Fundamenta Informaticae 00 (0000) 1–29
IOS Press

Located Actions in Process Algebra with Timing

J.A. Bergstra
Programming Research Group
University of Amsterdam
Amsterdam, the Netherlands
janb@science.uva.nl

Department of Philosophy
Utrecht University
Utrecht, the Netherlands
janb@phil.uu.nl

C.A. Middelburg
Computing Science Department
Eindhoven University of Technology
Eindhoven, the Netherlands
keesm@win.tue.nl

Abstract. We propose a process algebra obtained by adapting the process algebra with continuous relative timing from Baeten and Middelburg [Process Algebra with Timing, Springer, 2002, Chap. 4] to spatially located actions. This process algebra makes it possible to deal with the behaviour of systems with a known time-dependent spatial distribution, such as protocols transmitting data via a mobile intermediate station. It is a reformulation of the real space process algebra from Baeten and Bergstra [Formal Aspects of Computing, 5, 1993, 481–529] in a setting with urgent actions. This leads to many simplifications.

Keywords: process algebra, continuous relative timing, spatially located actions, distributed systems, state operator, maximal progress, asynchronous communication, urgent actions

Address for correspondence: Computing Science Department, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands

1. Introduction

In this paper, we study a process algebraic approach to the theory of distributed systems with a known, possibly time-dependent, spatial distribution.

In ACP-style process algebras, in common with most other process algebras, there is only one action left when two or more actions are performed synchronously. The reason for this is that two or more actions are considered to be performed synchronously if they are performed jointly at the same point of time. This is, for example, not the case for actions that are performed at the same point of time at different locations in a distributed system. Such actions are performed independently at the same point of time. Just like the situation that actions are to be performed jointly at the same point of time, the situation that actions are to be performed independently at the same point of time arises from processes that proceed in parallel. In process algebras with timing in which the possibility of two or more actions to be performed at the same point of time is excluded, such as the process algebras with continuous timing from [2], it requires an artifice such as multi-actions (see also [2]) to deal with the latter situation.

Many process algebras with timing, including the ACP-style process algebras with continuous timing from [6], feature urgent actions. This means that it is possible for two or more actions to be performed consecutively at the same point of time. Because of this feature, which is justified in [16], those process algebras from [6] can deal in a simple way with the situation that two or more actions are to be performed independently at the same point of time: interleaving sees to it that the actions concerned can be performed consecutively at the same point of time in any order.

As mentioned above, actions that are performed at different locations in a distributed system cannot be performed synchronously, in the strict sense that they cannot be performed jointly at the same point of time. In [3], an adaptation of ACPρ, the process algebra with continuous absolute timing from [2], is introduced which enforces this. For the enforcement, processes that are capable of performing an action at a fixed location in space, and then terminating successfully, at point of time 0 are taken as atomic processes. This adaptation, called ACP$\sigma\rho$, is further extended in [3] with a state operator and a maximal progress operator to model communication between processes at different places in space, which involves non-zero transmission times.

In this paper, we present a reformulation of the material presented in [3] in a setting with urgent actions. We propose a process algebra obtained by adapting $\mathrm{ACP}^{\mathrm{srt}}$, the process algebra with continuous relative timing from [6], to spatially located actions like in [3]. In accordance with [3], we extend this adaptation, called $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$, with a state operator and a maximal progress operator. As a result of the simpler way in which is dealt with the situation that two or more actions are to be performed independently at the same point of time, the state operator and the maximal progress operator are far less complicated than in [3].

In [6], a coherent collection of four process algebras with timing, each dealing with timing in a different way, is presented. The time scale on which the time is measured is either discrete or continuous, and the timing of actions is either relative or absolute. There is no other reason to choose for relative timing in this paper but the fact that it is generally considered to be simpler than absolute timing. This does not mean that relative timing has only merits, see e.g. [18].

Various constants and operators of the process algebra with continuous relative timing have counterparts in the other versions from the above-mentioned collection. A notational distinction is made between a constant or operator of one version and its counterpart in another version, by means of different decorations of a common symbol, if they should not be identified in case versions are integrated.

So long as one uses a single version, one can safely omit those decorations. However, we refrain from omitting them in this paper because we think that change of notation in a series of scientific publications is undesirable.

We distinguish between the process algebra that is the mere adaptation of $\mathrm{ACP}^{\mathrm{srt}}$ to spatially located actions and extensions that are useful in many applications. Integration, which provides for alternative composition over a continuum of differently timed alternatives, and guarded recursion, which allows for the description of (potentially) non-terminating processes, are needed in many applications of the proposed process algebra. Both integration and guarded recursion are treated as extensions. What is also needed in many applications is action renaming, providing for change of actions, and spatial replacement, providing for time-dependent change of locations. Spatial replacement makes it possible to describe the behaviour of processes of which the location in space changes while time passes, i.e. processes that move in space. Action renaming and spatial replacement are also treated as extensions.

The structure of this paper is as follows. First of all, we introduce the adaptation of the process algebra with continuous relative timing from [6] to spatially located actions (Section 2). Next, we consider the addition of integration and recursion (Section 3), the addition of action renaming and spatial replacement (Section 4), and the addition of the state operator and the maximal progress operator (Section 5). After that, we illustrate the use of the whole by means of an example concerning data transmission via a mobile intermediate station (Sections 6 and 7). Finally, some concluding remarks are made (Section 8).

In the remainder of this paper, we will mostly refer to process algebras by name. As mentioned above, the process algebra with continuous relative timing from [6] is called $\mathrm{ACP}^{\mathrm{srt}}$ and the new process algebra proposed in this paper is called $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$. Both process algebras are extensions of ACP [8, 9]. We will also refer to BPA and BPA_δ, which are names of subtheories of ACP that do not cover parallelism and communication. The difference between them is that BPA does not cover deadlock and BPA_δ does. $\mathrm{ACP}^{\mathrm{srt}}$ was first introduced in [5]. In this paper, we mostly refer to [6] because it contains a more extensive treatment of $\mathrm{ACP}^{\mathrm{srt}}$.

2. $\mathrm{ACP}^{\mathrm{srt}}$ with Spatially Located Actions

$\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$ is an adaptation of $\mathrm{ACP}^{\mathrm{srt}}$ [6] to spatially located actions. In $\mathrm{ACP}^{\mathrm{srt}}$, timing is relative to the time at which the preceding action is performed and time is measured on a continuous time scale. Roughly speaking, $\mathrm{ACP}^{\mathrm{srt}}$ is ACP [8, 9] extended with two operators to deal with timing: relative delay and relative undelayable time-out. The first operator is a basic one and the second operator is an auxiliary one.

In $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$, the atomic processes are undelayable located actions and undelayable deadlock. Let a be an action and $\xi \in \mathbb{R}^3$. Then *undelayable action a located at ξ*, written $\tilde{\tilde{a}}(\xi)$, is the process that immediately performs action a at location ξ, at the current point of time, and then terminates successfully. Undelayable located actions are idealized in the sense that they are treated as if they are performed instantaneously at a point in space. In order to deal with unsuccessful termination, we need an additional atomic process that is neither capable of performing any action nor capable of idling beyond the current point of time. This process, written $\tilde{\tilde{\delta}}$, is called *undelayable deadlock*.

$\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$ has the following operators:

- the *relative delay* of P for a period of time r, written $\sigma^r_{\mathsf{rel}}(P)$, is the process that idles for a period of time r and then behaves like P;

- the *alternative composition* of P_1 and P_2, written $P_1 + P_2$, is the process that behaves either like P_1 or like P_2, but not both;
- the *sequential composition* of P_1 and P_2, written $P_1 \cdot P_2$, is the process that first behaves like P_1, but when P_1 terminates successfully it continues by behaving like P_2;
- the *parallel composition* of P_1 and P_2, written $P_1 \parallel P_2$, is the process that proceeds with P_1 and P_2 in parallel;
- the *left merge* of P_1 and P_2, written $P_1 \lfloor\!\lfloor P_2$, is the same as $P_1 \parallel P_2$ except that $P_1 \lfloor\!\lfloor P_2$ starts with performing an action of P_1;
- the *communication merge* of P_1 and P_2, written $P_1 \mid P_2$, is the same as $P_1 \parallel P_2$ except that $P_1 \mid P_2$ starts with performing an action of P_1 and an action of P_2 synchronously;
- the *encapsulation* of P with respect to H, written $\partial_H(P)$, keeps P from performing actions in H;
- the *relative undelayable time-out* of P, written $\nu_{\mathsf{rel}}(P)$, keeps P entirely from idling.

These operators are essentially the same as the corresponding ones of $\mathrm{ACP}^{\mathrm{srt}}$, with the exception of the following: in the case where a process proceeds with two processes in parallel, the processes concerned are not able to perform actions synchronously if they cannot perform them at the same location. For a more comprehensive informal explanation of the operators, the reader is referred to [6]. Here, we only point at the most important issues:

- In $P_1 + P_2$, there is an arbitrary choice between P_1 and P_2. The choice is resolved on one of them performing its first action, and not otherwise. Consequently, the choice between two idling processes will always be postponed until at least one of the processes can perform its first action. Only when both processes cannot idle any longer, further postponement is not an option. If the choice has not yet been resolved when one of the processes cannot idle any longer, the choice will simply not be resolved in its favour. For example, the process $\sigma^3_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi)) + \sigma^1_{\mathsf{rel}}(\tilde{\tilde{b}}(\xi'))$ behaves the same as the process $\sigma^1_{\mathsf{rel}}(\tilde{\tilde{b}}(\xi') + \sigma^2_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi)))$. Both processes idle for a period of 1 time unit, and after that either (i) first perform action b at location ξ' and then terminate successfully or (ii) idle further for a period of 2 time units, and after that first perform action a at location ξ and then terminate successfully.
- $P_1 \parallel P_2$ can behave in the following ways: (i) first either P_1 or P_2 performs its first action and next it proceeds in parallel with the process following that action and the process that did not perform an action; (ii) if their first actions can be performed synchronously, first P_1 and P_2 perform their first actions synchronously and next it proceeds in parallel with the processes following those actions. However, P_1 and P_2 may have to idle before they can perform their first action. Therefore, $P_1 \parallel P_2$ can only start with: (i) performing an action of P_1 or P_2 if it can do so before or at the ultimate point of time for the other process to start performing actions or to deadlock; (ii) performing an action of P_1 and an action of P_2 synchronously if both processes can do so at the same point of time. For example, the process $(\sigma^2_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi)) + \sigma^3_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi))) \parallel \sigma^3_{\mathsf{rel}}(\tilde{\tilde{b}}(\xi))$ behaves the same as the process $\sigma^2_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi) \cdot \sigma^1_{\mathsf{rel}}(\tilde{\tilde{b}}(\xi))) + \sigma^3_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi) \cdot \tilde{\tilde{b}}(\xi)) + \sigma^3_{\mathsf{rel}}(\tilde{\tilde{b}}(\xi) \cdot \tilde{\tilde{a}}(\xi)) + \sigma^3_{\mathsf{rel}}(\tilde{\tilde{c}}(\xi))$ if performing a and b synchronously yields c.

Table 1. Axioms of $\mathrm{BPA}_{\mathrm{la}}^{\mathrm{srt}}$ ($p, q \geq 0$)

$x + y = y + x$	A1	$\sigma_{\mathsf{rel}}^0(x) = x$	SRT1
$(x + y) + z = x + (y + z)$	A2	$\sigma_{\mathsf{rel}}^p(\sigma_{\mathsf{rel}}^q(x)) = \sigma_{\mathsf{rel}}^{p+q}(x)$	SRT2
$x + x = x$	A3	$\sigma_{\mathsf{rel}}^p(x) + \sigma_{\mathsf{rel}}^p(y) = \sigma_{\mathsf{rel}}^p(x + y)$	SRT3
$(x + y) \cdot z = (x \cdot z) + (y \cdot z)$	A4	$\sigma_{\mathsf{rel}}^p(x) \cdot y = \sigma_{\mathsf{rel}}^p(x \cdot y)$	SRT4
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5		
$x + \tilde{\tilde{\delta}} = x$	A6SR		
$\tilde{\tilde{\delta}} \cdot x = \tilde{\tilde{\delta}}$	A7SR		

It is assumed that a fixed but arbitrary set A of *actions* has been given. We write A_δ for $\mathsf{A} \cup \{\delta\}$. It is also assumed that a fixed but arbitrary *communication function*, i.e. a partial commutative and associative function $\gamma : \mathsf{A} \times \mathsf{A} \to \mathsf{A}$, 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 be undefined otherwise.

We shall henceforth use $x, y, x', y', \ldots$ as variables ranging over processes. Furthermore, we shall henceforth use $a, b, c, \ldots$ to stand for arbitrary elements of A_δ in the context of equations and for arbitrary elements of A in the context of transition rules (unless explicitly indicated otherwise), $p, q, r, \ldots$ to stand for arbitrary closed terms denoting non-negative real numbers, and $\xi, \xi', \xi'', \ldots$ to stand for arbitrary closed terms denoting points in $\mathbb{R}^3$. We write ULA for the set $\{\tilde{\tilde{a}}(\xi) \mid a \in \mathsf{A}, \xi \in \mathbb{R}^3\}$ of undelayable located actions and ULA_δ for $\mathsf{ULA} \cup \{\tilde{\tilde{\delta}}\}$. We shall henceforth use $\alpha, \beta, \ldots$ to stand for arbitrary elements of ULA_δ. Let $H \subseteq \mathsf{A}$. Then we write $\mathrm{UL}(H)$ for $\{\tilde{\tilde{a}}(\xi) \in \mathsf{ULA} \mid a \in H\}$.

In the axiom system of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$, like in all ACP-style axiom systems, we distinguish a subsystem, called $\mathrm{BPA}_{\mathrm{la}}^{\mathrm{srt}}$, which does not cover parallelism and communication.

The axiom system of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ consists of the equations given in Tables 1 and 2. Many axioms in these tables and coming ones are actually axiom schemas. In Tables 1 and 2, for example, α and β stand for arbitrary members of ULA_δ, a, b and c stand for arbitrary members of A_δ, ξ and ξ' stand for arbitrary closed terms denoting members of $\mathbb{R}^3$, p and q stand for arbitrary closed terms denoting members of $\mathbb{R}_{\geq 0}$, and r stands for an arbitrary closed term denoting a member of $\mathbb{R}_{>0}$. Axioms A1–A5 are the axioms of BPA. Axioms A6SR and A7SR are simple reformulations of axioms A6 and A7 of BPA_δ: δ has been replaced by $\tilde{\tilde{\delta}}$. For a detailed introduction to BPA and BPA_δ, see [8]. Axioms SRT1 and SRT2 point out that a delay of 0 time units has no effect and that consecutive delays count up. Axiom SRT3, called the time-factorization axiom, shows that a delay by itself cannot determine a choice. Axiom SRT4 reflects that timing is relative. Axioms CM1, CM4, CM8, CM9, D3 and D4 are in common with the additional axioms for ACP. Axioms CM2SRLA, CM3SRLA, CM5SRLA–CM7SRLA, CF1SRLA, CF2SRLA, D1SRLA and D2SRLA are simple reformulations of axioms CM2, CM3, CM5–CM7, CF1, CF2, D1 and D2 of ACP: a, b, c and δ have been replaced by α or $\tilde{\tilde{a}}(\xi)$, β or $\tilde{\tilde{b}}(\xi)$, $\tilde{\tilde{c}}(\xi)$ and $\tilde{\tilde{\delta}}$, respectively. The use of α and β is meant to simplify matters. Axioms SRCM1a, SRCM1b, SRCM2–SRCM5 and SRD are new axioms concerning the interaction of relative delay with left merge, communication merge and encapsulation. Axioms SRU1LA and SRU2–SRU4 make clear that relative undelayable time-out prevents a process from idling at the start.

The axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ are the axioms of $\mathrm{ACP}^{\mathrm{srt}}$ without the deadlocked process (see [6]), but with $\tilde{\tilde{a}}$ replaced by α or $\tilde{\tilde{a}}(\xi)$, $\tilde{\tilde{b}}$ replaced by β or $\tilde{\tilde{b}}(\xi)$ and $\tilde{\tilde{c}}$ replaced by $\tilde{\tilde{c}}(\xi)$, and on top of that axioms

Table 2. Additional axioms for $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ ($\alpha, \beta \in \mathsf{ULA}_\delta$, $a, b, c \in \mathsf{A}$, $\xi, \xi' \in \mathbb{R}^3$, $r > 0$)

$x \parallel y = x \between y + y \between x + x \mid y$	CM1
$\alpha \between x = \alpha \cdot x$	CM2SRLA
$\alpha \cdot x \between y = \alpha \cdot (x \parallel y)$	CM3SRLA
$\sigma^r_{\mathsf{rel}}(x) \between \nu_{\mathsf{rel}}(y) = \tilde{\tilde{\delta}}$	SRCM1a
$\sigma^r_{\mathsf{rel}}(x) \between (\nu_{\mathsf{rel}}(y) + z) = \sigma^r_{\mathsf{rel}}(x) \between z$	SRCM1b
$\sigma^r_{\mathsf{rel}}(x) \between \sigma^r_{\mathsf{rel}}(y) = \sigma^r_{\mathsf{rel}}(x \between y)$	SRCM2
$(x + y) \between z = x \between z + y \between z$	CM4
$\alpha \cdot x \mid \beta = (\alpha \mid \beta) \cdot x$	CM5SRLA
$\alpha \mid \beta \cdot x = (\alpha \mid \beta) \cdot x$	CM6SRLA
$\alpha \cdot x \mid \beta \cdot y = (\alpha \mid \beta) \cdot (x \parallel y)$	CM7SRLA
$\nu_{\mathsf{rel}}(x) \mid \sigma^r_{\mathsf{rel}}(y) = \tilde{\tilde{\delta}}$	SRCM3
$\sigma^r_{\mathsf{rel}}(x) \mid \nu_{\mathsf{rel}}(y) = \tilde{\tilde{\delta}}$	SRCM4
$\sigma^r_{\mathsf{rel}}(x) \mid \sigma^r_{\mathsf{rel}}(y) = \sigma^r_{\mathsf{rel}}(x \mid y)$	SRCM5
$(x + y) \mid z = x \mid z + y \mid z$	CM8
$x \mid (y + z) = x \mid y + x \mid z$	CM9
$\tilde{\tilde{a}}(\xi) \mid \tilde{\tilde{\delta}} = \tilde{\tilde{\delta}}$	LACF1
$\tilde{\tilde{\delta}} \mid \tilde{\tilde{a}}(\xi) = \tilde{\tilde{\delta}}$	LACF2
$\tilde{\tilde{a}}(\xi) \mid \tilde{\tilde{b}}(\xi) = \tilde{\tilde{c}}(\xi)$ if $\gamma(a, b) = c$	CF1SRLA
$\tilde{\tilde{a}}(\xi) \mid \tilde{\tilde{b}}(\xi) = \tilde{\tilde{\delta}}$ if $\gamma(a, b)$ undefined	CF2SRLA
$\xi \neq \xi' \Rightarrow \tilde{\tilde{a}}(\xi) \mid \tilde{\tilde{b}}(\xi') = \tilde{\tilde{\delta}}$	LACF3
$\partial_H(\alpha) = \alpha$ if $\alpha \notin \mathrm{UL}(H)$	D1SRLA
$\partial_H(\alpha) = \tilde{\tilde{\delta}}$ if $\alpha \in \mathrm{UL}(H)$	D2SRLA
$\partial_H(\sigma^r_{\mathsf{rel}}(x)) = \sigma^r_{\mathsf{rel}}(\partial_H(x))$	SRD
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3
$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$	D4
$\nu_{\mathsf{rel}}(\alpha) = \alpha$	SRU1LA
$\nu_{\mathsf{rel}}(\sigma^r_{\mathsf{rel}}(x)) = \tilde{\tilde{\delta}}$	SRU2
$\nu_{\mathsf{rel}}(x + y) = \nu_{\mathsf{rel}}(x) + \nu_{\mathsf{rel}}(y)$	SRU3
$\nu_{\mathsf{rel}}(x \cdot y) = \nu_{\mathsf{rel}}(x) \cdot y$	SRU4

LACF1–LACF3. In the process algebras with timing from [6], axioms like LACF1 and LACF2 are not needed. The counterparts of these axioms are special cases of the counterparts of axiom CF2SRLA. In $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$, it is not like that because actions are located but deadlock is not located. Axiom LACF3 is the crucial axiom of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$. It points out that two actions cannot be performed synchronously if they are to be performed at different locations.

The need to use parentheses is reduced by using the associativity of the operators $+$, $\cdot$ and $\parallel$, and

Table 3. Transition rules for $\mathrm{BPA}_{\mathrm{la}}^{\mathrm{srt}}$ ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r, s > 0$)

$$\frac{}{\tilde{\tilde{a}}(\xi) \xrightarrow{a(\xi)} \surd} \qquad \frac{x \xrightarrow{a(\xi)} x'}{\sigma^0_{\mathsf{rel}}(x) \xrightarrow{a(\xi)} x'} \qquad \frac{x \xrightarrow{a(\xi)} \surd}{\sigma^0_{\mathsf{rel}}(x) \xrightarrow{a(\xi)} \surd} \qquad \frac{x \overset{r}{\longmapsto} x'}{\sigma^0_{\mathsf{rel}}(x) \overset{r}{\longmapsto} x'}$$

$$\frac{}{\sigma^{r+s}_{\mathsf{rel}}(x) \overset{r}{\longmapsto} \sigma^s_{\mathsf{rel}}(x)} \qquad \frac{}{\sigma^r_{\mathsf{rel}}(x) \overset{r}{\longmapsto} x} \qquad \frac{x \overset{s}{\longmapsto} x'}{\sigma^r_{\mathsf{rel}}(x) \overset{r+s}{\longmapsto} x'}$$

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

$$\frac{x \overset{r}{\longmapsto} x',\ y \overset{r}{\not\longmapsto}}{x + y \overset{r}{\longmapsto} x'} \qquad \frac{x \overset{r}{\not\longmapsto},\ y \overset{r}{\longmapsto} y'}{x + y \overset{r}{\longmapsto} y'} \qquad \frac{x \overset{r}{\longmapsto} x',\ y \overset{r}{\longmapsto} y'}{x + y \overset{r}{\longmapsto} x' + y'}$$

$$\frac{x \xrightarrow{a(\xi)} x'}{x \cdot y \xrightarrow{a(\xi)} x' \cdot y} \qquad \frac{x \xrightarrow{a(\xi)} \surd}{x \cdot y \xrightarrow{a(\xi)} y} \qquad \frac{x \overset{r}{\longmapsto} x'}{x \cdot y \overset{r}{\longmapsto} x' \cdot y}$$

by ranking the precedence of the binary operators. Throughout this paper we adhere to the following precedence rules: (i) the operator $+$ has lower precedence than all others, (ii) the operator $\cdot$ has higher precedence than all others, (iii) all other operators have the same precedence. The commutativity and associativity of the operator $+$ permit the use of the notation $\sum_{i \in \mathcal{I}} t_i$, where $\mathcal{I} = \{i_1, \ldots, i_n\}$, for $t_{i_1} + \ldots + t_{i_n}$. We further use the convention that $\sum_{i \in \mathcal{I}} t_i$ stands for $\tilde{\tilde{\delta}}$ if $\mathcal{I} = \emptyset$.

The structural operational semantics of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ is described by the transition rules given in Tables 3 and 4. The following transition relations are used:

- a binary *action step* relation $\_ \xrightarrow{a(\xi)} \_$ for each $a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$;
- a unary *action termination* relation $\_ \xrightarrow{a(\xi)} \surd$ for each $a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$;
- a binary *time step* relation $\_ \overset{r}{\longmapsto} \_$ for each $r \in \mathbb{R}_{>0}$.

We write $t \overset{r}{\not\longmapsto}$ for the set of all transition formulas $\neg(t \overset{r}{\longmapsto} t')$ where t' is a closed term of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$. The three kinds of transition relations can be explained as follows:

$t \xrightarrow{a(\xi)} t'$: process t is capable of first performing action a at location ξ, at the current point of time, and then proceeding as process t';

$t \xrightarrow{a(\xi)} \surd$: process t is capable of first performing action a at location ξ, at the current point of time, and then terminating successfully;

$t \overset{r}{\longmapsto} t'$: process t is capable of first idling for a period of time r and then proceeding as process t'.

The time step relations are defined such that $t \overset{r}{\longmapsto} t'$ and $t \overset{r}{\longmapsto} t''$ only if t' and t'' are the same. In other words, the time steps of a process are combined in the operational semantics of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$.

Table 4. Additional transition rules for $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ ($a, b, c \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$)

$$\frac{x \xrightarrow{a(\xi)} x'}{x \parallel y \xrightarrow{a(\xi)} x' \parallel y} \qquad \frac{y \xrightarrow{a(\xi)} y'}{x \parallel y \xrightarrow{a(\xi)} x \parallel y'} \qquad \frac{x \xrightarrow{a(\xi)} \surd}{x \parallel y \xrightarrow{a(\xi)} y} \qquad \frac{y \xrightarrow{a(\xi)} \surd}{x \parallel y \xrightarrow{a(\xi)} x}$$

$$\frac{x \xrightarrow{a(\xi)} x',\ y \xrightarrow{b(\xi)} y'}{x \parallel y \xrightarrow{c(\xi)} x' \parallel y'}\ \gamma(a,b) = c \qquad \frac{x \xrightarrow{a(\xi)} x',\ y \xrightarrow{b(\xi)} \surd}{x \parallel y \xrightarrow{c(\xi)} x'}\ \gamma(a,b) = c$$

$$\frac{x \xrightarrow{a(\xi)} \surd,\ y \xrightarrow{b(\xi)} y'}{x \parallel y \xrightarrow{c(\xi)} y'}\ \gamma(a,b) = c \qquad \frac{x \xrightarrow{a(\xi)} \surd,\ y \xrightarrow{b(\xi)} \surd}{x \parallel y \xrightarrow{c(\xi)} \surd}\ \gamma(a,b) = c$$

$$\frac{x \xmapsto{r} x',\ y \xmapsto{r} y'}{x \parallel y \xmapsto{r} x' \parallel y'}$$

$$\frac{x \xrightarrow{a(\xi)} x'}{x \between y \xrightarrow{a(\xi)} x' \parallel y} \qquad \frac{x \xrightarrow{a(\xi)} \surd}{x \between y \xrightarrow{a(\xi)} y} \qquad \frac{x \xmapsto{r} x',\ y \xmapsto{r} y'}{x \between y \xmapsto{r} x' \between y'}$$

$$\frac{x \xrightarrow{a(\xi)} x',\ y \xrightarrow{b(\xi)} y'}{x \mid y \xrightarrow{c(\xi)} x' \parallel y'}\ \gamma(a,b) = c \qquad \frac{x \xrightarrow{a(\xi)} x',\ y \xrightarrow{b(\xi)} \surd}{x \mid y \xrightarrow{c(\xi)} x'}\ \gamma(a,b) = c$$

$$\frac{x \xrightarrow{a(\xi)} \surd,\ y \xrightarrow{b(\xi)} y'}{x \mid y \xrightarrow{c(\xi)} y'}\ \gamma(a,b) = c \qquad \frac{x \xrightarrow{a(\xi)} \surd,\ y \xrightarrow{b(\xi)} \surd}{x \mid y \xrightarrow{c(\xi)} \surd}\ \gamma(a,b) = c$$

$$\frac{x \xmapsto{r} x',\ y \xmapsto{r} y'}{x \mid y \xmapsto{r} x' \mid y'}$$

$$\frac{x \xrightarrow{a(\xi)} x'}{\partial_H(x) \xrightarrow{a(\xi)} \partial_H(x')}\ a \notin H \qquad \frac{x \xrightarrow{a(\xi)} \surd}{\partial_H(x) \xrightarrow{a(\xi)} \surd}\ a \notin H \qquad \frac{x \xmapsto{r} x'}{\partial_H(x) \xmapsto{r} \partial_H(x')}$$

$$\frac{x \xrightarrow{a(\xi)} x'}{\nu_{\mathsf{rel}}(x) \xrightarrow{a(\xi)} x'} \qquad \frac{x \xrightarrow{a(\xi)} \surd}{\nu_{\mathsf{rel}}(x) \xrightarrow{a(\xi)} \surd}$$

The transition rules for the operational semantics of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ are the transition rules for the operational semantics of $\mathrm{ACP}^{\mathrm{srt}}$ without the deadlocked process (see [6]), but with a replaced by $a(\xi)$, b replaced by $b(\xi)$ and c replaced by $c(\xi)$. Thus, there are no transition rules allowing actions at different locations to be performed synchronously.

Bisimulation based on the transition rules for $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ is defined as usual. Bisimulation equivalence is a congruence on the algebra of closed terms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$, which follows immediately from the fact that the transition rules for $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ constitute a complete transition system specification in panth format. For more information on the panth format and the related congruence result, see e.g. [1, 17]. The quotient algebra of the algebra of closed terms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ by bisimulation equivalence is a model of the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$.

Theorem 2.1. (Soundness)
All axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ hold in the quotient algebra of the algebra of closed terms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ by bisimulation equivalence.

Models of this kind are called bisimulation models. The axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ make all equations between

closed terms that hold in its bisimulation model derivable.

Theorem 2.2. (Completeness)
All equations between closed terms that hold in the quotient algebra of the algebra of closed terms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ by bisimulation equivalence are derivable from the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$, under the assumption that we can derive all valid formulas about $\mathbb{R}_{\geq 0}$ and $\mathbb{R}^3$ that are needed.

The soundness and completeness proofs for $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ go completely analogous to the soundness and completeness proofs for $\mathrm{ACP}^{\mathrm{srt}}$ without the deadlocked process, except for some minor differences where communication merge is involved. It is worth noticing that $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ is a generalization of $\mathrm{ACP}^{\mathrm{srt}}$.

Theorem 2.3. (Embedding)
$\mathrm{ACP}^{\mathrm{srt}}$ without the deadlocked process can be embedded in $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$.

Proof:
Fix an arbitrary location ξ_0. Consider the function ϵ from the terms of $\mathrm{ACP}^{\mathrm{srt}}$ to the terms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ that simply replaces each occurrence of a constant $\tilde{\tilde{a}}$ by $\tilde{\tilde{a}}(\xi_0)$. Clearly, ϵ is term structure preserving and injective. Moreover, it is easy to see that for all closed terms t and t' of $\mathrm{ACP}^{\mathrm{srt}}$, if $t = t'$ is derivable from the axioms of $\mathrm{ACP}^{\mathrm{srt}}$, then $\epsilon(t) = \epsilon(t')$ is derivable from the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$. □

For more information on the construction of bisimulation models, soundness and completeness proofs for process algebras with timing, and embeddings, see e.g. Appendix B of [6].

We will use the conventions for synchronous communication between two processes that were introduced for ACP in [10]. It is assumed that a fixed but arbitrary set I of ports and a fixed but arbitrary set D_i of data for each $i \in \mathsf{I}$ have been given. It is further assumed that A contains for each $i \in \mathsf{I}$ and $d \in \mathsf{D}_i$ the following special actions:

- $s_i(d)$, the sending of datum d at port i;
- $r_i(d)$, the receiving of datum d at port i;
- $c_i(d)$, the communication of datum d at port i;

and that γ is defined such that for all $i \in \mathsf{I}$, $d \in \mathsf{D}_i$ and $a \in \mathsf{A}$:

$$\begin{aligned}
&\gamma(s_i(d), r_i(d)) = c_i(d)\\
&\gamma(s_i(d), a) \quad \text{undefined} \quad \mathsf{if}\ a \neq r_i(d)\\
&\gamma(a, r_i(d)) \quad \text{undefined} \quad \mathsf{if}\ a \neq s_i(d)\ .
\end{aligned}$$

3. Integration and Recursion

In this section, we extend $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ with integration and guarded recursion. These extensions will be needed in virtually all applications. The use of integration and guarded recursion will be illustrated in Section 6, where we consider a protocol transmitting data via a mobile intermediate station.

Table 5. Axioms for integration ($p \geq 0$)

$\int_{u\in U} F(u) = \int_{u'\in U} F(u')$	INT1
$\int_{u\in\emptyset} F(u) = \tilde{\tilde{\delta}}$	INT2
$\int_{u\in\{p\}} F(u) = F(p)$	INT3
$\int_{u\in U\cup U'} F(u) = \int_{u\in U} F(u) + \int_{u\in U'} F(u)$	INT4
$U \neq \emptyset \Rightarrow \int_{u\in U} x = x$	INT5
$(\forall u \in U \bullet F(u) = G(u)) \Rightarrow \int_{u\in U} F(u) = \int_{u\in U} G(u)$	INT6
U, U' unbounded $\Rightarrow \int_{u\in U} \sigma^u_{\mathsf{rel}}(\tilde{\tilde{\delta}}) = \int_{u\in U'} \sigma^u_{\mathsf{rel}}(\tilde{\tilde{\delta}})$	INT8SR
$\sup U = p,\ p \in U \Rightarrow \int_{u\in U} \sigma^u_{\mathsf{rel}}(\tilde{\tilde{\delta}}) = \sigma^p_{\mathsf{rel}}(\tilde{\tilde{\delta}})$	INT9SR
$\int_{u\in U} \sigma^p_{\mathsf{rel}}(F(u)) = \sigma^p_{\mathsf{rel}}(\int_{u\in U} F(u))$	INT10SR
$\int_{u\in U} (F(u) + G(u)) = \int_{u\in U} F(u) + \int_{u\in U} G(u)$	INT11
$\int_{u\in U} (F(u) \cdot x) = (\int_{u\in U} F(u)) \cdot x$	INT12
$\int_{u\in U} \nu_{\mathsf{rel}}(F(u)) = \nu_{\mathsf{rel}}(\int_{u\in U} F(u))$	INT13

3.1. Integration

In order to cover processes that are capable of performing an action at all points in a certain time interval, we add integration to $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$. Integration is represented by the variable-binding operator $\int$. Let P be an expression, possibly containing variable u, such that $P[p/u]$ (P with p substituted for u) represents a process for all $p \in \mathbb{R}_{\geq 0}$; and let $U \subseteq \mathbb{R}_{\geq 0}$. Then the *integration* $\int_{u\in U} P$ behaves like one of the processes $P[p/u]$ for $p \in U$. Hence, integration is a form of alternative composition over a set of alternatives that may even be a continuum.

We shall henceforth use F and G as variables ranging over functions that map each $p \in \mathbb{R}_{\geq 0}$ to a process and can be represented by terms containing a designated free variable ranging over $\mathbb{R}_{\geq 0}$. For more information on such second-order variables, see e.g. [15, 17]. Furthermore, we shall henceforth use $u, u', \ldots$ as variables ranging over $\mathbb{R}_{\geq 0}$. It is assumed that each first-order definable set of non-negative real numbers can be denoted by a closed term, and we shall henceforth use $U, U', \ldots$ to stand for arbitrary closed terms denoting first-order definable sets of non-negative real numbers.

The additional axioms for integration are the equations given in Table 5. Axiom INT1 is similar to the α-conversion rule of λ-calculus. Axioms INT2–INT4 show that integration is a form of alternative composition over a set of alternatives. Axiom INT5 can be regarded as the counterpart of axiom A3 for integration. Axiom INT6 is an extensionality axiom. The remaining axioms are easily understood by realizing that integration is a form of alternative composition over a set of alternatives. Axioms INT10SR, INT11, INT12 and INT13 can simply be regarded as variants of axioms SRT3, A2, A4 and SRU3, respectively. Axioms INT8SR and INT9SR are both reminiscent of the equation $\sigma^{p+q}_{\mathsf{rel}}(\tilde{\tilde{\delta}}) + \sigma^p_{\mathsf{rel}}(\tilde{\tilde{\delta}}) = \sigma^{p+q}_{\mathsf{rel}}(\tilde{\tilde{\delta}})$, which is derivable from axioms A6SR, SRT2 and SRT3.

The structural operational semantics for integration is described by the transition rules given in Table 6. The complexity of the transition rule concerning the time-related capabilities of a process $\int_{u\in U} F(u)$ is caused by the fact that the processes $F(p)$ with $p \in U$ that are capable of idling need not change uniformly while idling. For more information on this phenomenon, see e.g. [6, 18]. A bisimulation model of the axioms of $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$ with integration can be constructed in the same way as for $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$.

Table 6. Transition rules for integration ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $p, q \geq 0$, $r > 0$)

$$\frac{F(p) \xrightarrow{a(\xi)} x'}{\int_{u \in U} F(u) \xrightarrow{a(\xi)} x'} \; p \in U \qquad \frac{F(p) \xrightarrow{a(\xi)} \surd}{\int_{u \in U} F(u) \xrightarrow{a(\xi)} \surd} \; p \in U$$

$$\frac{\begin{array}{c}\{F(q) \overset{r}{\mapsto} F_1(q) \mid q \in U_1\},\\ \ldots,\\ \{F(q) \overset{r}{\mapsto} F_n(q) \mid q \in U_n\},\\ \{F(q) \overset{r}{\not\mapsto} \mid q \in U_{n+1}\}\end{array}}{\int_{u \in U} F(u) \overset{r}{\mapsto} \int_{u \in U_1} F_1(u) + \ldots + \int_{u \in U_n} F_n(u)} \; \begin{array}{l}\{U_1, \ldots, U_n\} \text{ partition of } U \setminus U_{n+1},\\ U_{n+1} \subset U\end{array}$$

Table 7. Axioms for recursion

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

3.2. Guarded Recursion

In order to allow for the description of (potentially) non-terminating processes, we add guarded recursion to $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$.

A *recursive specification* over $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ is a set of *recursive* equations $E = \{X = t_X \mid X \in V\}$ where V is a set of variables and each t_X is a term of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ that only contains variables from V. We write $\mathrm{V}(E)$ for the set of all variables that occur on the left-hand side of an equation in E. A *solution* of a recursive specification E is a set of processes (in some model of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$) $\{P_X \mid X \in \mathrm{V}(E)\}$ such that the equations of E hold if, for all $X \in \mathrm{V}(E)$, X stands for P_X.

Let t be a term of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ containing a variable X. We call an occurrence of X in t *guarded* if t has a subterm of the form $\tilde{\tilde{a}} \cdot t'$ or $\sigma_{\mathsf{rel}}^r(t')$, where $a \in \mathsf{A}$, $r \in \mathbb{R}_{>0}$ and t' a term of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$, with t' containing this occurrence of X. A recursive specification over $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ is called a *guarded* recursive specification if all occurrences of variables in the right-hand sides of its equations are guarded or it can be rewritten to such a recursive specification using the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ and the equations of the recursive specification. A guarded recursive specification has a unique solution.

For each guarded recursive specification E and each variable $X \in \mathrm{V}(E)$, we introduce a constant $\langle X \mid E \rangle$ which is interpreted as the unique solution of E for X. We often write X for $\langle X \mid E \rangle$ if E is clear from the context. In such cases, it should also be clear from the context that we use X as a constant.

We will also use the following notation. Let t be a term of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ and E be a guarded recursive specification. Then we write $\langle t \mid E \rangle$ for t with, for all $X \in \mathrm{V}(E)$, all occurrences of X in t replaced by $\langle X \mid E \rangle$.

We shall henceforth use $X, Y, \ldots$ as variables ranging over processes in the case where they occur in a recursive specification. Furthermore, we shall henceforth use $t_X, t_Y, \ldots$ to stand for arbitrary terms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ in the case where they occur in a recursive specification, and $E, E', \ldots$ to stand for arbitrary guarded recursive specifications.

The additional axioms for recursion are the equations given in Table 7. A side condition is added to restrict the variables, terms and guarded recursive specifications for which X, t_X and E stand. The additional axioms for recursion are known as the recursive definition principle (RDP) and the recursive

Table 8. Transition rules for recursion ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$)

$$\frac{\langle t_X|E\rangle \xrightarrow{a(\xi)} x'}{\langle X|E\rangle \xrightarrow{a(\xi)} x'} \; X = t_X \in E \qquad \frac{\langle t_X|E\rangle \xrightarrow{a(\xi)} \surd}{\langle X|E\rangle \xrightarrow{a(\xi)} \surd} \; X = t_X \in E$$

$$\frac{\langle t_X|E\rangle \overset{r}{\longmapsto} x'}{\langle X|E\rangle \overset{r}{\longmapsto} x'} \; X = t_X \in E$$

specification principle (RSP). 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.

It is sometimes helpful to rewrite guarded recursive specifications. The following useful fact about the rewriting of guarded recursive specifications can be proven. Let E and E' be two guarded recursive specifications over $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$, where E' is E rewritten using the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ and the equations of E. Then the equation $\langle X|E\rangle = \langle X|E'\rangle$ is derivable for all $X \in \mathrm{V}(E)$.

The structural operational semantics for recursion is described by the transition rules given in Table 8. A bisimulation model of the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ with integration and guarded recursion can be constructed in the same way as for $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$.

4. Action Renaming and Spatial Replacement

In this section, we extend $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ with action renaming and spatial replacement. These extensions will be needed in many applications. In common with the use of integration and guarded recursion, the use of action renaming and spatial replacement will be illustrated in Section 6.

4.1. Action Renaming

Action renaming provides for change of actions. It facilitates dealing with a number of processes that only differ in the ports and/or channels used for communication.[1] Let P be a process and $f : \mathsf{A} \to \mathsf{A}$. Then the *action renaming* of P according to f, written $\rho_f(P)$, behaves like P, but with located actions $\tilde{\tilde{a}}(\xi)$ replaced by $\widetilde{\widetilde{f(a)}}(\xi)$.

The additional axioms for action renaming are given in Table 9. In this table, we use a to stand for elements of A. The axioms for action renaming do not need further explanation.

The structural operational semantics for action renaming is described by the transition rules given in Table 10. A bisimulation model of the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ with integration and action renaming, with or without guarded recursion, can be constructed in the same way as for $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$.

4.2. Spatial Replacement

Spatial replacement facilitates dealing with a number of processes that only differ in the locations at which they perform their actions. It provides for time-dependent change of locations. In this way, the behaviour of processes that move in space can be described. Let P be a process and $f : \mathbb{R}^3 \times \mathbb{R}_{\geq 0} \to \mathbb{R}^3$

[1] In Section 6.1, we introduce channels for asynchronous communication in space.

Table 9. Axioms for action renaming ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$)

$\rho_f(\tilde{\tilde{\delta}}) = \tilde{\tilde{\delta}}$	LARN1
$\rho_f(\tilde{\tilde{a}}(\xi)) = \widetilde{\widetilde{f(a)}}(\xi)$	LARN2
$\rho_f(\tilde{\tilde{a}}(\xi) \cdot x) = \widetilde{\widetilde{f(a)}}(\xi) \cdot \rho_f(x)$	LARN3
$\rho_f(\sigma^r_{\mathsf{rel}}(x)) = \sigma^r_{\mathsf{rel}}(\rho_f(x))$	LARN4
$\rho_f(x + y) = \rho_f(x) + \rho_f(y)$	LARN5
$\rho_f(\int_{u \in U} F(u)) = \int_{u \in U} \rho_f(F(u))$	LARN6

Table 10. Transition rules for action renaming ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$)

$\dfrac{x \xrightarrow{a(\xi)} x'}{\rho_f(x) \xrightarrow{f(a)(\xi)} \rho_f(x')}$	$\dfrac{x \xrightarrow{a(\xi)} \surd}{\rho_f(x) \xrightarrow{f(a)(\xi)} \surd}$	$\dfrac{x \overset{r}{\longmapsto} x'}{\rho_f(x) \overset{r}{\longmapsto} \rho_f(x')}$

Table 11. Axioms for spatial replacement ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$)

$\vec{\rho}_f(\tilde{\tilde{\delta}}) = \tilde{\tilde{\delta}}$	LASR1
$\vec{\rho}_f(\tilde{\tilde{a}}(\xi)) = \tilde{\tilde{a}}(f(\xi, 0))$	LASR2
$\vec{\rho}_f(\tilde{\tilde{a}}(\xi) \cdot x) = \tilde{\tilde{a}}(f(\xi, 0)) \cdot \vec{\rho}_f(x)$	LASR3
$\vec{\rho}_f(\sigma^r_{\mathsf{rel}}(x)) = \sigma^r_{\mathsf{rel}}(\vec{\rho}_{f+r}(x))$	LASR4
$\vec{\rho}_f(x + y) = \vec{\rho}_f(x) + \vec{\rho}_f(y)$	LASR5
$\vec{\rho}_f(\int_{u \in U} F(u)) = \int_{u \in U} \vec{\rho}_f(F(u))$	LASR6

be a continuous function. Then the *spatial replacement* of P according to f, written $\vec{\rho}_f(P)$, behaves like P, but performing located actions $\tilde{\tilde{a}}(\xi)$ at location $f(\xi, t)$, where t is the time passed since the start of P, instead of location ξ. Spatial replacement is reminiscent of action renaming.

As an example, we consider the process of which the recursive specification consists of the equation $X = \tilde{\tilde{a}}(\xi_1) \cdot \sigma^1_{\mathsf{rel}}(X)$ and the replacement function f defined by $f(\xi, t) = \xi + \upsilon \cdot t + \xi_0 - \xi_1$, where υ is a velocity vector. Now the process $\vec{\rho}_f(X)$ behaves as

$$\tilde{\tilde{a}}(\xi_0) \cdot \sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi_0 + 1\upsilon)) \cdot \sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi_0 + 2\upsilon)) \cdot \sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}}(\xi_0 + 3\upsilon)) \cdot \ \cdots \ .$$

Note that $+$ stands here for addition of vectors instead of alternative composition. From now on, it depends on the context in which $+$ occurs whether it stands for addition or alternative composition.

The additional axioms for spatial replacement are given in Table 11. In this table, we use a to stand for elements of A. In axiom LASR4, we write $f + r$ for the function $f' : \mathbb{R}^3 \times \mathbb{R}_{\geq 0} \to \mathbb{R}^3$ such that, for all $\xi \in \mathbb{R}^3$ and $p \in \mathbb{R}_{\geq 0}$, $f'(\xi, p) = f(\xi, p + r)$. Only axiom LASR4 may need further explanation: the absolute time is taken into account by changing the replacement function f with each relative delay.

The structural operational semantics for spatial replacement is described by the transition rules given in Table 12. A bisimulation model of the axioms of $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$ with integration, action renaming and spatial replacement, with or without guarded recursion, can be constructed in the same way as for $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$.

Table 12. Transition rules for spatial replacement ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$)

$x \xrightarrow{a(\xi)} x'$	$x \xrightarrow{a(\xi)} \surd$	$x \overset{r}{\mapsto} x'$
$\vec{\rho}_f(x) \xrightarrow{a(f(\xi,0))} \vec{\rho}_f(x')$	$\vec{\rho}_f(x) \xrightarrow{a(f(\xi,0))} \surd$	$\vec{\rho}_f(x) \overset{r}{\mapsto} \vec{\rho}_{f+r}(x')$

5. State Operator and Maximal Progress Operator

In this section, we extend $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ with a state operator and a maximal progress operator. The state operator enables processes to interact with a state. The maximal progress operator makes it possible to give performing certain actions priority over idling. Both extensions are useful in modeling communication between processes at different places in space. Just like the extensions of Sections 3 and 4, the use of these extensions will be illustrated in Section 6.

5.1. State Operator

The state operator makes it easy to represent the execution of a process in a state. The basic idea is that the execution of an action in a state has effect on the state, i.e. it causes a discrete change of state. Moreover, there is an action left when an action is executed in a state. For example, in case the states are queues of data, when the action of instructing the addition or removal of a certain datum is executed in a state, the action of adding or removing that datum is left. The operator introduced here generalizes the state operator added to ACP without timing in [8]. The main difference with that operator is that idling may cause a continuous change of state.

It is assumed that a fixed but arbitrary set S of *states* has been given, together with functions $\mathrm{act} : \mathsf{A} \times \mathbb{R}^3 \times \mathsf{S} \to \mathsf{A}_\delta$, $\mathrm{eff} : \mathsf{A} \times \mathbb{R}^3 \times \mathsf{S} \to \mathsf{S}$ and $\mathrm{eff}' : \mathbb{R}_{>0} \times \mathsf{S} \to \mathsf{S}$ such that $\mathrm{eff}'(r + r', s) = \mathrm{eff}'(r, \mathrm{eff}'(r', s))$ for all $r, r' \in \mathbb{R}_{>0}$ and $s \in \mathsf{S}$.

The state operator λ_s ($s \in \mathsf{S}$) allows, given the functions act, eff and eff', processes to interact with a state. Let P be a process. Then $\lambda_s(P)$ is the process P executed in state s. The function act gives, for each action a, location ξ and state s, the action that results from executing a in state s at location ξ. The function eff gives, for each action a, location ξ and state s, the state that results from executing a in state s at location ξ. The function eff' gives, for each period of time r and state s, the state that results from idling for a period of time r from state s. The restriction on function eff' stems from axiom SRT2.

As an example, we consider the process of which the recursive specification consists of the equation $X = \tilde{\tilde{a}}(\xi_1) \cdot \sigma^1_{\mathsf{rel}}(X)$, and a state operator with $\mathbb{N}$ as set of states and the functions act, eff and eff' defined such that $\mathrm{act}(a, \xi, n) = a_n$, $\mathrm{eff}(a, \xi, n) = n + 1$ and $\mathrm{eff}'(r, n) = n$. Now the process $\lambda_0(X)$ behaves as

$$\tilde{\tilde{a}}_0(\xi_1) \cdot \sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}}_1(\xi_1)) \cdot \sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}}_2(\xi_1)) \cdot \sigma^1_{\mathsf{rel}}(\tilde{\tilde{a}}_3(\xi_1)) \cdot \ \cdots \ .$$

The additional axioms for the *state* operator λ_s ($s \in \mathsf{S}$) are given in Table 13. In this table, we use a to stand for elements of A. The axioms for the state operator reflect the intended meaning of the state operator clearly. Except for axiom LASO4, they are simple reformulations of the axioms for the state operator added to ACP without timing in [8]: roughly speaking, a has been replaced by $\tilde{\tilde{a}}(\xi)$.

The structural operational semantics of the state operator is described by the transition rules given in Table 14. A bisimulation model of the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ with integration and the state operator,

Table 13. Axioms for state operator ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$, $s \in \mathsf{S}$)

$\lambda_s(\tilde{\delta}) = \tilde{\delta}$	LASO1
$\lambda_s(\tilde{a}(\xi)) = \widetilde{\mathrm{act}(a,\xi,s)}(\xi)$	LASO2
$\lambda_s(\tilde{a}(\xi) \cdot x) = \widetilde{\mathrm{act}(a,\xi,s)}(\xi) \cdot \lambda_{\mathrm{eff}(a,\xi,s)}(x)$	LASO3
$\lambda_s(\sigma^r_{\mathsf{rel}}(x)) = \sigma^r_{\mathsf{rel}}(\lambda_{\mathrm{eff}'(r,s)}(x))$	LASO4
$\lambda_s(x + y) = \lambda_s(x) + \lambda_s(y)$	LASO5
$\lambda_s(\int_{u\in U} F(u)) = \int_{u\in U} \lambda_s(F(u))$	LASO6

Table 14. Transition rules for state operator ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$, $s \in \mathsf{S}$)

$$\frac{x \xrightarrow{a(\xi)} x'}{\lambda_s(x) \xrightarrow{\mathrm{act}(a,\xi,s)(\xi)} \lambda_{\mathrm{eff}(a,\xi,s)}(x')} \ \mathrm{act}(a,\xi,s) \neq \delta \qquad \frac{x \xrightarrow{a(\xi)} \surd}{\lambda_s(x) \xrightarrow{\mathrm{act}(a,\xi,s)(\xi)} \surd} \ \mathrm{act}(a,\xi,s) \neq \delta$$

$$\frac{x \overset{r}{\longmapsto} x'}{\lambda_s(x) \overset{r}{\longmapsto} \lambda_{\mathrm{eff}'(r,s)}(x')}$$

with or without action renaming and spatial replacement, and with or without guarded recursion, can be constructed in the same way as for $\mathrm{ACP}^{\mathrm{srt}}_{\mathrm{la}}$.

5.2. Maximal Progress Operator

The maximal progress operator is a means to express that certain actions must take place as soon as possible. In case performing one of these actions and idling are both options, performing the action gets priority over idling. Let P be a process and H be a set of actions. Then P with *maximal progress* for H, written $\theta_H(P)$, behaves like P, but performing actions $a \in H$ is given priority over idling. The maximal progress operator is a variant of the priority operator added to ACP without timing in [8]. The main difference with that operator is that performing an action can have priority over idling instead of priority over performing certain other actions.

As an example, we consider the process $\theta_{\{a\}}(\int_{u\in[p,\infty)} \sigma^u_{\mathsf{rel}}(\tilde{a}(\xi)))$. This process behaves the same as the process $\sigma^p_{\mathsf{rel}}(\tilde{a}(\xi))$.

The additional axioms for the maximal progress operator are given in Table 15. Axioms LAMP2 and LAMP4–LAMP10 are reformulations of the axioms for the maximal progress operator added to a version of ACP for hybrid systems in [11]. They are adapted to the absence of conditional proceeding, signal emission, signal evolution and signal transition. Axiom LAMP8 makes use of the auxiliary *relative delayable time-out* operator ν_{rel} axiomatized by the equations given in Table 16. These axioms make clear that this operator generalizes the relative undelayable time-out operator: it keeps a process from idling for a period of time longer than a certain period of time. Obviously, we have that the equation $\nu_{\mathsf{rel}}(t) = \nu^0_{\mathsf{rel}}(t)$ is derivable for all closed terms t. Note that the operator differs from the operator υ_{rel} from [6]: the latter keeps a process from idling for a period of time longer than or equal to a certain period of time. The condition in axiom LAMP8 expresses that the equation only applies to processes y that cannot perform any action for a period of time that is longer than r. Axioms LAMP9 and LAMP10 are additional axioms for integration. The condition in axiom LAMP9 expresses that the equation only

Table 15. Axioms for maximal progress operator ($\alpha \in \mathsf{ULA}_\delta$, $r > 0$)

$\theta_H(\tilde{\tilde{\delta}}) = \tilde{\tilde{\delta}}$		LAMP2
$\theta_H(\alpha + x) = \alpha + \theta_H(x)$	if $a \notin \mathrm{UL}(H)$	LAMP4
$\theta_H(\alpha \cdot x + y) = \alpha \cdot \theta_H(x) + \theta_H(y)$	if $a \notin \mathrm{UL}(H)$	LAMP5
$\theta_H(\alpha + x) = \alpha + \theta_H(\nu_{\mathsf{rel}}(x))$	if $a \in \mathrm{UL}(H)$	LAMP6
$\theta_H(\alpha \cdot x + y) = \alpha \cdot \theta_H(x) + \theta_H(\nu_{\mathsf{rel}}(y))$	if $a \in \mathrm{UL}(H)$	LAMP7
$\nu^r_{\mathsf{rel}}(y) = \tilde{\tilde{\delta}} \Rightarrow \theta_H(\sigma^r_{\mathsf{rel}}(x) + y) = \sigma^r_{\mathsf{rel}}(\theta_H(x)) + \theta_H(y)$		LAMP8
$\partial_{\mathsf{A}\setminus H}(\nu_{\mathsf{rel}}(\int_{u\in U} F(u))) = \tilde{\tilde{\delta}} \Rightarrow \theta_H(\int_{u\in U} F(u)) = \int_{u\in U} \theta_H(F(u))$		LAMP9
$\int_{u\in U} F(u) = \nu_{\mathsf{rel}}(\int_{u\in U} F(u)) \Rightarrow \theta_H(\int_{u\in U} F(u)) = \int_{u\in U} \theta_H(F(u))$		LAMP10

Table 16. Axioms for relative delayable time-out operator ($\alpha \in \mathsf{ULA}_\delta$, $p \geq 0$, $r > 0$)

$\nu^p_{\mathsf{rel}}(\alpha) = \alpha$	SRD1
$\nu^0_{\mathsf{rel}}(\sigma^r_{\mathsf{rel}}(x)) = \tilde{\tilde{\delta}}$	SRD2
$\nu^{p+r}_{\mathsf{rel}}(\sigma^r_{\mathsf{rel}}(x)) = \sigma^r_{\mathsf{rel}}(\nu^p_{\mathsf{rel}}(x))$	SRD3
$\nu^p_{\mathsf{rel}}(x + y) = \nu^p_{\mathsf{rel}}(x) + \nu^p_{\mathsf{rel}}(y)$	SRD4
$\nu^p_{\mathsf{rel}}(x \cdot y) = \nu^p_{\mathsf{rel}}(x) \cdot y$	SRD5
$\int_{u\in U} \nu^p_{\mathsf{rel}}(F(u)) = \nu^p_{\mathsf{rel}}(\int_{u\in U} F(u))$	INT14

Table 17. Transition rules for maximal progress operator ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $r > 0$)

$$\frac{x \xrightarrow{a(\xi)} x'}{\theta_H(x) \xrightarrow{a(\xi)} \theta_H(x')} \qquad \frac{x \xrightarrow{a(\xi)} \surd}{\theta_H(x) \xrightarrow{a(\xi)} \surd} \qquad \frac{x \overset{r}{\longmapsto} x',\ \mathcal{F}^H_r(x)}{\theta_H(x) \overset{r}{\longmapsto} \theta_H(x')}$$

Table 18. Transition rules for relative delayable time-out ($a \in \mathsf{A}$, $\xi \in \mathbb{R}^3$, $p \geq 0$, $r > 0$)

$$\frac{x \xrightarrow{a(\xi)} x'}{\nu^p_{\mathsf{rel}}(x) \xrightarrow{a(\xi)} x'} \qquad \frac{x \xrightarrow{a(\xi)} \surd}{\nu^p_{\mathsf{rel}}(x) \xrightarrow{a(\xi)} \surd} \qquad \frac{x \overset{r}{\longmapsto} x'}{\nu^{p+r}_{\mathsf{rel}}(x) \overset{r}{\longmapsto} \nu^p_{\mathsf{rel}}(x')}$$

applies to processes $\int_{u\in U} F(u)$ that cannot perform any action from H at the current point of time. The condition in axiom LAMP10 expresses that the equation only applies to processes $\int_{u\in U} F(u)$ that cannot idle at the current point of time.

The structural operational semantics for the maximal progress operator and the relative delayable time-out operator are described by the transition rules given in Tables 17 and 18, respectively. Additional transition relations $\mathcal{F}^H_r(\_)$, for $H \subseteq \mathsf{A}$ and $r \in \mathbb{R}_{>0}$, are used in Table 17. These auxiliary *failure* relations can be explained as follows:

$\mathcal{F}^H_r(t)$: process t is not capable of performing an action from H before r time units are past.

Transition rules defining these relations can be given in the format used for all transition rules given in this paper, viz. the panth format from [23]. However, we will not give those transition rules here.

Instead, we confine ourselves to give a property that characterizes the failure relations in terms of the other transition relations. We have for all closed terms t, $H \subseteq \mathsf{A}$, and $r \in \mathbb{R}_{>0}$:

$$\begin{aligned}
&\mathcal{F}_r^H(t) \text{ iff}\\
&\text{for all } a \in H, \xi \in \mathbb{R}^3 :\ t \not\xrightarrow{a(\xi)} \text{ and}\\
&\text{for all closed terms } t',\ s < r :\\
&\quad \text{if } t \overset{s}{\mapsto} t' \text{, then for all } a \in H, \xi \in \mathbb{R}^3 :\ t' \not\xrightarrow{a(\xi)} .
\end{aligned}$$

We write $t \not\xrightarrow{a(\xi)}$ for the set that consists of all transition formulas $\neg(t \xrightarrow{a(\xi)} t')$ where t' is a closed term, as well as the transition formula $\neg(t \xrightarrow{a(\xi)} \surd)$.

The structural operational semantics for the maximal progress operator is tricky. As a matter of fact, wrong transition rules were given in [6] for the maximal progress operator added there to a version of ACP with continuous absolute timing. A bisimulation model of the axioms of $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$ with integration, the state operator and the maximal progress operator, with or without action renaming and spatial replacement, and with or without guarded recursion, can be constructed in the same way as for $\mathrm{ACP}_{\mathrm{la}}^{\mathrm{srt}}$.

6. Data Transmission via a Mobile Intermediate Station

In this section, we consider a protocol transmitting data via a mobile intermediate station. First of all, we introduce a kind of asynchronous communication which is suitable in the case where processes are located at different places in space. Next, we introduce the protocol used for the data transmission from the sender to the intermediate station and the data transmission from the intermediate station to the receiver. After that, we introduce a buffer needed by the intermediate station. Finally, we describe the protocol transmitting data via the mobile intermediate station.

6.1. Asynchronous Communication in Space

Communication between processes at different places in space calls for a kind of asynchronous communication. Characteristic of this kind of asynchronous communication is that a datum sent at one location can only be received at another location at the point of time that it reaches that location, and that it may subsequently be received at still other locations lying at a greater distance.

This kind of asynchronous communication can be modelled using the state operator and the maximal progress operator as introduced in Section 5.

It is assumed that a fixed but arbitrary finite set C of *asynchronous channels* and a fixed but arbitrary finite set D_c of *data* for each $c \in \mathsf{C}$ have been given. It is further assumed that A contains for each $c \in \mathsf{C}$ and $d \in \mathsf{D}_c$ the following special actions:

- $c \uparrow d$, the potential sending of datum d along asynchronous channel c;
- $c \Uparrow d$, the effectuated sending of datum d along asynchronous channel c;
- $c \downarrow d$, the potential receiving of datum d along asynchronous channel c;
- $c \Downarrow d$, the effectuated receiving of datum d along asynchronous channel c.

Moreover, it is assumed that a fixed but arbitrary transmission speed v has been given. Let $c \in \mathsf{C}$. Then we write $c\Downarrow$ for the set $\{c\Downarrow d \mid d \in \mathsf{D}_c\}$.

We define a special state operator λ_V^c (V a finite subset of $\mathsf{D}_c \times \mathbb{R}^3 \times \mathbb{R}_{\geq 0}$) for asynchronous communication along channel c. The functions act_c, eff_c and eff'_c are defined as follows:

$$\begin{aligned}
\mathrm{act}_c(c\uparrow d,\xi,V) &= c\Uparrow d \\
\mathrm{act}_c(c\downarrow d,\xi,V) &= c\Downarrow d && \text{if } \exists\,(d,\xi',r)\in V \bullet |\xi-\xi'| = v\cdot r \\
&= \delta && \text{otherwise} \\
\mathrm{act}_c(a,\xi,V) &= a && \text{if } a \notin \{c\uparrow d \mid d\in \mathsf{D}_c\}\cup\{c\downarrow d \mid d\in \mathsf{D}_c\}\ , \\
\mathrm{eff}_c(c\uparrow d,\xi,V) &= V\cup\{(d,\xi,0)\} \\
\mathrm{eff}_c(a,\xi,V) &= V && \text{if } a \notin \{c\uparrow d \mid d\in \mathsf{D}_c\}\ , \\
\mathrm{eff}'_c(r,V) &= \{(d,\xi,r+r') \mid (d,\xi,r')\in V\}\ .
\end{aligned}$$

Here, we write $|\xi-\xi'|$ for the distance between ξ and ξ'. Note that the state of a channel (V) records, for each datum that has been sent along that channel, the location from which it has been sent and the time passed since it has been sent. The state does not change when a datum is received because it may later be received at a location further away from the location from which it has been sent. It is possible that two or more data arrive simultaneously at the same location. Note that, at the point of time upon which this situation arises, each datum may be received at the location concerned. There is no ordering imposed. We think that there is no sense in imposing an ordering, possibly with the exception of a few special cases.

Note further that $\widetilde{\widetilde{c\downarrow d}}(\xi)$ is transformed into $\widetilde{\widetilde{c\Downarrow d}}(\xi)$ at the one point of time, say t_0, that datum d has arrived at location ξ. Otherwise, it is transformed into $\tilde{\tilde{\delta}}$. The process $\int_{t\in[0,\infty)}\sigma^t_{\mathsf{rel}}(\widetilde{\widetilde{c\downarrow d}}(\xi))$ is capable of waiting till datum d has arrived at location ξ. However, this process may bypass the right alternative, i.e. $\sigma^{t_0}_{\mathsf{rel}}(\widetilde{\widetilde{c\downarrow d}}(\xi))$, waiting too long before trying to receive, and then no further action is possible. The solution is to enforce that $\widetilde{\widetilde{c\Downarrow d}}(\xi)$ takes place as soon as possible, by means of the maximal progress operator introduced in Section 5.2. Examples of this use of the maximal progress operator are given in other subsections of the current section.

6.2. Asynchronous Version of the PAR Protocol

We introduce an asynchronous version of the communication protocol known as the *PAR* (Positive Acknowledgement with Retransmission) protocol [22]. The sender waits for an acknowledgement before a new datum is transmitted. If an acknowledgement is not received within a complete protocol cycle, the old datum is retransmitted. In order to avoid duplicates due to retransmission, data are labeled with an alternating bit from $B = \{0,1\}$. The protocol uses the kind of asynchronous communication introduced in Section 6.1.

We have a sender process S, a receiver process R, and two asynchronous channels K and L. The process S checks with a certain frequency whether a datum d is offered at an external port (port 1). When a datum is offered, S consumes it, packs it with an alternating bit b in a frame (d,b), and then delivers the frame at one end of channel K. Next, S waits until an acknowledgement *ack* is offered at one end of

channel L. When the acknowledgement does not arrive within a certain time period, S delivers the same frame again and goes back to waiting for an acknowledgement. When the acknowledgement arrives within that time period, S goes back to checking whether a datum is offered. The process R waits until a frame with a datum and an alternating bit (d, b) is offered at the other end of channel K. When a frame is offered, R consumes it, unpacks it, and then delivers the datum d at an external port (port 2) if the alternating bit b is the right one and in any case an acknowledgement *ack* at the other end of channel L. After that, R goes back to waiting for a frame, but the right bit changes to $(1-b)$ if the alternating bit was the right one. The time t_S is the time after which the sender checks again whether a datum is offered in the case of an unsuccessful attempt. The time t'_S is the time after which the sender retransmits a datum in the case where it is still waiting for an acknowledgement. The time t_R is the time that it takes the receiver to process a frame. The time t'_R is the time that it takes the receiver to produce an acknowledgement. The locations ξ_S and ξ_R are the locations of the processes S and R. The ends of channel K are located at $\xi_S + \epsilon_S$ and $\xi_R + \epsilon_R$. The ends of channel L are located at $\xi_R - \epsilon_R$ and $\xi_S - \epsilon_S$.

We assume that the times t_S, t'_S, t_R and t'_R are non-zero. We also assume a finite set of data D. Let $F = D \times B$ be the set of frames. For $d \in D$ and $b \in B$, we also write d, b for the frame (d, b). We use the standardized notation for handshaking communication introduced in Section 2. The recursive specification of the sender consists of the following equations:

$$
\begin{aligned}
S &= S_0 \;, \\
S_b &= \widetilde{\widetilde{r_1(\mathit{error})}}(\xi_S) \cdot \sigma^{t_S}_{\mathsf{rel}}(S_b) + \sum_{d \in D} \widetilde{\widetilde{r_1(d)}}(\xi_S) \cdot \sigma^{t_S}_{\mathsf{rel}}(\mathit{SF}_{d,b}) \\
&\quad (\text{for every } b \in B), \\
\mathit{SF}_{d,b} &= \widetilde{\widetilde{K \uparrow (d,b)}}(\xi_S + \epsilon_S) \\
&\qquad \cdot \Bigl(\int_{t \in [0,t'_S)} \sigma^t_{\mathsf{rel}}\Bigl(\widetilde{\widetilde{L \downarrow \mathit{ack}}}(\xi_S - \epsilon_S) \Bigr) \cdot S_{1-b} + \sigma^{t'_S}_{\mathsf{rel}}(\mathit{SF}_{d,b}) \Bigr) \\
&\quad (\text{for every } d \in D \text{ and } b \in B).
\end{aligned}
$$

The recursive specification of the receiver consists of the following equations:

$$
\begin{aligned}
R &= R_0 \;, \\
R_b &= \int_{t \in [0,\infty)} \sum_{d \in D} \sigma^t_{\mathsf{rel}}\Bigl(\widetilde{\widetilde{K \downarrow (d,b)}}(\xi_R + \epsilon_R) \Bigr) \cdot \sigma^{t_R}_{\mathsf{rel}}\Bigl(\widetilde{\widetilde{s_2(d)}}(\xi_R) \Bigr) \\
&\qquad \cdot \sigma^{t'_R}_{\mathsf{rel}}\Bigl(\widetilde{\widetilde{L \uparrow \mathit{ack}}}(\xi_R - \epsilon_R) \Bigr) \cdot R_{1-b} \\
&\quad + \int_{t \in [0,\infty)} \sum_{d \in D} \sigma^t_{\mathsf{rel}}\Bigl(K \downarrow \widetilde{\widetilde{(d, 1-b)}}(\xi_R + \epsilon_R) \Bigr) \\
&\qquad \cdot \sigma^{t'_R}_{\mathsf{rel}}\Bigl(\widetilde{\widetilde{L \uparrow \mathit{ack}}}(\xi_R - \epsilon_R) \Bigr) \cdot R_b \\
&\quad (\text{for every } b \in B).
\end{aligned}
$$

Now consider the protocol described by the following term:

$$\theta_{K\Downarrow}(\lambda^K_\emptyset(\theta_{L\Downarrow}(\lambda^L_\emptyset(S \parallel R)))) \ .$$

This protocol is not very interesting with asynchronous communication as introduced in Section 6.1: frames and acknowledgements are transmitted through channels in which nothing will ever get lost. The protocol is intended for the case where channels are unreliable, i.e. the case where frames and acknowledgements may get lost.

Frames or acknowledgements may get lost, for example, in the case where the sender and/or the receiver move in space and there is a solid object (through which transmission is impossible) that is sometimes in between the sender and receiver. For instance, suppose that the sender is stationary and the receiver moves on a line between location ξ_R and location ξ'_R, starting at location ξ_R, such that its location at time t, say $\xi_R(t)$, is given by

$$\xi_R + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega_R t)) \cdot (\xi'_R - \xi_R) \ .$$

Moreover, suppose that the solid object is stationary. Because of the presence of this solid object, we have to impose a more restrictive condition than in Section 6.1 on $\mathrm{act}_c(c\downarrow d, \xi, V)$ to yield $c\Downarrow d$: there must exist a $(d, \xi', r) \in V$ such that $|\xi - \xi'| = v \cdot r$ and the line segment from ξ to ξ' does not intersect the solid object. Now consider the protocol described by the following term:

$$\theta_{K\Downarrow}(\lambda^K_\emptyset(\theta_{L\Downarrow}(\lambda^L_\emptyset(S \parallel \vec{\rho}_{f_R}(R))))) \ ,$$

where f_R is defined such that

$$f_R(\xi, t) = \xi + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega_R t)) \cdot (\xi'_R - \xi_R)$$

for all $\xi \in \mathbb{R}^3$ and $t \in \mathbb{R}_{\geq 0}$. A necessary condition for this protocol to be correct is that the retransmission time t'_S is longer than the longest duration of a complete protocol cycle, i.e.

$$t'_S > t_K + t_R + t'_R + t_L \ ,$$

where

$$t_K = \tfrac{\sup\{|(\xi_S+\epsilon_S)-(\xi_R(t)+\epsilon_R)| \mid t\in\mathbb{R}_{\geq 0}\}}{v} \ ,$$
$$t_L = \tfrac{\sup\{|(\xi_S-\epsilon_S)-(\xi_R(t)-\epsilon_R)| \mid t\in\mathbb{R}_{\geq 0}\}}{v} \ .$$

If the retransmission time is shorter than a complete protocol cycle, the retransmission is called premature. In that case, while an acknowledgement is still on the way, the sender will retransmit the current frame. When the acknowledgement finally arrives, the sender will treat this acknowledgement as an acknowledgement of the retransmitted frame. However, an acknowledgement of the retransmitted frame may be on the way. If the next frame transmitted gets lost and the latter acknowledgement arrives, no retransmission of that frame will follow and the protocol will fail.

6.3. Buffer

We introduce a buffer that is able to consume data at any rate and to deliver them at any rate. This buffer is needed by the mobile intermediate station used for data transmission in Section 6.4 because the transmission from the sender to the intermediate station or the transmission from the intermediate station to the receiver may be blocked by impeding solid objects. The recursive specification of such a buffer consists of the following equations:

$$\begin{aligned}
B &= B_\epsilon \;, \\
B_\epsilon &= \int_{t\in[0,\infty)} \sigma^t_{\mathsf{rel}}\Big(\widetilde{\widetilde{s_4(\mathit{error})}}(\xi_B+\epsilon_B)\Big)\cdot B_\epsilon \\
&\quad + \int_{t\in[0,\infty)} \sum_{d\in D} \sigma^t_{\mathsf{rel}}\Big(\widetilde{\widetilde{r_3(d)}}(\xi_B-\epsilon_B)\Big)\cdot B_d \;, \\
B_{\sigma d} &= \int_{t\in[0,\infty)} \sigma^t_{\mathsf{rel}}\Big(\widetilde{\widetilde{s_4(d)}}(\xi_B+\epsilon_B)\Big)\cdot B_\sigma \\
&\quad + \int_{t\in[0,\infty)} \sum_{d'\in D} \sigma^t_{\mathsf{rel}}\Big(\widetilde{\widetilde{r_3(d')}}(\xi_B-\epsilon_B)\Big)\cdot B_{d'\sigma d} \\
&(\text{for every } d\in D,\ \sigma\in D^*).
\end{aligned}$$

The process B is always ready to deliver output. If there is no datum in the buffer, an error message is delivered.

6.4. Putting the Whole Thing Together

We describe a protocol transmitting data via a mobile intermediate station. The protocol concerned uses the asynchronous version of the PAR protocol from Section 6.2 for the data transmission from the sender to the intermediate station and the data transmission from the intermediate station to the receiver. The intermediate station consists of three parts: a receiving part, a buffering part and a sending part. The buffer from Section 6.3 is used for the buffering part. The required variants of the PAR protocol and the buffer are obtained by means of action renaming and spatial replacement.

The intermediate station moves on a line between location ξ_T and location ξ'_T, starting at location ξ_T, such that its location at time t, say $\xi_T(t)$, is given by

$$\xi_T + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega_T t))\cdot(\xi'_T - \xi_T)\;.$$

The receiving part, buffering part and sending part of the intermediate station are located at $\xi_T-\epsilon_T$, ξ_T and $\xi_T+\epsilon_T$, respectively.

We use the following spatial replacement functions:

- for the starting position of the receiving part, the buffering part and the sending part of the intermediate station:

$$\begin{aligned}
f_{T_R}(\xi,t) &= \xi + (\xi_T-\epsilon_T) - (\xi_R+\epsilon_R)\;, \\
f_{T_B}(\xi,t) &= \xi + \xi_T - \xi_B\;, \\
f_{T_S}(\xi,t) &= \xi + (\xi_T+\epsilon_T) - (\xi_S+\epsilon_S)\;,
\end{aligned}$$

respectively;

- for the movement of the intermediate station:

$$f_M(\xi, t) = \xi + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega_T t)) \cdot (\xi'_T - \xi_T) \ .$$

We use the following action renaming functions:

- for the receiving part and the sending part of the intermediate station:

$$\begin{aligned}
g_{T_R}(s_2(d)) &= s_3(d) && \text{for all } d \in D \ , \\
g_{T_S}(r_1(d)) &= r_4(d) && \text{for all } d \in D \cup \{\mathsf{error}\} \\
g_{T_S}(K \uparrow (d,b)) &= K' \uparrow (d,b) && \text{for all } (d,b) \in D \times B \\
g_{T_S}(L \downarrow \mathsf{ack}) &= L' \downarrow \mathsf{ack} \ ;
\end{aligned}$$

- for the receiver:

$$\begin{aligned}
g_R(K \downarrow (d,b)) &= K' \downarrow (d,b) \text{ for all } (d,b) \in D \times B \\
g_R(L \uparrow \mathsf{ack}) &= L' \uparrow \mathsf{ack} \ .
\end{aligned}$$

The functions g_{T_R}, g_{T_S} and g_R leave all unmentioned actions unchanged.

The functions act_c, for $c \in \{K, L, K', L'\}$, have to be adapted to the presence of solid objects that are sometimes in between the sender and the intermediate station or the intermediate station and the receiver. This goes similar to the adaptation outlined at the end of Section 6.2. In Section 7.1, we will present a mathematically precise way to deal with transmission limitations resulting from solid objects that are in the way.

The receiving part, the buffering part and the sending part of the intermediate station are defined as follows:

$$\begin{aligned}
T_R &= \rho_{g_{T_R}}(\vec{\rho}_{f_M}(\vec{\rho}_{f_{T_R}}(R))) \ , \\
T_B &= \vec{\rho}_{f_M}(\vec{\rho}_{f_{T_B}}(B)) \ , \\
T_S &= \rho_{g_{T_S}}(\vec{\rho}_{f_M}(\vec{\rho}_{f_{T_S}}(S))) \ .
\end{aligned}$$

The variants of the PAR protocol for the data transmission from the sender to the intermediate station and the data transmission from the intermediate station to the receiver are defined as follows:

$$\begin{aligned}
\mathsf{PAR} &= \theta_{K\Downarrow}(\lambda_{\emptyset}^{K}(\theta_{L\Downarrow}(\lambda_{\emptyset}^{L}(S \parallel T_R)))) \ , \\
\mathsf{PAR}' &= \theta_{K'\Downarrow}(\lambda_{\emptyset}^{K'}(\theta_{L'\Downarrow}(\lambda_{\emptyset}^{L'}(T_S \parallel \rho_{g_R}(R))))) \ .
\end{aligned}$$

The whole system is described by the following term:

$$\partial_H(\mathsf{PAR} \parallel T_B \parallel \mathsf{PAR}') \ ,$$

where

$$H = \{s_3(d), r_3(d) \mid d \in D\} \cup \{s_4(d), r_4(d) \mid d \in D \cup \{\mathsf{error}\}\} \ .$$

A necessary condition for this protocol to be correct is that the retransmission time t'_S is longer than the longest duration of a complete cycle of *PAR* or *PAR*′, i.e.

$$t'_S > \max\{t_K + t_R + t'_R + t_L, t_{K'} + t_R + t'_R + t_{L'}\} \;,$$

where

$$\begin{aligned}
t_K &= \tfrac{\sup\{|(\xi_S+\epsilon_S)-((\xi_T(t)-\epsilon_T)+\epsilon_R)| \mid t\in\mathbb{R}_{\geq 0}\}}{v} \;,\\
t_L &= \tfrac{\sup\{|(\xi_S-\epsilon_S)-((\xi_T(t)-\epsilon_T)-\epsilon_R)| \mid t\in\mathbb{R}_{\geq 0}\}}{v} \;,\\
t_{K'} &= \tfrac{\sup\{|(\xi_R+\epsilon_R)-((\xi_T(t)+\epsilon_T)+\epsilon_S)| \mid t\in\mathbb{R}_{\geq 0}\}}{v} \;,\\
t_{L'} &= \tfrac{\sup\{|(\xi_R-\epsilon_R)-((\xi_T(t)+\epsilon_T)-\epsilon_S)| \mid t\in\mathbb{R}_{\geq 0}\}}{v} \;.
\end{aligned}$$

Even if the retransmission time is long enough, the protocol may have shortcomings. One shortcoming would be that the buffer size increases permanently because solid objects between the intermediate station and the receiver are in the way more often than solid objects between the sender and the intermediate station. Another shortcoming would be that the first datum ever transmitted by the sender never reaches the receiver because there are solid objects in the way on all transmission attempts of the sender or all transmission attempts of the intermediate station. Whether these shortcomings occur, depends not only on timing details of the protocol and motion details of the intermediate station, but also on motion details of the solid objects concerned.

7. Elaboration on the Foregoing

In this section, we elaborate on what is presented in Section 6. First, we introduce a way to deal uniformly with all transmission limitations resulting from solid objects that are in the way. Secondly, we introduce a variant of the kind of asynchronous communication introduced earlier which is based on frequencies instead of channels.

7.1. Transmission with Solid Objects in the Way

Until now, we have dealt in an ad hoc way with transmission limitations resulting from solid objects that are in the way, using sentences like "the line segment from ξ and ξ' does not intersect the solid object". In order to deal uniformly with all transmission limitations resulting from solid objects that are in the way, we add a parameter to the special state operators for asynchronous communication and adapt the functions act_c uniformly for all $c \in \mathsf{C}$. The parameter is an *occupancy function* $\mu : \mathbb{R}_{\geq 0} \to \mathcal{P}(\mathbb{R}^3)$. The function μ is regarded to give for any point of time the points in space that are occupied by solid objects. This provides for dealing with both stationary and moving solid objects.

The function act_c ($c \in \mathsf{C}$) is adapted as follows:

$$\begin{aligned}
&\mathrm{act}_c(c\uparrow d, \xi, V) = c \Uparrow d\\
&\mathrm{act}_c(c\downarrow d, \xi, V) = c \Downarrow d \text{ if } \exists\, (d, \xi', r) \in V \bullet |\xi - \xi'| = v \cdot r \wedge\\
&\qquad\qquad\qquad\qquad \forall\, r' \leq r \bullet \xi' + v \cdot r' \cdot \tfrac{\xi-\xi'}{|\xi-\xi'|} \notin \mu(r')\\
&\qquad\qquad = \delta \quad \text{otherwise}\\
&\mathrm{act}_c(a, \xi, V) \quad = a \quad \text{if } a \notin \{c\uparrow d \mid d \in \mathsf{D}_c\} \cup \{c\downarrow d \mid d \in \mathsf{D}_c\} \;.
\end{aligned}$$

With this adaptation, a datum sent at one location cannot be received at another location if at some point of time during the transmission from the former location to the latter location a point in space was reached that was occupied at that point of time by a solid object according to the function μ. Notice that we get back the kind of asynchronous communication in space introduced in Section 6.1 by taking the unique μ such that $\mu(t) = \emptyset$ for all $t \in \mathbb{R}_{\geq 0}$.

Consider again the protocol described in Section 6.4. Suppose that there are two moving solid balls O and O' with radius ρ_O and $\rho_{O'}$, respectively. Ball O moves on a line between location ξ_O and location ξ'_O, and ball O' moves on a line between location $\xi_{O'}$ and location $\xi'_{O'}$. The location of ball O at time t, say $\xi_O(t)$, and the location of ball O' at time t, say $\xi_{O'}(t)$, are given by

$$\xi_O + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega_O t)) \cdot (\xi'_O - \xi_O) \;,$$

$$\xi_{O'} + (\tfrac{1}{2} - \tfrac{1}{2}\cos(\omega_{O'} t)) \cdot (\xi'_{O'} - \xi_{O'}) \;.$$

To make the example somewhat realistic, we impose the following conditions on ξ_O ξ'_O, ω_O, ρ_O, $\xi_{O'}$, $\xi'_{O'}$, $\omega_{O'}$, $\rho_{O'}$:

- there exist a $t \in \mathbb{R}_{\geq 0}$ and a $\xi \in \mathbb{R}^3$ such that $|\xi - \xi_O(t)| \leq \rho_O$ and either ξ is on the line segment $(\xi_S + \epsilon_S, (\xi_T(t) - \epsilon_T) + \epsilon_R)$ or ξ is on the line segment $(\xi_S - \epsilon_S, (\xi_T(t) - \epsilon_T) - \epsilon_R)$;
- there exists a $t \in \mathbb{R}_{\geq 0}$ such that there does not exist a $\xi \in \mathbb{R}^3$ such that $|\xi - \xi_O(t)| \leq \rho_O$ and ξ is on the line segment $(\xi_S + \epsilon_S, (\xi_T(t) - \epsilon_T) + \epsilon_R)$; and there exists a $t \in \mathbb{R}_{\geq 0}$ such that there does not exist a $\xi \in \mathbb{R}^3$ such that $|\xi - \xi_O(t)| \leq \rho_O$ and ξ is on the line segment $(\xi_S - \epsilon_S, (\xi_T(t) - \epsilon_T) - \epsilon_R)$;
- there exist a $t \in \mathbb{R}_{\geq 0}$ and a $\xi \in \mathbb{R}^3$ such that $|\xi - \xi_{O'}(t)| \leq \rho_{O'}$ and either ξ is on the line segment $((\xi_T(t) + \epsilon_T) + \epsilon_S, \xi_R + \epsilon_R)$ or ξ is on the line segment $((\xi_T(t) + \epsilon_T) - \epsilon_S, \xi_R - \epsilon_R)$;
- there exists a $t \in \mathbb{R}_{\geq 0}$ such that there does not exist a $\xi \in \mathbb{R}^3$ such that $|\xi - \xi_{O'}(t)| \leq \rho_{O'}$ and ξ is on the line segment $((\xi_T(t) + \epsilon_T) + \epsilon_S, \xi_R + \epsilon_R)$; and there exists a $t \in \mathbb{R}_{\geq 0}$ such that there does not exist a $\xi \in \mathbb{R}^3$ such that $|\xi - \xi_{O'}(t)| \leq \rho_{O'}$ and ξ is on the line segment $((\xi_T(t) + \epsilon_T) - \epsilon_S, \xi_R - \epsilon_R)$;
- $(\xi_O, \xi'_O) \perp (\xi_T, \xi'_T)$;
- $(\xi_{O'}, \xi'_{O'}) \perp (\xi_T, \xi'_T)$.

If these conditions are met, the solid object O blocks the transmission along channel K or channel L regularly, but neither of them permanently, the solid object O' blocks the transmission along channel K' or channel L' regularly, but neither of them permanently, and the lines on which the solid objects O and O' move are perpendicular to the line on which the intermediate station T moves.

The following occupancy function gives for any point of time the points in space that are occupied by O or O':

$$\mu(t) = \{\xi \mid |\xi - \xi_O(t)| \leq \rho_O\} \cup \{\xi \mid |\xi - \xi_{O'}(t)| \leq \rho_{O'}\} \;.$$

With this occupancy function, and the adapted functions act_c given above, we have made precise what the sentence "The functions act_c, for $c \in \{K, L, K', L'\}$, have to be adapted to the presence of solid objects that are sometimes in between the sender and the intermediate station or the intermediate station and the receiver" from Section 6.4 referred to.

Using this occupancy function as actual parameter of the parametrized state operators for asynchronous communication introduces above, the protocol from Section 6.4 can be described by the following term:

$$\partial_H(\mathit{PAR} \parallel T_B \parallel \mathit{PAR}') \ ,$$

where

$$\begin{aligned}\mathit{PAR} &= \theta_{K\Downarrow}(\lambda[\mu]^K_\emptyset(\theta_{L\Downarrow}(\lambda[\mu]^L_\emptyset(S \parallel T_R)))) \ ,\\ \mathit{PAR}' &= \theta_{K'\Downarrow}(\lambda[\mu]^{K'}_\emptyset(\theta_{L'\Downarrow}(\lambda[\mu]^{L'}_\emptyset(T_S \parallel \rho_{g_R}(R))))) \ ,\end{aligned}$$

and where the processes S, R, T_R, T_B, T_S, the set of actions H, and the renaming function g_R are defined as in Section 6.

The conditions imposed above on the solid objects O and O' exclude the really uninteresting cases, viz. the case where these objects never block the transmission and the case where they always block the transmission. However, the conditions do not guarantee that the shortcomings mentioned at the end of Section 6.4 do not occur. The description of the protocol given here provides all that may be needed to check whether a given condition is sufficient for that.

7.2. From Channels to Frequencies

Until now, data was communicated along different channels to prevent that data meant to be communicated between certain processes to a certain end would be mixed up with data meant to be communicated between other processes or to another end. In concrete cases, this effect is reached by communicating data at different frequencies. The abstraction made by using channels may hide details that are relevant to the behaviour of the system concerned. In order to support asynchronous communication at frequencies as well, we introduce a variant of the kind of asynchronous communication introduced earlier.

In the case of asynchronous communication along channels, we have chosen for a separate state operator for each channel. This choice emphasizes that data communicated along different channels are not mixed up. We could have chosen for a common state operator for all channels as well. In the case of asynchronous communication at frequencies, we have chosen for a common state operator for all frequencies. This choice emphasizes that in concrete cases all data transmission concerned takes place through the same medium, no matter what frequency is used.

It is assumed that a fixed but arbitrary finite set D of *data* has been given. It is further assumed that A contains for each frequency $\nu \in \mathbb{R}_{>0}$ and $d \in \mathsf{D}$ the following special actions:

- $\nu \uparrow d$, the potential sending of datum d at frequency ν;
- $\nu \Uparrow d$, the effectuated sending of datum d at frequency ν;
- $\nu \downarrow d$, the potential receiving of datum d at frequency ν;
- $\nu \Downarrow d$, the effectuated receiving of datum d at frequency ν.

We define a state operator $\lambda[\mu]_V$ (V a finite subset of $\mathsf{D} \times \mathbb{R}_{>0} \times \mathbb{R}^3 \times \mathbb{R}_{\geq 0}$) for asynchronous

communication at any frequency. The functions act, eff and eff' are defined as follows:

$$\begin{aligned}
\mathrm{act}(\nu \uparrow d, \xi, V) &= \nu \Uparrow d \\
\mathrm{act}(\nu \downarrow d, \xi, V) &= \nu \Downarrow d && \text{if } \exists\, (d, \nu, \xi', r) \in V \bullet |\xi - \xi'| = v \cdot r \wedge \\
&&& \qquad \forall\, r' \leq r \bullet \xi' + v \cdot r' \cdot \tfrac{\xi - \xi'}{|\xi - \xi'|} \notin \mu(r') \\
&= \delta && \text{otherwise} \\
\mathrm{act}(a, \xi, V) &= a && \text{if } a \notin \{\nu \uparrow d \mid d \in \mathsf{D}\} \cup \{\nu \downarrow d \mid d \in \mathsf{D}\}\,, \\
\mathrm{eff}(\nu \uparrow d, \xi, V) &= V \cup \{(d, \nu, \xi, 0)\} \\
\mathrm{eff}(a, \xi, V) &= V && \text{if } a \notin \{\nu \uparrow d \mid d \in \mathsf{D}\}\,, \\
\mathrm{eff}'(r, V) &= \{(d, \nu, \xi, r + r') \mid (d, \nu, \xi, r') \in V\}\,.
\end{aligned}$$

Note that the state now records, for each datum that has been sent, in addition to the location from which it has been sent and the time passed since it has been sent, the frequency at which it has been sent.

Consider again the asynchronous version of the PAR protocol from Section 6.2. Simply changing the channels into frequencies is not very interesting. More interesting is to add details to the protocol that cannot be described by means of channels, for example, fluctuations that may occur in the frequencies used. Therefore, suppose that the frames that were previously communicated along channel K are now communicated at a frequency between ν_K and ν'_K, and the acknowledgements that were previously communicated along channel L are now communicated at a frequency between ν_L and ν'_L.

Obviously, fluctuations may be caused by inaccuracies of transmission hardware at the sending side. However, the Doppler effect is usually the most important source of fluctuations if not both the sender and the receiver are stationary. The Doppler effect makes it appear to the receiver that the sender has used a frequency different from the frequency that actually has been used. As a result, the Doppler effect can be modelled by arranging for fluctuations in the frequency used by the sender. The bounds of the Doppler effect can be calculated by means of physical laws.

We replace the sender S and the receiver R by the more concrete processes S^* and R^*. The recursive specification of S^* consists of the following equations:

$$\begin{aligned}
S^* &= S^*_0\,, \\
S^*_b &= r_1\widetilde{\widetilde{(\mathit{error})}}(\xi_S) \cdot \sigma^{t_S}_{\mathsf{rel}}(S^*_b) + \sum_{d \in D} \widetilde{\widetilde{r_1(d)}}(\xi_S) \cdot \sigma^{t_S}_{\mathsf{rel}}(\mathit{SF}^*_{d,b}) \\
&\text{(for every } b \in B\text{)}, \\
\mathit{SF}^*_{d,b} &= \int_{\nu \in (\nu_K, \nu'_K)} \nu\widetilde{\widetilde{\uparrow(d,b)}}(\xi_S + \epsilon_S) \\
&\qquad \cdot \Bigg(\int_{t \in [0, t'_S)} \int_{\nu \in (\nu_L, \nu'_L)} \sigma^t_{\mathsf{rel}}\Big(\widetilde{\widetilde{\nu \downarrow \mathit{ack}}}(\xi_S - \epsilon_S) \Big) \cdot S^*_{1-b} + \sigma^{t'_S}_{\mathsf{rel}}(\mathit{SF}^*_{d,b}) \Bigg) \\
&\text{(for every } d \in D \text{ and } b \in B\text{)}.
\end{aligned}$$

The recursive specification of R^* consists of the following equations:

$$\begin{aligned}
R^* &= R^*_0 \,,\\
R^*_b &= \int_{t\in[0,\infty)}\int_{\nu\in(\nu_K,\nu'_K)}\sum_{d\in D}\sigma^t_{\mathsf{rel}}\Big(\nu\,\widetilde{\widetilde{\downarrow(d,b)}}(\xi_R+\epsilon_R)\Big)\cdot\sigma^{t_R}_{\mathsf{rel}}\Big(\widetilde{\widetilde{s_2(d)}}(\xi_R)\Big)\\
&\qquad\cdot\int_{\nu\in(\nu_L,\nu'_L)}\sigma^{t'_R}_{\mathsf{rel}}\Big(\widetilde{\widetilde{\nu\uparrow \mathsf{ack}}}(\xi_R-\epsilon_R)\Big)\cdot R^*_{1-b}\\
&\quad+\int_{t\in[0,\infty)}\int_{\nu\in(\nu_K,\nu'_K)}\sum_{d\in D}\sigma^t_{\mathsf{rel}}\Big(\nu\downarrow\widetilde{\widetilde{(d,1-b)}}(\xi_R+\epsilon_R)\Big)\\
&\qquad\cdot\int_{\nu\in(\nu_L,\nu'_L)}\sigma^{t'_R}_{\mathsf{rel}}\Big(\widetilde{\widetilde{\nu\uparrow \mathsf{ack}}}(\xi_R-\epsilon_R)\Big)\cdot R^*_b
\end{aligned}$$

(for every $b\in B$).

The more concrete version of the protocol from Section 6.2 is described by the following term:

$$\theta_H(\lambda[\mu]_\emptyset(S^*\parallel\vec{\rho}_{f_R}(R^*)))\,,$$

where

$$H=\{\nu\Downarrow(d,b)\mid\nu\in(\nu_K,\nu'_K),(d,b)\in D\times B\}\cup\{\nu\Downarrow\mathsf{ack}\mid\nu\in(\nu_L,\nu'_L)\}\,,$$

where the replacement function f_R for the movement of the receiver is defined as in Section 6.2, and where the occupancy function μ is defined similar to the one defined in Section 7.1. To prevent that data being communicated are unintensionally mixed up, we generally use frequency ranges that are disjunct. In this particular case, this is not needed because frames will never be mixed up with acknowledgements.

8. Concluding Remarks

A process algebra has been presented which makes it possible to deal with the behaviour of systems with a known, possibly time-dependent, spatial distribution. This process algebra is a process algebra with timing and located actions. It is intended as an algebraic framework for the description and analysis of spatially distributed systems. The application of the framework has been illustrated by means of an example concerning data transmission via a mobile intermediate station.

The presented process algebra is a reformulation of the real space process algebra from [3] in a setting with urgent actions. Other process algebras featuring urgent actions include the ACP-style process algebras with timing presented in [6], ATP [20], the different versions of CCS with timing [12, 19, 24], Timed CSP [13], TIC [21], and TPL [14].

The presented process algebra is obtained by adapting $\mathrm{ACP}^{\mathrm{srt}}$, one of the process algebras with timing from [6], to spatially located actions. As a result of the simpler way in which is dealt with the situation that two or more actions are to be performed independently at the same point of time, the state operator and the maximal progress operator turn out to be far less complicated than in [3]. We have found that those operators are useful to model the kind of asynchronous communication to be used in the case where processes are located at different places in space.

We further call to mind the power of spatial replacement. The example worked out in this paper demonstrates that by means of spatial replacement a communication protocol between a stationary sender and a stationary receiver can easily be turned into one of which the sender or the receiver moves in space.

The process algebra presented in this paper does not incorporate abstraction from internal actions. This issue is not even fully understood in process algebras with timing that do not support located actions. The version of branching bisimulation equivalence for processes with discrete relative timing proposed in [4] for this purpose, and adapted to continuous relative timing in [6], is too fine for many applications. A slightly coarser equivalence is proposed in [7], but unfortunately the definition given in that paper is faulty.

To the best of our knowledge, there is no related work, other than the work on real space process algebra presented in [3]. One of the options for further work that we consider is to investigate how the possibility to deal with the behaviour of systems with a known time-dependent spatial distribution can be incorporated in the process algebra for hybrid systems from [11].

References

[1] Aceto, L., Fokkink, W. J., Verhoef, C.: Structural Operational Semantics, in: *Handbook of Process Algebra* (J. A. Bergstra, A. Ponse, S. A. Smolka, Eds.), Elsevier, Amsterdam, 2001, 197–292.

[2] Baeten, J. C. M., Bergstra, J. A.: Real Time Process Algebra, *Formal Aspects of Computing*, **3**(2), 1991, 142–188.

[3] Baeten, J. C. M., Bergstra, J. A.: Real Space Process Algebra, *Formal Aspects of Computing*, **5**(6), 1993, 481–529.

[4] Baeten, J. C. M., Bergstra, J. A., Reniers, M. A.: Discrete Time Process Algebra with Silent Step, in: *Proof, Language and Interaction: Essays in Honour of Robin Milner* (G. D. Plotkin, C. Stirling, M. Tofte, Eds.), MIT Press, Cambridge, MA, 2000, 535–569.

[5] Baeten, J. C. M., Middelburg, C. A.: Process Algebra with Timing: Real Time and Discrete Time, in: *Handbook of Process Algebra* (J. A. Bergstra, A. Ponse, S. A. Smolka, Eds.), Elsevier, Amsterdam, 2001, 627–684.

[6] Baeten, J. C. M., Middelburg, C. A.: *Process Algebra with Timing*, Monographs in Theoretical Computer Science, An EATCS Series, Springer-Verlag, Berlin, 2002.

[7] Baeten, J. C. M., Middelburg, C. A., Reniers, M. A.: *A New Equivalence for Processes with Timing – With an Application to Protocol Verification*, Computer Science Report 02-10, Department of Mathematics and Computer Science, Eindhoven University of Technology, October 2002.

[8] Baeten, J. C. M., Weijland, W. P.: *Process Algebra*, vol. 18 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, Cambridge, 1990.

[9] Bergstra, J. A., Klop, J. W.: Process Algebra for Synchronous Communication, *Information and Control*, **60**(1/3), 1984, 109–137.

[10] Bergstra, J. A., Klop, J. W.: Verification of an Alternating Bit Protocol by Means of Process Algebra, *Mathematical Methods of Specification and Synthesis of Software Systems* (W. Bibel, K. P. Jantke, Eds.), 215, Springer-Verlag, 1986.

[11] Bergstra, J. A., Middelburg, C. A.: *Process Algebra for Hybrid Systems*, Computer Science Report 03-06, Department of Mathematics and Computer Science, Eindhoven University of Technology, June 2003.

[12] Chen, L.: An Interleaving Model for Real-Time Systems, *Symposium on Logical Foundations of Computer Science* (A. Nerode, M. Taitslin, Eds.), 620, Springer-Verlag, 1992.

[13] Davies, J., et al.: Timed CSP: Theory and Practice, *Real Time: Theory and Practice* (J. W. de Bakker, C. Huizing, W. P. de Roever, G. Rozenberg, Eds.), 600, Springer-Verlag, 1992.

[14] Hennessy, M., Regan, T.: A Process Algebra for Timed Systems, *Information and Computation*, **117**, 1995, 221–239.

[15] Middelburg, C. A.: Variable Binding Operators in Transition System Specifications, *Journal of Logic and Algebraic Programming*, **47**(1), 2001, 15–45.

[16] Middelburg, C. A.: Process Algebra with Nonstandard Timing, *Fundamenta Informaticae*, **53**(1), 2002, 55–77.

[17] Middelburg, C. A.: An Alternative Formulation of Operational Conservativity with Binding Terms, *Journal of Logic and Algebraic Programming*, **55**(1/2), 2003, 1–19.

[18] Middelburg, C. A.: Revisiting Timing in Process Algebra, *Journal of Logic and Algebraic Programming*, **54**(1/2), 2003, 109–127.

[19] Moller, F., Tofts, C.: A Temporal Calculus of Communicating Systems, *CONCUR'90* (J. C. M. Baeten, J. W. Klop, Eds.), 458, Springer-Verlag, 1990.

[20] Nicollin, X., Sifakis, J.: The Algebra of Timed Processes ATP: Theory and Application, *Information and Computation*, **114**(1), 1994, 131–178.

[21] Quemada, J., de Frutos, D., Azcorra, A.: TIC: A Timed Calculus, *Formal Aspects of Computing*, **5**(3), 1993, 224–252.

[22] Tanenbaum, A. S.: *Computer Networks*, Prentice-Hall, Englewood Cliffs, 1981.

[23] Verhoef, C.: A Congruence Theorem for Structured Operational Semantics with Predicates and Negative Premises, *Nordic Journal of Computing*, **2**(2), 1995, 274–302.

[24] Wang Yi: Real-Time Behaviour of Asynchronous Agents, *CONCUR'90* (J. C. M. Baeten, J. W. Klop, Eds.), 458, Springer-Verlag, 1990.