t_1$ and $s \equiv t_{n+1}$. \mathcal{R} is *Church-Rosser modulo E* if for all

⁸See e.g. Definition 10 in [20] for the definitions of most general unifier and complete set of unifiers modulo E .

$\mathcal{R} \cup E$ -equalities $t \stackrel{\text{tr}}{\equiv}_E t'$ there exist reductions modulo E $t \rightarrow_E s$ and $t' \rightarrow_E s$ of \mathcal{R} . If \mathcal{R} is terminating modulo E , then \mathcal{R} is Church-Rosser modulo E if and only if \mathcal{R} is confluent modulo E .

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

- America, P., de Bakker, J.W.: Designing equivalent semantic models for process creation. *Theor. Comput. Sci.* **60**(2), 109–176 (1988)
- Baeten, J.C.M., Bergstra, J.A.: Real space process algebra. *Form. Asp. Comput.* **5**(6), 481–529 (1993)
- Baeten, J.C.M., Middelburg, C.A.: *Process Algebra with Timing*. Monographs in Theoretical Computer Science, an EATCS Series. Springer, Berlin (2002)
- Baeten, J.C.M., Vaandrager, F.W.: An algebra of process creation. *Acta Informatica* **29**(4), 303–334 (1992)
- Baeten, J.C.M., Weijland, W.P.: *Process Algebra* Cambridge Tracts in Theoretical Computer Science, vol. 18. Cambridge University Press, Cambridge (1990)
- Bergstra, J.A.: A process creation mechanism in process algebra. In: Baeten, J.C.M. (ed.) *Applications of Process Algebra*, Cambridge Tracts in Theoretical Computer Science, vol. 17, pp. 81–88. Cambridge University Press, Cambridge (1990)
- Bergstra, J.A., Klop, J.W.: Process algebra for synchronous communication. *Inf. Control.* **60**(1–3), 109–137 (1984)
- Bergstra, J.A., Middelburg, C.A.: Thread algebra for strategic interleaving. *Form. Asp. Comput.* **19**(4), 445–474 (2007)
- Bergstra, J.A., Middelburg, C.A.: A thread algebra with multi-level strategic interleaving. *Theory Comput. Syst.* **41**(1), 3–32 (2007)
- Bergstra, J.A., Middelburg, C.A.: Distributed strategic interleaving with load balancing. *Futur. Gener. Comput. Syst.* **24**(6), 530–548 (2008)
- Bergstra, J.A., Middelburg, C.A., Usenko, Y.S.: Discrete time process algebra and the semantics of SDL. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) *Handbook of Process Algebra*, pp. 1209–1268. Elsevier, Amsterdam (2001)
- Brookes, S.D., Hoare, C.A.R., Roscoe, A.W.: A theory of communicating sequential processes. *J. ACM* **31**(3), 560–599 (1984)
- Fokkink, W.J.: *Introduction to Process Algebra*. Texts in Theoretical Computer Science, an EATCS Series. Springer, Berlin (2000)
- Gehrke, T., Rensink, A.: Process creation and full sequential composition in a name-passing calculus. *Electron. Notes Theor. Comput. Sci.* **7**, 141–160 (1997)
- van Glabbeek, R.J., Vaandrager, F.W.: Modular specification of process algebras. *Theor. Comput. Sci.* **113**(2), 293–348 (1993)
- Gosling, J., Joy, B., Steele, G., Bracha, G. *The Java Language Specification*, 2nd edn. Addison-Wesley, Reading (2000)
- Hejlsberg, A., Wiltamuth, S., Golde, P.: *C# Language Specification*. Addison-Wesley, Reading (2003)
- Hennessy, M., Milner, R.: Algebraic laws for non-determinism and concurrency. *J. ACM* **32**(1), 137–161 (1985)
- Hoare, C.A.R.: *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs (1985)
- Jouannaud, J.P., Kirchner, H.: Completion of a set of rules modulo a set of equations. *SIAM J. Comput.* **15**(4), 1155–1194 (1986)
- Milner, R.: *Communication and Concurrency*. Prentice-Hall, Englewood Cliffs (1989)
- Sabelfeld, A., Sands, D.: Probabilistic noninterference for multi-threaded programs. In: *Computer Security Foundations Workshop 2000*, pp. 200–214. IEEE Computer Society Press (2000)