A thread calculus with molecular dynamics

A Thread Calculus with Molecular Dynamics*

J.A. Bergstra[1,2] and C.A. Middelburg[3,1]

[1] Programming Research Group, University of Amsterdam,
P.O. Box 41882, 1009 DB Amsterdam, the Netherlands
[2] Department of Philosophy, Utrecht University,
P.O. Box 80126, 3508 TC Utrecht, the Netherlands
[3] Computing Science Department, Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
janb@science.uva.nl,keesm@win.tue.nl

Abstract. In a previous paper, we developed an algebraic theory of threads, interleaving of threads, and interaction between threads and services. In the current paper, we extend that theory with features of molecular dynamics, a model of computation suitable for object-based programs. In this model, threads interact with a service of which the states resemble collections of molecules composed of atoms and computations take place by means of actions which transform the structure of molecules like in chemical reactions. The features introduced include a feature to restrict the scope of names used in threads to refer to molecules. Because that feature makes it troublesome to provide a structural operational semantics, we construct a projective limit model for the extended theory.

Keywords: thread calculus, thread algebra, molecular dynamics, restriction, projective limit model.

1998 CR Categories: D.1.3, D.1.5, D.3.3, F.1.1, F.1.2, F.3.2.

1 Introduction

A thread is the behaviour of a deterministic sequential program under execution. Multi-threading refers to the concurrent existence of several threads in a program under execution. Multi-threading is the dominant form of concurrency provided by recent object-oriented programming languages such as Java [19] and C# [20]. Arbitrary interleaving, on which theories about concurrent processes such as ACP [5] are based, is not an appropriate abstraction when dealing with multi-threading. In the case of multi-threading, some deterministic interleaving strategy is used. In [8], we introduced a number of plausible deterministic interleaving strategies for multi-threading. We proposed to use the phrase strategic interleaving for the more constrained form of interleaving obtained by using such

* This research was partly carried out in the framework of the GLANCE-project MICROGRIDS, which is funded by the Netherlands Organisation for Scientific Research (NWO).

a strategy. We also introduced a feature for interaction of threads with services. The algebraic theory of threads, multi-threading, and interaction of threads with services is called thread algebra.

In the current paper, we extend thread algebra with features of molecular dynamics, a model of computation suitable for object-based programs. In this model, threads interact with a service of which the states resemble collections of molecules composed of atoms and computations take place by means of actions which transform the structure of molecules like in chemical reactions. The model introduced in the current paper elaborates on the model described informally in [2]. The additional features include a feature to restrict the scope of names used in threads to refer to molecules. That feature, which has no counterpart in [2], turns thread algebra into a calculus. Although it occurs in quite another setting, it is reminiscent of restriction in the π-calculus [25].

In thread algebra, we abandon the point of view that arbitrary interleaving is an appropriate abstraction when dealing with multi-threading. The following points illustrate why that point of view is untenable: (a) whether the interleaving of certain threads leads to deadlock depends on the interleaving strategy used; (b) sometimes deadlock takes place with a particular interleaving strategy whereas arbitrary interleaving would not lead to deadlock, and vice versa. We give demonstrations of (a) and (b) in [8] and [12], respectively. The thread-service dichotomy that we make in thread algebra is useful for the following reasons: (a) for services, a state-based description is generally more convenient than an action-based description whereas it is the other way round for threads; (b) the interaction between threads and services is of an asymmetric nature. In [12], evidence of both (a) and (b) is produced by the established connections of threads and services with processes as considered in an extension of ACP with conditions introduced in [11].

Although thread algebra is concerned with the constrained form of interleaving found in multi-threading as provided by contemporary programming languages, not all relevant details of multi-threading as provided by those languages can be modelled with thread algebra. The details concerned come up where multi-threading is intertwined with object-orientation. The form of thread forking where a unique identity object is associated with the thread being forked off is an example of this. Setting up a framework in which such details can be modelled as well is the main objective with which we have extended thread algebra with features of molecular dynamics. The form of thread forking mentioned above is modelled in this paper using the thread calculus developed. The feature to restrict the scope of names used in threads to refer to molecules turns out to be indispensable when modelling this form of thread forking.

Associating transition systems with closed terms over the signature of the full thread calculus developed in this paper by means of structural operational semantics is troublesome. The feature to restrict the scope of names used in threads to refer to molecules complicate matters to such an extent that a structural operational semantics would add at most marginally to a better understanding. Therefore, we provide instead a projective limit model. In process algebra, projective limit models have been given for the first time by Bergstra and Klop [5]. Following [23], we make the domain of the projective limit model into a metric space to show, using Banach's fixed point theorem, that recursion equations satisfying a guardedness condition have unique solutions. Metric spaces have also been applied in concurrency theory by de Bakker and others to solve domain equations for process domains [17] and to establish uniqueness results for recursion equations [16].

Thread forking is inherent in multi-threading. However, we will not introduce thread interleaving and thread forking combined. Thread forking is presented at a later stage as an extension. This is for expository reasons only. The formulations of many results, as well as their proofs, would be complicated by introducing thread forking at an early stage because the presence of thread forking would be accountable to many exceptions in the results. In the set-up in which thread forking is introduced later on, we can simply summarize which results need to be adapted to the presence of thread forking and how.

Thread algebra is a design on top of an algebraic theory of the behaviour of deterministic sequential programs under execution introduced in [6] under the name basic polarized process algebra. Prompted by the development of thread algebra, basic polarized process algebra has been renamed to basic thread algebra.

This paper is organized as follows. First, we review basic thread algebra (Section 2). After that, we extend basic thread algebra to a theory of threads, interleaving of threads and interaction of threads with services (Sections 3 and 4) and introduce recursion in this setting (Section 5). Next, we propose a state-based approach to describe services (Section 6) and use it to describe services for molecular dynamics (Section 7). We also discuss some syntactic issues concerning molecular dynamics (Section 8). Then, we introduce a feature to restrict the scope of names used in threads to refer to molecules (Section 9). Following this, we introduce the approximation induction principle to reason about infinite threads (Section 10). After that, we introduce a basic form of thread forking and demonstrate how the restriction feature can be used to model a form of thread forking found in object-oriented programming languages (Section 11). Next, we construct the projective limit model for the thread calculus developed in this paper (Sections 12–14). Then, we show that recursion equations satisfying a guardedness condition have unique solutions in this model (Section 15). Following this, we outline the adaptation of the projective limit model to the basic form of thread forking introduced earlier (Section 16). Finally, we make some concluding remarks (Section 17).

The proofs of the theorems and propositions for which no proof is given in this paper can be found in [7]. In Sections 13–15, some familiarity with metric spaces is assumed. The definitions of all notions concerning metric spaces that are assumed known in those sections can be found in most introductory textbooks on topology. We mention [15] as an example of an introductory textbook in which those notions are introduced in an intuitively appealing way.

Table 1. Axiom of BTA

$x \unlhd \mathsf{tau} \unrhd y = x \unlhd \mathsf{tau} \unrhd x$	T1

2 Basic Thread Algebra

In this section, we review BTA (Basic Thread Algebra), introduced in [6] under the name BPPA (Basic Polarized Process Algebra). BTA is a form of process algebra which is tailored to the description of the behaviour of deterministic sequential programs under execution.

In BTA, it is assumed that there is a fixed but arbitrary set of *basic actions* $\mathcal{A}$ with $\mathsf{tau} \notin \mathcal{A}$. We write $\mathcal{A}_{\mathsf{tau}}$ for $\mathcal{A} \cup \{\mathsf{tau}\}$. The signature of BTA consists of the following constants and operators:

- the *deadlock* constant D;
- the *termination* constant S;
- for each $a \in \mathcal{A}_{\mathsf{tau}}$, a binary *postconditional composition* operator $\_ \unlhd a \unrhd \_$.

We use infix notation for postconditional composition. We introduce *action prefixing* as an abbreviation: $a \circ p$, where p is a term over the signature of BTA, abbreviates $p \unlhd a \unrhd p$.

The intuition is that each basic action performed by a thread is taken as a command to be processed by the execution environment of the thread. The processing of a command may involve a change of state of the execution environment. At completion of the processing of the command, the execution environment produces a reply value. This reply is either T or F and is returned to the thread concerned. Let p and q be closed terms over the signature of BTA and $a \in \mathcal{A}_{\mathsf{tau}}$. Then $p \unlhd a \unrhd q$ will perform action a, and after that proceed as p if the processing of a leads to the reply T (called a positive reply), and it will proceed as q if the processing of a leads to the reply F (called a negative reply). The action tau plays a special role. Its execution will never change any state and always lead to a positive reply.

BTA has only one axiom. This axiom is given in Table 1. Using the abbreviation introduced above, axiom T1 can be written as follows: $x \unlhd \mathsf{tau} \unrhd y = \mathsf{tau} \circ x$.

Henceforth, we will write BTA(A) for BTA with the set of basic actions $\mathcal{A}$ fixed to be the set A.

As mentioned above, the behaviour of a thread depends upon its execution environment. Each basic action performed by the thread is taken as a command to be processed by the execution environment. At any stage, the commands that the execution environment can accept depend only on its history, i.e. the sequence of commands processed before and the sequence of replies produced for those commands. When the execution environment accepts a command, it will produce a positive reply or a negative reply. Whether the reply is positive or negative usually depends on the execution history. However, it may also depend on external conditions.

In the structural operational semantics of BTA, we represent an execution environment by a function $\rho : (\mathcal{A} \times \{\mathsf{T}, \mathsf{F}\})^* \to \mathcal{P}(\mathcal{A} \times \{\mathsf{T}, \mathsf{F}\})$ that satisfies the following condition: $(a, b) \notin \rho(\alpha) \Rightarrow \rho(\alpha \frown \langle (a, b) \rangle) = \emptyset$ for all $a \in \mathcal{A}$, $b \in \{\mathsf{T}, \mathsf{F}\}$ and $\alpha \in (\mathcal{A} \times \{\mathsf{T}, \mathsf{F}\})^*$.[4] We write $\mathcal{E}$ for the set of all those functions. Given an execution environment $\rho \in \mathcal{E}$ and a basic action $a \in \mathcal{A}$, the *derived* execution environment of ρ after processing a with a *positive* reply, written $\frac{\partial^+}{\partial a} \rho$, is defined by $\frac{\partial^+}{\partial a} \rho(\alpha) = \rho(\langle (a, \mathsf{T}) \rangle \frown \alpha)$; and the *derived* execution environment of ρ after processing a with a *negative* reply, written $\frac{\partial^-}{\partial a} \rho$, is defined by $\frac{\partial^-}{\partial a} \rho(\alpha) = \rho(\langle (a, \mathsf{F}) \rangle \frown \alpha)$.

The following transition relations on closed terms over the signature of BTA are used in the structural operational semantics of BTA:

- a binary relation $\langle \_, \rho \rangle \xrightarrow{a} \langle \_, \rho' \rangle$ for each $a \in \mathcal{A}_{\mathsf{tau}}$ and $\rho, \rho' \in \mathcal{E}$;
- a unary relation $\_ \downarrow$;
- a unary relation $\_ \uparrow$;
- a unary relation $\_ \updownarrow$.

The four kinds of transition relations are called the *action step*, *termination*, *deadlock*, and *termination or deadlock* relations, respectively. They can be explained as follows:

- $\langle p, \rho \rangle \xrightarrow{a} \langle p', \rho' \rangle$: in execution environment ρ, thread p can perform action a and after that proceed as thread p' in execution environment ρ';
- $p \downarrow$: thread p cannot but terminate successfully;
- $p \uparrow$: thread p cannot but become inactive;
- $p \updownarrow$: thread p cannot but terminate successfully or become inactive.

The termination or deadlock relation is an auxiliary relation needed when we extend BTA in Section 3.

The structural operational semantics of BTA is described by the transition rules given in Table 2. In this table a stands for an arbitrary action from $\mathcal{A}$.

Bisimulation equivalence is defined as follows. A *bisimulation* is a symmetric binary relation B on closed terms over the signature of BTA such that for all closed terms p and q:

- if $B(p, q)$ and $\langle p, \rho \rangle \xrightarrow{a} \langle p', \rho' \rangle$, then there is a q' such that $\langle q, \rho \rangle \xrightarrow{a} \langle q', \rho' \rangle$ and $B(p', q')$;
- if $B(p, q)$ and $p \downarrow$, then $q \downarrow$;
- if $B(p, q)$ and $p \uparrow$, then $q \uparrow$.

Two closed terms p and q are *bisimulation equivalent*, written $p \leftrightarrows q$, if there exists a bisimulation B such that $B(p, q)$.

[4] We write $\langle\,\rangle$ for the empty sequence, $\langle d \rangle$ for the sequence having d as sole element, and $\alpha \frown \beta$ for the concatenation of finite sequences α and β. We assume the usual laws for concatenation of finite sequences. We write D^* for the set of all finite sequences with elements from set D, and D^+ for the set of all non-empty finite sequences with elements from set D.

Table 2. Transition rules of BTA

$$\frac{}{\mathsf{S}\downarrow} \qquad \frac{}{\mathsf{D}\uparrow} \qquad \frac{}{\langle x \unlhd \mathsf{tau} \unrhd y, \rho\rangle \xrightarrow{\mathsf{tau}} \langle x, \rho\rangle}$$

$$\frac{}{\langle x \unlhd a \unrhd y, \rho\rangle \xrightarrow{a} \langle x, \frac{\partial^{+}}{\partial a}\,\rho\rangle}\ (a, \mathsf{T}) \in \rho(\langle\,\rangle) \qquad \frac{}{\langle x \unlhd a \unrhd y, \rho\rangle \xrightarrow{a} \langle y, \frac{\partial^{-}}{\partial a}\,\rho\rangle}\ (a, \mathsf{F}) \in \rho(\langle\,\rangle)$$

$$\frac{x\downarrow}{x\updownarrow} \qquad \frac{x\uparrow}{x\updownarrow}$$

Bisimulation equivalence is a congruence with respect to the postconditional composition operators. This follows immediately from the fact that the transition rules for these operators are in the path format (see e.g. [1]). The axiom given in Table 1 is sound with respect to bisimulation equivalence.

3 Strategic Interleaving of Threads

In this section, we take up the extension of BTA to a theory about threads and multi-threading by introducing a very simple interleaving strategy.

It is assumed that the collection of threads to be interleaved takes the form of a sequence of threads, called a *thread vector*. Strategic interleaving operators turn a thread vector of arbitrary length into a single thread. This single thread obtained via a strategic interleaving operator is also called a multi-thread. Formally, however multi-threads are threads as well. In this paper, we only cover the simplest interleaving strategy, namely *cyclic interleaving*. Cyclic interleaving basically operates as follows: at each stage of the interleaving, the first thread in the thread vector gets a turn to perform a basic action and then the thread vector undergoes cyclic permutation. We mean by cyclic permutation of a thread vector that the first thread in the thread vector becomes the last one and all others move one position to the left. If one thread in the thread vector deadlocks, the whole does not deadlock till all others have terminated or deadlocked. An important property of cyclic interleaving is that it is fair, i.e. there will always come a next turn for all active threads.

Other plausible interleaving strategies are treated in [8]. They can also be adapted to the features of molecular dynamics that will be introduced in the current paper. The strategic interleaving operator for cyclic interleaving is denoted by $\|$. In [8], it was denoted by $\|_{csi}$ to distinguish it from other strategic interleaving operators.

The axioms for cyclic interleaving are given in Table 3. In CSI3, the auxiliary *deadlock at termination* operator $\mathsf{S_D}$ is used. This operator turns termination into deadlock. Its axioms appear in Table 4. In Tables 3 and 4, a stands for an arbitrary action from $\mathcal{A}$.

Henceforth, we will write TA for BTA extended with the strategic interleaving operator for cyclic interleaving, the deadlock at termination operator, and the

Table 3. Axioms for cyclic interleaving

$\parallel(\langle\,\rangle) = \mathsf{S}$	CSI1
$\parallel(\langle \mathsf{S} \rangle \frown \alpha) = \parallel(\alpha)$	CSI2
$\parallel(\langle \mathsf{D} \rangle \frown \alpha) = \mathsf{S_D}(\parallel(\alpha))$	CSI3
$\parallel(\langle \mathsf{tau} \circ x \rangle \frown \alpha) = \mathsf{tau} \circ \parallel(\alpha \frown \langle x \rangle)$	CSI4
$\parallel(\langle x \trianglelefteq a \trianglerighteq y \rangle \frown \alpha) = \parallel(\alpha \frown \langle x \rangle) \trianglelefteq a \trianglerighteq \parallel(\alpha \frown \langle y \rangle)$	CSI5

Table 4. Axioms for deadlock at termination

$\mathsf{S_D}(\mathsf{S}) = \mathsf{D}$	S2D1
$\mathsf{S_D}(\mathsf{D}) = \mathsf{D}$	S2D2
$\mathsf{S_D}(\mathsf{tau} \circ x) = \mathsf{tau} \circ \mathsf{S_D}(x)$	S2D3
$\mathsf{S_D}(x \trianglelefteq a \trianglerighteq y) = \mathsf{S_D}(x) \trianglelefteq a \trianglerighteq \mathsf{S_D}(y)$	S2D4

axioms from Tables 3 and 4, and TA(A) for TA with the set of basic actions $\mathcal{A}$ fixed to be the set A.

We can prove that each closed term over the signature of TA can be reduced to a closed term over the signature of BTA.

Theorem 1 (Elimination). *For all closed terms p over the signature of* TA*, there exists a closed term q over the signature of* BTA *such that $p = q$ is derivable from the axioms of* TA*.*

The following proposition, concerning the cyclic interleaving of a thread vector of length 1, is easily proved using Theorem 1.

Proposition 1. *For all closed terms p over the signature of* TA*, the equation $\parallel(\langle p \rangle) = p$ is derivable from the axioms of* TA*.*

The equation $\parallel(\langle p \rangle) = p$ from Proposition 1 expresses the obvious fact that in the cyclic interleaving of a thread vector of length 1 no proper interleaving is involved.

The following are useful properties of the deadlock at termination operator which are proved using Theorem 1 as well.

Proposition 2. *For all closed terms $p_1, \ldots, p_n$ over the signature of* TA*, the following equations are derivable from the axioms of* TA*:*

$$\mathsf{S_D}(\parallel(\langle p_1 \rangle \frown \ldots \frown \langle p_n \rangle)) = \parallel(\langle \mathsf{S_D}(p_1) \rangle \frown \ldots \frown \langle \mathsf{S_D}(p_n) \rangle)\ , \tag{1}$$

$$\mathsf{S_D}(\mathsf{S_D}(p_1)) = \mathsf{S_D}(p_1)\ . \tag{2}$$

The structural operational semantics of TA is described by the transition rules given in Tables 2 and 5. In Table 5, a stands for an arbitrary action from $\mathcal{A}_{\mathsf{tau}}$.

Table 5. Transition rules for cyclic interleaving and deadlock at termination

$$\frac{x_1{\downarrow}, \ldots, x_k{\downarrow}, \langle x_{k+1}, \rho\rangle \xrightarrow{a} \langle x'_{k+1}, \rho'\rangle}{\langle \|(\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha), \rho\rangle \xrightarrow{a} \langle \|(\alpha \frown \langle x'_{k+1}\rangle), \rho'\rangle} \quad (k \geq 0)$$

$$\frac{x_1{\updownarrow}, \ldots, x_k{\updownarrow}, x_l{\uparrow}, \langle x_{k+1}, \rho\rangle \xrightarrow{a} \langle x'_{k+1}, \rho'\rangle}{\langle \|(\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha), \rho\rangle \xrightarrow{a} \langle \|(\alpha \frown \langle \mathsf{D}\rangle \frown \langle x'_{k+1}\rangle), \rho'\rangle} \quad (k \geq l > 0)$$

$$\frac{x_1{\downarrow}, \ldots, x_k{\downarrow}}{\|(\langle x_1\rangle \frown \ldots \frown \langle x_k\rangle){\downarrow}} \qquad \frac{x_1{\updownarrow}, \ldots, x_k{\updownarrow}, x_l{\uparrow}}{\|(\langle x_1\rangle \frown \ldots \frown \langle x_k\rangle){\uparrow}} \quad (k \geq l > 0)$$

$$\frac{\langle x, \rho\rangle \xrightarrow{a} \langle x', \rho'\rangle}{\langle \mathsf{S_D}(x), \rho\rangle \xrightarrow{a} \langle \mathsf{S_D}(x'), \rho'\rangle} \qquad \frac{x{\updownarrow}}{\mathsf{S_D}(x){\uparrow}}$$

Bisimulation equivalence is also a congruence with respect to the strategic interleaving operator for cyclic interleaving and the deadlock at termination operator. This follows immediately from the fact that the transition rules for TA constitute a complete transition system specification in the relaxed panth format (see e.g. [24]). The axioms given in Tables 3 and 4 are sound with respect to bisimulation equivalence.

We have taken the operator $\|$ for a unary operator of which the operand denotes a sequence of threads. This matches well with the intuition that an interleaving strategy such as cyclic interleaving operates on a thread vector. We can look upon the operator $\|$ as if there is actually an n-ary operator, of which the operands denote threads, for every $n \in \mathbb{N}$. From Section 12, we will freely look upon the operator $\|$ in this way for the purpose of more concise expression of definitions and results concerning the projective limit model for the thread calculus presented in this paper.

4 Interaction between Threads and Services

In this section, we extend the thread algebra introduced in Section 3 to a theory that takes the interaction between threads and services into account.

It is assumed that there is a fixed but arbitrary set of *foci* $\mathcal{F}$ and a fixed but arbitrary set of *methods* $\mathcal{M}$. For the set of basic actions $\mathcal{A}$, we take the set $FM = \{f.m \mid f \in \mathcal{F}, m \in \mathcal{M}\}$. Each focus plays the role of a name of a service provided by the execution environment that can be requested to process a command. Each method plays the role of a command proper. Performing a basic action $f.m$ is taken as making a request to the service named f to process the command m.

For each $f \in \mathcal{F}$, we introduce a *thread-service composition* operator $\_ \mathbin{/_f} \_$. These operators have a thread as first argument and a service as second argument. Intuitively, $p \mathbin{/_f} H$ is the thread that results from processing all basic actions performed by thread p that are of the form $f.m$ by service H. When

Table 6. Axioms for thread-service composition

$\mathsf{S} /_f H = \mathsf{S}$		TSC1
$\mathsf{D} /_f H = \mathsf{D}$		TSC2
$(\mathsf{tau} \circ x) /_f H = \mathsf{tau} \circ (x /_f H)$		TSC3
$(x \trianglelefteq g.m \trianglerighteq y) /_f H = (x /_f H) \trianglelefteq g.m \trianglerighteq (y /_f H)$	if $f \neq g$	TSC4
$(x \trianglelefteq f.m \trianglerighteq y) /_f H = \mathsf{tau} \circ (x /_f \frac{\partial}{\partial m} H)$	if $H(\langle m \rangle) = \mathsf{T}$	TSC5
$(x \trianglelefteq f.m \trianglerighteq y) /_f H = \mathsf{tau} \circ (y /_f \frac{\partial}{\partial m} H)$	if $H(\langle m \rangle) = \mathsf{F}$	TSC6
$(x \trianglelefteq f.m \trianglerighteq y) /_f H = \mathsf{D}$	if $H(\langle m \rangle) = \mathsf{R}$	TSC7

a basic action $f.m$ performed by thread p is processed by H, it is turned into the action tau and postconditional composition is removed in favour of action prefixing on the basis of the reply value produced by H.

A service is represented by a function $H : \mathcal{M}^+ \to \{\mathsf{T}, \mathsf{F}, \mathsf{R}\}$ with the property that $H(\alpha) = \mathsf{R} \Rightarrow H(\alpha \frown \langle m \rangle) = \mathsf{R}$ for all $\alpha \in \mathcal{M}^+$ and $m \in \mathcal{M}$. This function is called the *reply* function of the service. We write $\mathcal{RF}$ for the set of all reply functions. Given a reply function $H \in \mathcal{RF}$ and a method $m \in \mathcal{M}$, the *derived* reply function of H after processing m, written $\frac{\partial}{\partial m} H$, is defined by $\frac{\partial}{\partial m} H(\alpha) = H(\langle m \rangle \frown \alpha)$.

The connection between a reply function H and the service represented by it can be understood as follows:

- If $H(\langle m \rangle) = \mathsf{T}$, the request to process command m is accepted by the service, the reply is positive and the service proceeds as $\frac{\partial}{\partial m} H$.
- If $H(\langle m \rangle) = \mathsf{F}$, the request to process command m is accepted by the service, the reply is negative and the service proceeds as $\frac{\partial}{\partial m} H$.
- If $H(\langle m \rangle) = \mathsf{R}$, the request to process command m is refused by the service.

Henceforth, we will identify a reply function with the service represented by it.

The axioms for the thread-service composition operators are given in Table 6. In this table, f and g stand for arbitrary foci from $\mathcal{F}$ and m stands for an arbitrary method from $\mathcal{M}$. Axiom TSC3 expresses that the action tau is always accepted. Axioms TSC5 and TSC6 make it clear that tau arises as the residue of processing commands. Therefore, tau is not connected to a particular focus, and is always accepted. Axiom TSC7 expresses that refusal of processing a command leads to deadlock.

Henceforth, we write $\mathrm{TA}^{\mathrm{tsc}}$ for $\mathrm{TA}(\mathit{FM})$ extended with the thread-service composition operators and the axioms from Table 6.

We can prove that each closed term over the signature of $\mathrm{TA}^{\mathrm{tsc}}$ can be reduced to a closed term over the signature of $\mathrm{BTA}(\mathit{FM})$.

Theorem 2 (Elimination). *For all closed terms p over the signature of* $\mathrm{TA}^{\mathrm{tsc}}$*, there exists a closed term q over the signature of* $\mathrm{BTA}(\mathit{FM})$ *such that $p = q$ is derivable from the axioms of* $\mathrm{TA}^{\mathrm{tsc}}$*.*

Table 7. Transition rules for thread-service composition

$$\frac{\langle x,\rho\rangle \xrightarrow{\mathsf{tau}} \langle x',\rho'\rangle}{\langle x \mathbin{/_f} H,\rho\rangle \xrightarrow{\mathsf{tau}} \langle x' \mathbin{/_f} H,\rho'\rangle} \qquad \frac{\langle x,\rho\rangle \xrightarrow{g.m} \langle x',\rho'\rangle}{\langle x \mathbin{/_f} H,\rho\rangle \xrightarrow{g.m} \langle x' \mathbin{/_f} H,\rho'\rangle}\; f \neq g$$

$$\frac{\langle x,\rho\rangle \xrightarrow{f.m} \langle x',\rho'\rangle}{\langle x \mathbin{/_f} H,\rho\rangle \xrightarrow{\mathsf{tau}} \langle x' \mathbin{/_f} \frac{\partial}{\partial m} H,\rho'\rangle}\; H(\langle m\rangle) \neq \mathsf{R},\, (f.m, H(\langle m\rangle)) \in \rho(\langle\,\rangle)$$

$$\frac{\langle x,\rho\rangle \xrightarrow{f.m} \langle x',\rho'\rangle}{x \mathbin{/_f} H\uparrow}\; H(\langle m\rangle) = \mathsf{R} \qquad \frac{x\downarrow}{x \mathbin{/_f} H\downarrow} \qquad \frac{x\uparrow}{x \mathbin{/_f} H\uparrow}$$

The following are useful properties of the deadlock at termination operator in the presence of both cyclic interleaving and thread-service composition which are proved using Theorem 2.

Proposition 3. *For all closed terms* $p_1, \ldots, p_n$ *over the signature of* $\mathrm{TA}^{\mathrm{tsc}}$*, the following equations are derivable from the axioms of* $\mathrm{TA}^{\mathrm{tsc}}$*:*

$$\mathsf{S_D}(\|(\langle p_1\rangle \curvearrowright \ldots \curvearrowright \langle p_n\rangle)) = \|(\langle \mathsf{S_D}(p_1)\rangle \curvearrowright \ldots \curvearrowright \langle \mathsf{S_D}(p_n)\rangle)\ , \quad (1)$$

$$\mathsf{S_D}(\mathsf{S_D}(p_1)) = \mathsf{S_D}(p_1)\ , \quad (2)$$

$$\mathsf{S_D}(p_1 \mathbin{/_f} H) = \mathsf{S_D}(p_1) \mathbin{/_f} H\ . \quad (3)$$

The structural operational semantics of $\mathrm{TA}^{\mathrm{tsc}}$ is described by the transition rules given in Tables 2, 5 and 7. In Table 7, f and g stand for arbitrary foci from $\mathcal{F}$ and m stands for an arbitrary method from $\mathcal{M}$.

Bisimulation equivalence is also a congruence with respect to the thread-service composition operators. This follows immediately from the fact that the transition rules for these operators are in the path format. The axioms given in Table 6 are sound with respect to bisimulation equivalence.

We end this section with a precise statement of what we mean by a regular service. Let $H \in \mathcal{RF}$. Then the set $\Delta(H) \subseteq \mathcal{RF}$ is inductively defined by the following rules:

- $H \in \Delta(H)$;
- if $m \in \mathcal{M}$ and $H' \in \Delta(H)$, then $\frac{\partial}{\partial m} H' \in \Delta(H)$.

We say that H is a *regular* service if $\Delta(H)$ is a finite set.

In Section 5, we need the notion of a regular service in Proposition 6. In the state-based approach to describe services that will be introduced in Section 6, a service can be described using a finite set of states if and only if it is regular.

5 Recursion

We proceed to recursion in the current setting. In this section, T stands for either BTA, TA, $\mathrm{TA}^{\mathrm{tsc}}$ or $\mathrm{TC}_{\mathrm{md}}$ ($\mathrm{TC}_{\mathrm{md}}$ will be introduced in Section 9). We extend

Table 8. Axioms for recursion

$\mathsf{fix}_x(t) = t[\mathsf{fix}_x(t)/x]$		REC1
$y = t[y/x] \Rightarrow y = \mathsf{fix}_x(t)$	if x guarded in t	REC2
$\mathsf{fix}_x(x) = \mathsf{D}$		REC3

T with recursion by adding variable binding operators and axioms concerning these additional operators. We will write $T + \text{REC}$ for the resulting theory.

For each variable x, we add a variable binding *recursion* operator fix_x to the operators of T.

Let t be a term over the signature of $T+\text{REC}$. Then x is *guarded* in t if each occurrence of x in t is within some subterm of the form $t_1 \trianglelefteq a \trianglerighteq t_2$.

Let t be a term over the signature of $T + \text{REC}$ such that $\mathsf{fix}_x(t)$ is a closed term. Then $\mathsf{fix}_x(t)$ stands for a solution of the equation $x = t$. We are only interested in models of $T + \text{REC}$ in which $x = t$ has a unique solution if x is guarded in t. If x is unguarded in t, then D is always one of the solutions of $x = t$. We stipulate that $\mathsf{fix}_x(t)$ stands for D if x is unguarded in t.

We add the axioms for recursion given in Table 8 to the axioms of T. In this table, t stands for an arbitrary term over the signature of $T + \text{REC}$. The side-condition added to REC2 restricts the terms for which t stands to the terms in which x is guarded. For a fixed t such that $\mathsf{fix}_x(t)$ is a closed term, REC1 expresses that $\mathsf{fix}_x(t)$ is a solution of $x = t$ and REC2 expresses that this solution is the only one if x is guarded in t. REC3 expresses that $\mathsf{fix}_x(x)$ is the non-unique solution D of the equation $x = x$.

Let t and t' be terms over the signature of $T + \text{REC}$ such that $\mathsf{fix}_x(t)$ and $\mathsf{fix}_x(t')$ are closed terms and $t = t'$ is derivable by either applying an axiom of T in either direction or axiom REC1 from left to right. Then it is straightforwardly proved, using the necessary and sufficient condition for preservation of solutions given in [27], that $x = t$ and $x = t'$ have the same set of solutions in any model of T. Hence, if $x = t$ has a unique solution, then $x = t'$ has a unique solution and those solutions are the same. This justifies a weakening of the side-condition of axiom REC2 in the case where $\mathsf{fix}_x(t)$ is a closed term. In that case, it can be replaced by "x is guarded in some term t' for which $t = t'$ is derivable by applying axioms of T in either direction and/or axiom REC1 from left to right".

Theorem 1 states that the strategic interleaving operator for cyclic interleaving and the deadlock at termination operator can be eliminated from closed terms over the signature of TA. Theorem 2 states that beside that the thread-service composition operators can be eliminated from closed terms over the signature of TA^{tsc}. These theorems do not state anything concerning closed terms over the signature of TA+REC or closed terms over the signature of TA^{tsc}+REC. The following three propositions concern the case where the operand of the strategic interleaving operator for cyclic interleaving is a sequence of closed terms over the signature of BTA+REC of the form $\mathsf{fix}_x(t)$, the case where the operand of

Table 9. Transition rules for recursion

$\dfrac{\langle t[\mathsf{fix}_x(t)/x], \rho\rangle \xrightarrow{a} \langle x', \rho'\rangle}{\langle \mathsf{fix}_x(t), \rho\rangle \xrightarrow{a} \langle x', \rho'\rangle}$	$\dfrac{t[\mathsf{fix}_x(t)/x]\downarrow}{\mathsf{fix}_x(t)\downarrow}$	$\dfrac{t[\mathsf{fix}_x(t)/x]\uparrow}{\mathsf{fix}_x(t)\uparrow}$	$\dfrac{}{\mathsf{fix}_x(x)\uparrow}$

the deadlock at termination operator is such a closed term, and the case where the first operand of a thread-service composition operator is such a closed term.

Proposition 4. *Let t and t' be terms over the signature of* BTA+REC *such that $\mathsf{fix}_x(t)$ and $\mathsf{fix}_y(t')$ are closed terms. Then there exists a term t'' over the signature of* BTA+REC *such that $\|(\langle \mathsf{fix}_x(t)\rangle \frown \langle \mathsf{fix}_y(t')\rangle) = \mathsf{fix}_z(t'')$ is derivable from the axioms of* TA+REC*.*

Proposition 5. *Let t be a term over the signature of* BTA+REC *such that $\mathsf{fix}_x(t)$ is a closed term. Then there exists a term t' over the signature of* BTA+REC *such that $\mathsf{S_D}(\mathsf{fix}_x(t)) = \mathsf{fix}_y(t')$ is derivable from the axioms of* TA+REC*.*

Proposition 6. *Let t be a term over the signature of* BTA+REC *such that $\mathsf{fix}_x(t)$ is a closed term. Moreover, let $f \in \mathcal{F}$ and let $H \in \mathcal{RF}$ be a regular service. Then there exists a term t' over the signature of* BTA+REC *such that $\mathsf{fix}_x(t) /_f H = \mathsf{fix}_y(t')$ is derivable from the axioms of* $\mathrm{TA}^{\mathrm{tsc}}$+REC*.*

Propositions 4, 5 and 6 state that the strategic interleaving operator for cyclic interleaving, the deadlock at termination operator and the thread-service composition operators can be eliminated from closed terms of the form $\|(\langle \mathsf{fix}_x(t)\rangle \frown \langle \mathsf{fix}_y(t')\rangle)$, $\mathsf{S_D}(\mathsf{fix}_x(t))$ and $\mathsf{fix}_x(t) /_f H$, where t and t' are terms over the signature of BTA+REC and H is a regular service. Moreover, they state that the resulting term is a closed term of the form $\mathsf{fix}_z(t'')$, where t'' is a term over the signature of BTA+REC. Proposition 4 generalizes to the case where the operand is a sequence of length greater than 2.

The transition rules for recursion are given in Table 9. In this table, x and t stand for an arbitrary variable and an arbitrary term over the signature of $T +$ REC, respectively, such that $\mathsf{fix}_x(t)$ is a closed term. In this table, a stands for an arbitrary action from $\mathcal{A}_{\mathsf{tau}}$.

The transition rules for recursion given in Table 9 are not in the path format. They can be put in the generalized panth format from [24], which guarantees that bisimulation equivalence is a congruence with respect to the recursion operators, but that requires generalizations of many notions that are material to structural operational semantics. The axioms given in Table 8 are sound with respect to bisimulation equivalence.

6 State-Based Description of Services

In this section, we introduce the state-based approach to describe services that will be used in Section 7 to describe services for molecular dynamics. This approach is similar to the approach to describe state machines introduced in [13].

In this approach, a service is described by

– a set of states S;
– an initial state $s_0 \in S$;
– an effect function $\mathit{eff} : \mathcal{M} \times S \to S$;
– a yield function $\mathit{yld} : \mathcal{M} \times S \to \{\mathsf{T}, \mathsf{F}, \mathsf{R}\}$.

The set S contains the states in which the service may be; and the functions *eff* and *yld* give, for each method m and state s, the state and reply, respectively, that result from processing m in state s.

We define a cumulative effect function $\mathit{ceff} : \mathcal{M}^* \to S$ in terms of s_0 and *eff* as follows:

$$\mathit{ceff}(\langle\,\rangle) = s_0$$

$$\mathit{ceff}(\alpha \curvearrowright \langle m \rangle) = \mathit{eff}(m, \mathit{ceff}(\alpha)) \ .$$

We define a service $H : \mathcal{M}^+ \to \{\mathsf{T}, \mathsf{F}, \mathsf{R}\}$ in terms of *ceff* and *yld* as follows:

$$H(\alpha \curvearrowright \langle m \rangle) = \mathit{yld}(m, \mathit{ceff}(\alpha)) \ .$$

We consider H to be the service described by S, s_0, *eff* and *yld*.

Note that $H(\langle m \rangle) = \mathit{yld}(m, s_0)$ and $\frac{\partial}{\partial m} H$ is the service obtained by taking $\mathit{eff}(m, s_0)$ instead of s_0 as the initial state.

7 Services for Molecular Dynamics

In this section, we describe services which concerns molecular dynamics. The services introduced here elaborates on an informal description of molecular dynamics given in [2].

The states of a service for molecular dynamics resemble collections of molecules composed of atoms and the methods of this service transform the structure of molecules like in chemical reactions. An atom can have *fields* and each of those fields can contain an atom. An atom together with the ones it has links to via fields can be viewed as a submolecule, and a submolecule that is not contained in a larger submolecule can be viewed as a molecule. Thus, the collection of molecules that make up the state of the service can be viewed as a fluid. By means of methods, new atoms can be created, fields can be added to and removed from atoms, and the contents of fields of atoms can be examined and modified. A few methods use a *spot* to put an atom in or to get an atom from. By means of methods, the contents of spots can be compared and modified as well. Creating an atom is thought of as turning an element of a given set of *proto-atoms* into an atom. If there are no proto-atoms left, then atoms can no longer be created.

It is assumed that a set Spot of *spots* and a set Field of *fields* have been given. It is also assumed that a countable set PAtom of *proto-atoms* such that $\bot \notin \mathsf{PAtom}$ and a bijection $\mathit{patom} : [1, \mathrm{card}(\mathsf{PAtom})] \to \mathsf{PAtom}$ have been given. Although the set of proto-atoms may be infinite, there exists at any time only a finite number of atoms. Each of those atoms has only a finite number of fields. A modular dynamics service has the following methods:

– for each $s \in \mathsf{Spot}$, a *create atom method* $s\,!$;
– for each $s, s' \in \mathsf{Spot}$, a *set spot method* $s = s'$;
– for each $s, \in \mathsf{Spot}$, a *clear spot method* $s = 0$;
– for each $s, s' \in \mathsf{Spot}$, an *equality test method* $s == s'$;
– for each $s \in \mathsf{Spot}$, an *undefinedness test method* $s == 0$;
– for each $s \in \mathsf{Spot}$ and $v \in \mathsf{Field}$, a *add field method* $s \mathbin{/} v$;
– for each $s \in \mathsf{Spot}$ and $v \in \mathsf{Field}$, a *remove field method* $s \setminus v$;
– for each $s \in \mathsf{Spot}$ and $v \in \mathsf{Field}$, a *has field method* $s \mid v$;
– for each $s \in \mathsf{Spot}$ and $v \in \mathsf{Field}$, a *set field method* $s.v = s'$;
– for each $s \in \mathsf{Spot}$ and $v \in \mathsf{Field}$, a *get field method* $s = s'.v$.

We write $\mathcal{M}_{\mathsf{md}}$ for the set of all methods of a modular dynamics service. It is assumed that $\mathcal{M}_{\mathsf{md}} \subseteq \mathcal{M}$.

The state of a modular dynamics service comprises the contents of all spots, the fields of the existing atoms, and the contents of those fields. The methods of a modular dynamics service can be explained as follows:

– $s\,!$: if an atom can be created, then the contents of spot s becomes a newly created atom and the reply is T; otherwise, nothing changes and the reply is F;
– $s = s'$: the contents of spot s' becomes the contents of spot s too and the reply is T;
– $s = 0$: the contents of spot s becomes undefined and the reply is T;
– $s == s'$: if the contents of spot s equals the contents of spot s', then nothing changes and the reply is T; otherwise, nothing changes and the reply is F;
– $s == 0$: if the contents of spot s is undefined, then nothing changes and the reply is T; otherwise, nothing changes and the reply is F;
– $s \mathbin{/} v$: if the contents of spot s is an atom and v is not yet a field of that atom, then v is added (with undefined contents) to the fields of that atom and the reply is T; otherwise, nothing changes and the reply is F;
– $s \setminus v$: if the contents of spot s is an atom and v is a field of that atom, then v is removed from the fields of that atom and the reply is T; otherwise, nothing changes and the reply is F;
– $s \mid v$: if the contents of spot s is an atom and v is a field of that atom, then nothing changes and the reply is T; otherwise, nothing changes and the reply is F;
– $s.v = s'$: if the contents of spot s is an atom and v is a field of that atom, then the contents of spot s' becomes the contents of that field and the reply is T; otherwise, nothing changes and the reply is F;
– $s = s'.v$: if the contents of spot s' is an atom and v is a field of that atom, then the contents of that field becomes the contents of spot s and the reply is T; otherwise, nothing changes and the reply is F.

In the explanation given above, wherever we say that the contents of a spot or field becomes the contents of another spot or field, this is meant to imply that the former contents becomes undefined if the latter contents is undefined.

Table 10. Effect function for a molecular dynamics service

$\mathit{eff}(s\,!, (\sigma, \alpha)) =$	
$\quad (\sigma \oplus [s \mapsto \mathit{new}(\mathrm{dom}(\alpha))], \alpha \oplus [\mathit{new}(\mathrm{dom}(\alpha)) \mapsto [\,]])$	if $\mathit{new}(\mathrm{dom}(\alpha)) \neq \bot$
$\mathit{eff}(s\,!, (\sigma, \alpha)) = (\sigma, \alpha)$	if $\mathit{new}(\mathrm{dom}(\alpha)) = \bot$
$\mathit{eff}(s = s', (\sigma, \alpha)) = (\sigma \oplus [s \mapsto \sigma(s')], \alpha)$	
$\mathit{eff}(s = 0, (\sigma, \alpha)) = (\sigma \oplus [s \mapsto \bot], \alpha)$	
$\mathit{eff}(s == s', (\sigma, \alpha)) = (\sigma, \alpha)$	
$\mathit{eff}(s == 0, (\sigma, \alpha)) = (\sigma, \alpha)$	
$\mathit{eff}(s \,/\, v, (\sigma, \alpha)) =$	
$\quad (\sigma, \alpha \oplus [\sigma(s) \mapsto \alpha(\sigma(s)) \oplus [v \mapsto \bot]])$	if $\sigma(s) \neq \bot \wedge v \notin \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{eff}(s \,/\, v, (\sigma, \alpha)) = (\sigma, \alpha)$	if $\sigma(s) = \bot \vee v \in \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{eff}(s \setminus v, (\sigma, \alpha)) = (\sigma, \alpha \oplus [\sigma(s) \mapsto \alpha(\sigma(s)) \lhd \{v\}])$	if $\sigma(s) \neq \bot \wedge v \in \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{eff}(s \setminus v, (\sigma, \alpha)) = (\sigma, \alpha)$	if $\sigma(s) = \bot \vee v \notin \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{eff}(s \mid v, (\sigma, \alpha)) = (\sigma, \alpha)$	
$\mathit{eff}(s.v = s', (\sigma, \alpha)) =$	
$\quad (\sigma, \alpha \oplus [\sigma(s) \mapsto \alpha(\sigma(s)) \oplus [v \mapsto \sigma(s')]])$	if $\sigma(s) \neq \bot \wedge v \in \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{eff}(s.v = s', (\sigma, \alpha)) = (\sigma, \alpha)$	if $\sigma(s) = \bot \vee v \notin \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{eff}(s = s'.v, (\sigma, \alpha)) = (\sigma \oplus [s \mapsto \alpha(\sigma(s))(v)], \alpha)$	if $\sigma(s') \neq \bot \wedge v \in \mathrm{dom}(\alpha(\sigma(s')))$
$\mathit{eff}(s = s'.v, (\sigma, \alpha)) = (\sigma, \alpha)$	if $\sigma(s') = \bot \vee v \notin \mathrm{dom}(\alpha(\sigma(s')))$
$\mathit{eff}(m, (\sigma, \alpha)) = (\sigma, \alpha)$	if $m \notin \mathcal{M}_{\mathsf{md}}$

The state-based description of a modular dynamics service is as follows:

$$S = \{(\sigma, \alpha) \in \mathit{SS} \times \mathit{AS} \mid \mathrm{rng}(\sigma) \subseteq \mathrm{dom}(\alpha) \cup \{\bot\} \wedge \\ \forall a \in \mathrm{dom}(\alpha) \bullet \mathrm{rng}(\alpha(a)) \subseteq \mathrm{dom}(\alpha) \cup \{\bot\}\}\,,$$

where

$$\mathit{SS} = \mathsf{Spot} \to (\mathsf{PAtom} \cup \{\bot\})$$

$$\mathit{AS} = \bigcup_{A \in \mathcal{P}_{\mathrm{fin}}(\mathsf{PAtom})} (A \to \bigcup_{F \in \mathcal{P}_{\mathrm{fin}}(\mathsf{Field})} (F \to (\mathsf{PAtom} \cup \{\bot\})))\,;$$

s_0 is some $(\sigma, \alpha) \in S$; and *eff* and *yld* are defined in Tables 10 and 11. We use the following notation for functions: $[\,]$ for the empty function; $[d \mapsto r]$ for the function f with $\mathrm{dom}(f) = \{d\}$ such that $f(d) = r$; $f \oplus g$ for the function h with $\mathrm{dom}(h) = \mathrm{dom}(f) \cup \mathrm{dom}(g)$ such that for all $d \in \mathrm{dom}(h)$, $h(d) = f(d)$

Table 11. Yield function for a molecular dynamics service

$\mathit{yld}(s\,!, (\sigma, \alpha)) = \mathsf{T}$	if $\mathit{new}(\mathrm{dom}(\alpha)) \neq \bot$
$\mathit{yld}(s\,!, (\sigma, \alpha)) = \mathsf{F}$	if $\mathit{new}(\mathrm{dom}(\alpha)) = \bot$
$\mathit{yld}(s = s', (\sigma, \alpha)) = \mathsf{T}$	
$\mathit{yld}(s = 0, (\sigma, \alpha)) = \mathsf{T}$	
$\mathit{yld}(s == s', (\sigma, \alpha)) = \mathsf{T}$	if $\sigma(s) = \sigma(s')$
$\mathit{yld}(s == s', (\sigma, \alpha)) = \mathsf{F}$	if $\sigma(s) \neq \sigma(s')$
$\mathit{yld}(s == 0, (\sigma, \alpha)) = \mathsf{T}$	if $\sigma(s) = \bot$
$\mathit{yld}(s == 0, (\sigma, \alpha)) = \mathsf{F}$	if $\sigma(s) \neq \bot$
$\mathit{yld}(s\,/\,v, (\sigma, \alpha)) = \mathsf{T}$	if $\sigma(s) \neq \bot \land v \notin \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{yld}(s\,/\,v, (\sigma, \alpha)) = \mathsf{F}$	if $\sigma(s) = \bot \lor v \in \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{yld}(s \setminus v, (\sigma, \alpha)) = \mathsf{T}$	if $\sigma(s) \neq \bot \land v \in \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{yld}(s \setminus v, (\sigma, \alpha)) = \mathsf{F}$	if $\sigma(s) = \bot \lor v \notin \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{yld}(s \mid v, (\sigma, \alpha)) = \mathsf{T}$	if $\sigma(s) \neq \bot \land v \in \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{yld}(s \mid v, (\sigma, \alpha)) = \mathsf{F}$	if $\sigma(s) = \bot \lor v \notin \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{yld}(s.v = s', (\sigma, \alpha)) = \mathsf{T}$	if $\sigma(s) \neq \bot \land v \in \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{yld}(s.v = s', (\sigma, \alpha)) = \mathsf{F}$	if $\sigma(s) = \bot \lor v \notin \mathrm{dom}(\alpha(\sigma(s)))$
$\mathit{yld}(s = s'.v, (\sigma, \alpha)) = \mathsf{T}$	if $\sigma(s') \neq \bot \land v \in \mathrm{dom}(\alpha(\sigma(s')))$
$\mathit{yld}(s = s'.v, (\sigma, \alpha)) = \mathsf{F}$	if $\sigma(s') = \bot \lor v \notin \mathrm{dom}(\alpha(\sigma(s')))$
$\mathit{yld}(m, (\sigma, \alpha)) = \mathsf{R}$	if $m \notin \mathcal{M}_{\mathsf{md}}$

if $d \notin \mathrm{dom}(g)$ and $h(d) = g(d)$ otherwise; and $f \triangleleft D$ for the function g with $\mathrm{dom}(g) = \mathrm{dom}(f) \setminus D$ such that for all $d \in \mathrm{dom}(g)$, $g(d) = f(d)$. The function $\mathit{new} : \mathcal{P}_{\mathrm{fin}}(\mathsf{PAtom}) \to (\mathsf{PAtom} \cup \{\bot\})$ is defined by

$$\mathit{new}(A) = \mathit{patom}(m + 1) \;\text{ if } m < \mathrm{card}(\mathsf{PAtom})\;,$$
$$\mathit{new}(A) = \bot \qquad\qquad\;\; \text{ if } m \geq \mathrm{card}(\mathsf{PAtom})\;,$$

where $m = \max\{n \mid \mathit{patom}(n) \in A\}$.

Let $(\sigma, \alpha) \in S$, let $s \in \mathsf{Spot}$, let $a \in \mathrm{dom}(\alpha)$, and let $v \in \mathrm{dom}(\alpha(a))$. Then $\sigma(s)$ is the contents of spot s if $\sigma(s) \neq \bot$, v is a field of atom a, and $\alpha(a)(v)$ is the contents of field v of atom a if $\alpha(a)(v) \neq \bot$. The contents of spot s is undefined if $\sigma(s) = \bot$, and the contents of field v of atom a is undefined if $\alpha(a)(v) = \bot$. Notice that $\mathrm{dom}(\alpha)$ is taken for the set of all existing atoms. Therefore, the contents of each spot, i.e. each element of $\mathrm{rng}(\sigma)$, must be in $\mathrm{dom}(\alpha)$ if the contents is defined. Moreover, for each existing atom a, the contents of each of its fields, i.e. each element of $\mathrm{rng}(\alpha(a))$, must be in $\mathrm{dom}(\alpha)$ if the contents is

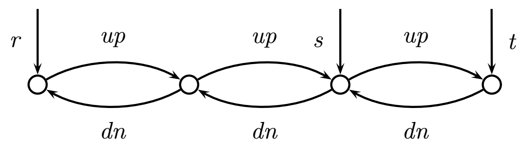

Fig. 1. Molecule yielded by thread P_4

defined. The function *new* turns proto-atoms into atoms. After all proto-atoms have been turned into atoms, *new* yields $\perp$. This can only happen if the number of proto-atoms is finite.

We write $\mathcal{MDS}$ for the set of all modular dynamics services and MDS_0 for the modular dynamics service where s_0 is the unique $(\sigma, \alpha) \in S$ such that $\alpha = [\,]$.

We conclude this section with a simple example of the use of the methods of molecular dynamics services. In this example, we will write $f(m)$ instead of $f.m$. The reasons for this change of notation will be explained in Section 8.

Example 1. Consider the threads

$$P_{n+1} = \mathsf{md}(r\,!) \circ \mathsf{md}(t = r) \circ Q_n$$

where

$$Q_0 \quad = \mathsf{S}\ ,$$

$$Q_{i+1} = \mathsf{md}(s = t) \circ \mathsf{md}(t\,!) \circ \mathsf{md}(s \mathbin{/} up) \circ \mathsf{md}(t \mathbin{/} dn) \circ \mathsf{md}(s.up = t) \circ \mathsf{md}(t.dn = s) \circ Q_i\ .$$

The processing of all basic actions performed by thread P_4 by service MDS_0 yields the molecule depicted in Figure 1.

8 Syntactic Issues

In this short section, we treat some syntactic issues concerning molecular dynamics.

The notation for the methods of molecular dynamics services introduced in Section 7 has a rather mathematical style. The mathematical style fits in with the mathematical nature of the thread calculus being developed in this paper. However, it makes the notation $f.m$ less suitable in the case where m is a method of molecular dynamics services. Therefore, we will henceforth write $f(m)$ instead of $f.m$ if $m \in \mathcal{M}_{\mathsf{md}}$.

A less mathematical style fits in with a program notation for threads, such as the one provided by program algebra [6]. If a less mathematical style is desirable, we propose to use the following notation:

$\mathsf{CA}{:}s$	instead of $s\,!$;	$\mathsf{AF}{:}s{:}v$	instead of $s\ /\ v$;
$\mathsf{SS}{:}s{:}s'$	instead of $s = s'$;	$\mathsf{RF}{:}s{:}v$	instead of $s \setminus v$;
$\mathsf{CS}{:}s$	instead of $s = 0$;	$\mathsf{HF}{:}s{:}v$	instead of $s \mid v$;
$\mathsf{ET}{:}s{:}s'$	instead of $s == s'$;	$\mathsf{SF}{:}s{:}v{:}s'$	instead of $s.v = s'$;
$\mathsf{UT}{:}s$	instead of $s == 0$;	$\mathsf{GF}{:}s{:}s'{:}v$	instead of $s = s'.v$.

9 A Thread Calculus with Molecular Dynamics

In this section, $\mathrm{TC}_{\mathrm{md}}$ is introduced. $\mathrm{TC}_{\mathrm{md}}$ is a version of $\mathrm{TA}^{\mathrm{tsc}}$ with built-in features of molecular dynamics and additional operators to restrict the use of spots. Because spots are means of access to atoms, restriction of their use can be needed.

Like in $\mathrm{TA}^{\mathrm{tsc}}$, it is assumed that there is a fixed but arbitrary set of foci $\mathcal{F}$ and a fixed but arbitrary set of methods $\mathcal{M}$. In addition, it is assumed that $\mathcal{M}_{\mathsf{md}} \subseteq \mathcal{M}$, spots do not occur in $m \in \mathcal{M}$ if $m \notin \mathcal{M}_{\mathsf{md}}$, and $H(\langle m \rangle) = \mathsf{R}$ for all $m \in \mathcal{M}_{\mathsf{md}}$ if $H \notin \mathcal{MDS}$. These additional assumptions express that the methods of molecular dynamics services are supposed to be built-in and that those methods cannot be processed by other services. The last assumption implies that access to atoms is supposed to be provided by molecular dynamics services only. The operators introduced below to restrict the use of spots are not very useful if Spot is a finite set. Therefore, it is also assumed that Spot is an infinite set.

Where restriction of their use is concerned, spots are thought of as names by which atoms are located. Restriction of the use of spots serves a similar purpose as restriction of the use of names in the π-calculus [25].

For each $f \in \mathcal{F}$ and $s \in \mathsf{Spot}$, we add a *restriction* operator local_s^f to the operators of $\mathrm{TA}^{\mathrm{tsc}}$.

Let $f \in \mathcal{F}$, $s \in \mathsf{Spot}$ and p be a term over the signature of $\mathrm{TC}_{\mathrm{md}}$. Then $\mathsf{local}_s^f(p)$ restricts s for use in p, i.e. it makes s local to p, where basic actions of the form $f.m$ are concerned. In that way, s is available to access some atom via f to p only. The availability of s to access some atom via a focus other than f is not restricted.

The restriction operators of $\mathrm{TC}_{\mathrm{md}}$ are name binding operators of a special kind. In $\mathsf{local}_s^f(p)$, the occurrence of s in the subscript is a binding occurrence, but the scope of that occurrence is not simply p: an occurrence of s in p lies within the scope of the binding occurrence if and only if that occurrence is in a basic action of the form $f.m$. As a result, the set of free names of a term, the set of bound names of a term, and substitutions of names for free occurrences of names in a term always have a bearing on some focus. Spot s is a *free name* of term p with respect to focus f if there is an occurrence of s in p that is in a basic action of the form $f.m$ that is not in a subterm of the form $\mathsf{local}_s^f(p')$. Spot s is a *bound name* of term p with respect to focus f if there is an occurrence of s in p that is in a basic action of the form $f.m$ that is in a subterm of the form $\mathsf{local}_s^f(p')$. The *substitution* of spot s' for free occurrences of spot s with respect to focus f in term p replaces in p all occurrences of s in basic actions of the form $f.m$ that are not in a subterm of the form $\mathsf{local}_s^f(p')$ by s'.

Table 12. Axioms for restriction

$\mathsf{local}_s^f(t) = \mathsf{local}_{s'}^f(t[s'/s]^f)$	if $s' \notin \mathrm{fn}^f(t)$	R1
$\mathsf{local}_s^f(\mathsf{S}) = \mathsf{S}$		R2
$\mathsf{local}_s^f(\mathsf{D}) = \mathsf{D}$		R3
$\mathsf{local}_s^f(\mathsf{tau} \circ x) = \mathsf{tau} \circ \mathsf{local}_s^f(x)$		R4
$\mathsf{local}_s^f(x \trianglelefteq g.m \trianglerighteq y) = \mathsf{local}_s^f(x) \trianglelefteq g.m \trianglerighteq \mathsf{local}_s^f(y)$	if $f \neq g$	R5
$\mathsf{local}_s^f(x \trianglelefteq f.m \trianglerighteq y) = \mathsf{local}_s^f(x) \trianglelefteq f.m \trianglerighteq \mathsf{local}_s^f(y)$	if $s \notin \mathrm{n}(m)$	R6
$\Vert(\langle \mathsf{local}_s^f(x)\rangle \frown \alpha) = \mathsf{local}_s^f(\Vert(\langle x\rangle \frown \alpha))$	if $s \notin \mathrm{fn}^f(\alpha)$	R7
$\mathsf{S}_\mathsf{D}(\mathsf{local}_s^f(x)) = \mathsf{local}_s^f(\mathsf{S}_\mathsf{D}(x))$		R8
$\mathsf{local}_s^f(x) \ /_g\ H = \mathsf{local}_s^f(x \ /_g\ H)$	if $f \neq g$	R9
$\mathsf{local}_s^f(x) \ /_f\ H = x \ /_f\ H$	if $H(\langle s == 0\rangle) \neq \mathsf{F}$	R10
$\mathsf{local}_s^f(\mathsf{local}_{s'}^g(x)) = \mathsf{local}_{s'}^g(\mathsf{local}_s^f(x))$		R11

In Appendix A, $\mathrm{fn}^f(p)$, the set of free names of term p with respect to focus f, $\mathrm{bn}^f(p)$, the set of bound names of term p with respect to focus f, and $p[s'/s]^f$, the substitution of name s' for free occurrences of name s with respect to focus f in term p, are defined. We will write $\mathrm{n}(m)$, where $m \in \mathcal{M}$, for the set of all names occurring in m.

Par abus de langage, we will henceforth refer to term p as the scope of the binding occurrence of s in $\mathsf{local}_s^f(p)$.

The axioms for restriction are given in Table 12. In this table, s and s' stand for arbitrary spots from Spot, and t stands for an arbitrary term over the signature of $\mathrm{TC}_{\mathrm{md}}$. The crucial axioms are R1, R7, R9 and R10. Axiom R1 asserts that alpha-convertible restrictions are equal. Axiom R7 expresses that, in case the scope of a restricted spot is a thread in a thread vector, the scope can safely be extended to the strategic interleaving of that thread vector if the restricted spot is not freely used by the other threads in the thread vector through the focus concerned. Axiom R9 expresses that, in case the scope of a restricted spot is a thread that is composed with a service and the foci concerned are different, the scope can safely be extended to the thread-service composition. Axiom R10 expresses that, in case the scope of a restricted spot is a thread that is composed with a service and the foci concerned are equal, the restriction can be raised if the contents of the restricted spot is undefined – indicating that it is not in use by any thread to access some atom.

Axiom R1, together with the assumption that Spot is infinite, has far-reaching consequences: in case axiom R7 or axiom R10 cannot be applied directly because the condition on the restricted spot is not satisfied, it can always be applied after application of axiom R1.

Next we give a simple example of the use of restriction.

Example 2. In the expressions $p \trianglelefteq \mathsf{md}(s.v = s'.w) \trianglerighteq q$ and $p \trianglelefteq \mathsf{md}(s.v.w = s') \trianglerighteq q$, where p and q are terms over the signature of $\mathrm{TC}_{\mathrm{md}}$, a get field method is combined in different ways with a set field method. This results in expressions that are not terms over the signature of $\mathrm{TC}_{\mathrm{md}}$. However, these expressions could be considered abbreviations for the following terms over the signature of $\mathrm{TC}_{\mathrm{md}}$:

$$\mathsf{local}^{\mathsf{md}}_{s''}(\mathsf{md}(s'' = s'.w) \circ (p \trianglelefteq \mathsf{md}(s.v = s'') \trianglerighteq q)) \;,$$

$$\mathsf{local}^{\mathsf{md}}_{s''}(\mathsf{md}(s'' = s.v) \circ (p \trianglelefteq \mathsf{md}(s''.w = s') \trianglerighteq q)) \;,$$

where $s'' \notin \mathrm{fn}^{\mathsf{md}}(p) \cup \mathrm{fn}^{\mathsf{md}}(q)$. The importance of the use of restriction here is that it prevents interference by means of s'' in the case where interleaving is involved, as illustrated by the following derivable equations:

$$\begin{aligned} &\|(\langle \mathsf{md}(s'' = s'.w) \circ (p \trianglelefteq \mathsf{md}(s.v = s'') \trianglerighteq q)\rangle \curvearrowright \langle \mathsf{md}(s'' = 0) \circ \mathsf{S}\rangle) \\ &\quad = \mathsf{md}(s'' = s'.w) \circ \mathsf{md}(s'' = 0) \circ (p \trianglelefteq \mathsf{md}(s.v = s'') \trianglerighteq q) \;, \end{aligned}$$

$$\begin{aligned} &\|(\langle \mathsf{local}^{\mathsf{md}}_{s''}(\mathsf{md}(s'' = s'.w) \circ (p \trianglelefteq \mathsf{md}(s.v = s'') \trianglerighteq q))\rangle \curvearrowright \langle \mathsf{md}(s'' = 0) \circ \mathsf{S}\rangle) \\ &\quad = \mathsf{local}^{\mathsf{md}}_{s'''}(\mathsf{md}(s''' = s'.w) \circ \mathsf{md}(s'' = 0) \circ (p \trianglelefteq \mathsf{md}(s.v = s''') \trianglerighteq q)) \;, \end{aligned}$$

where $s''' \notin \mathrm{fn}^{\mathsf{md}}(p) \cup \mathrm{fn}^{\mathsf{md}}(q) \cup \{s''\}$. The first equation shows that there is interference if restriction is not used, whereas the second equation shows that there is no interference if restriction is used. Notice that derivation of the second equation requires that axiom R1 is applied before axiom R7 is applied.

Not every closed term over the signature of $\mathrm{TC}_{\mathrm{md}}$ can be reduced to a closed term over the signature of $\mathrm{BTA}(\mathit{FM})$, e.g. a term of the form $\mathsf{local}^{f}_{s}(p \trianglelefteq f.m \trianglerighteq q)$, where p and q are closed terms over the signature of $\mathrm{BTA}(\mathit{FM})$, cannot be reduced further if $s \in \mathrm{n}(m)$. To elaborate on this remark, we introduce the notion of a basic term. The set $\mathcal{B}$ of *basic terms* is inductively defined by the following rules:

- $\mathsf{S}, \mathsf{D} \in \mathcal{B}$;
- if $p \in \mathcal{B}$, then $\mathsf{tau} \circ p \in \mathcal{B}$;
- if $f \in \mathcal{F}$, $m \in \mathcal{M}$ and $p, q \in \mathcal{B}$, then $p \trianglelefteq f.m \trianglerighteq q \in \mathcal{B}$;
- if $f \in \mathcal{F}$, $m \in \mathcal{M}$, $s \in \mathrm{n}(m)$ and $p, q \in \mathcal{B}$, then $\mathsf{local}^{f}_{s}(p \trianglelefteq f.m \trianglerighteq q) \in \mathcal{B}$.

We can prove that each closed term over the signature of $\mathrm{TC}_{\mathrm{md}}$ can be reduced to a term from $\mathcal{B}$.

Theorem 3 (Elimination). *For all closed terms p over the signature of $\mathrm{TC}_{\mathrm{md}}$, there exists a term $q \in \mathcal{B}$ such that $p = q$ is derivable from the axioms of $\mathrm{TC}_{\mathrm{md}}$.*

Proof. The proof follows the same line as the proof of Theorem 2 presented in [7]. This means that it is a proof by induction on the structure of p in which some cases boil down to proving a lemma by some form of induction or another, mostly structural induction again. Here, we have to consider the additional case

$p \equiv \mathsf{local}_s^f(p')$, where we can restrict ourselves to basic terms p'. This case is easily proved by structural induction using axioms R2–R6 and R11. In the case $p \equiv \|(\langle p'_1\rangle \frown \ldots \frown \langle p'_n\rangle)$, where we can restrict ourselves to basic terms $p'_1, \ldots, p'_n$, we have to consider the additional case $p'_1 \equiv \mathsf{local}_s^f(p''_1 \trianglelefteq f.m \trianglerighteq p'''_1)$ with $s \in \mathrm{n}(m)$. After applying axioms R1 and R7 at the beginning, this case goes analogous to the case $p'_1 \equiv p''_1 \trianglelefteq f.m \trianglerighteq p'''_1$. In the case $p \equiv \mathsf{S_D}(p')$, where we can restrict ourselves to basic terms p', we have to consider the additional case $p' \equiv \mathsf{local}_s^f(p'' \trianglelefteq f.m \trianglerighteq p''')$ with $s \in \mathrm{n}(m)$. After applying axiom R8 at the beginning, this case goes analogous to the case $p' \equiv p'' \trianglelefteq f.m \trianglerighteq p'''$. In the case $p \equiv p' /_f H$, where we can restrict ourselves to basic terms p', we have to consider the additional case $p' \equiv \mathsf{local}_s^f(p'' \trianglelefteq f.m \trianglerighteq p''')$ with $s \in \mathrm{n}(m)$. After applying axiom R9 or axioms R1 and R10, this case goes analogous to the case $p' \equiv p'' \trianglelefteq f.m \trianglerighteq p'''$. □

The following proposition, concerning the cyclic interleaving of a thread vector of length 1 in the presence of thread-service composition and restriction, is easily proved using Theorem 3.

Proposition 7. *For all closed terms p over the signature of $\mathrm{TC}_{\mathrm{md}}$, the equation $\|(\langle p\rangle) = p$ is derivable from the axioms of $\mathrm{TC}_{\mathrm{md}}$.*

Proof. The proof follows the same line as the proof of Proposition 1 presented in [7]. This means that it is a rather trivial proof by induction on the structure of p. Here, we have to consider the additional case $p \equiv \mathsf{local}_s^f(p' \trianglelefteq f.m \trianglerighteq p'')$ with $s \in \mathrm{n}(m)$. This case goes similar to the case $p \equiv p' \trianglelefteq f.m \trianglerighteq p''$. Axioms R1 and R7 are applied at the beginning and at the end. □

The following are useful properties of the deadlock at termination operator in the presence of thread-service composition and restriction which are proved using Theorem 3.

Proposition 8. *For all closed terms $p_1, \ldots, p_n$ over the signature of $\mathrm{TC}_{\mathrm{md}}$, the following equations are derivable from the axioms of $\mathrm{TC}_{\mathrm{md}}$:*

$$\mathsf{S_D}(\|(\langle p_1\rangle \frown \ldots \frown \langle p_n\rangle)) = \|(\langle \mathsf{S_D}(p_1)\rangle \frown \ldots \frown \langle \mathsf{S_D}(p_n)\rangle)\ , \tag{1}$$

$$\mathsf{S_D}(\mathsf{S_D}(p_1)) = \mathsf{S_D}(p_1)\ , \tag{2}$$

$$\mathsf{S_D}(p_1 /_f H) = \mathsf{S_D}(p_1) /_f H\ . \tag{3}$$

Proof. The proof follows the same line as the proof of Proposition 3 presented in [7]. This means that equation (1) is proved by induction on the sum of the depths plus one of $p_1, \ldots, p_n$ and case distinction on the structure of p_1, and that equations (2) and (3) are proved by induction on the structure of p_1. For each of the equations, we have to consider the additional case $p_1 \equiv \mathsf{local}_s^f(p'_1 \trianglelefteq f.m \trianglerighteq p''_1)$ with $s \in \mathrm{n}(m)$. For each of the equations, this case goes similar to the case $p_1 \equiv p'_1 \trianglelefteq f.m \trianglerighteq p''_1$. In case of equation (1), axioms R1 and R7 are applied at the beginning and at the end. In case of equation (2), axiom R8 is applied at the beginning and at the end. In case of equation (3), axiom R9 or axioms R1 and R10 are applied at the beginning and at the end. □

Table 13. Approximation induction principle

$\bigwedge_{n \geq 0} \pi_n(x) = \pi_n(y) \Rightarrow x = y$	AIP

Proposition 9. *Let t be a term over the signature of* BTA+REC *such that* $\mathsf{fix}_x(t)$ *is a closed term. Then there exists a term t' over the signature of* BTA+REC *such that* $\mathsf{local}_s^f(\mathsf{fix}_x(t)) = \mathsf{fix}_y(t')$ *is derivable from the axioms of* $\mathrm{TC}_{\mathrm{md}}$+REC *provided for all actions $g.m$ occurring in t either $f \neq g$ or $s \notin \mathrm{n}(m)$.*

Proof. The proof follows the same line as the proofs of Propositions 4–6 presented in [7]. □

We refrain from providing a structural operational semantics of $\mathrm{TC}_{\mathrm{md}}$. In the case where we do not deviate from the style of structural operational semantics adopted for BTA, TA and $\mathrm{TA}^{\mathrm{tsc}}$, the obvious way to deal with restriction involves the introduction of bound actions, together with a scope opening transition rule (for restriction) and a scope closing transition rule (for thread-service composition), like in [25]. This would complicate matters to such an extent that a structural operational semantics of $\mathrm{TC}_{\mathrm{md}}$ would add at most marginally to a better understanding. Therefore, we provide instead a projective limit model of $\mathrm{TC}_{\mathrm{md}}$ in Section 12.

10 Approximation Induction Principle

Each closed term over the signature of $\mathrm{TC}_{\mathrm{md}}$ denotes a finite thread, i.e. a thread of which the length of the sequences of actions that it can perform is bounded. However, not each closed term over the signature of $\mathrm{TC}_{\mathrm{md}}$+REC denotes a finite thread: recursion gives rise to infinite threads. Closed terms over the signature of $\mathrm{TC}_{\mathrm{md}}$+REC that denote the same infinite thread cannot always be proved equal by means of the axioms of $\mathrm{TC}_{\mathrm{md}}$+REC. In this section, we introduce the approximation induction principle to reason about infinite threads.

The approximation induction principle, AIP in short, is based on the view that two threads are identical if their approximations up to any finite depth are identical. The approximation up to depth n of a thread is obtained by cutting it off after performing a sequence of actions of length n.

AIP is the infinitary conditional equation given in Table 13. Here, following [6], approximation of depth n is phrased in terms of a unary *projection* operator π_n. The axioms for the projection operators are given in Table 14. In this table, a stands for an arbitrary member of $\mathcal{A}_{\mathsf{tau}}$ and s stands for an arbitrary member of Spot.

Let T stand for either $\mathrm{TC}_{\mathrm{md}}$ or $\mathrm{TC}_{\mathrm{md}}$+REC. Then we will write T+AIP for T extended with the projections operators π_n and the axioms P0–P4 and axiom AIP.

AIP holds in the projective limit models for $\mathrm{TC}_{\mathrm{md}}$ and $\mathrm{TC}_{\mathrm{md}}$+REC that will be constructed in Sections 12 and 14, respectively. Axiom REC2 is derivable from the axioms of $\mathrm{TC}_{\mathrm{md}}$, axiom REC1 and AIP.

Table 14. Axioms for projection operators

$\pi_0(x) = \mathsf{D}$	P0
$\pi_{n+1}(\mathsf{S}) = \mathsf{S}$	P1
$\pi_{n+1}(\mathsf{D}) = \mathsf{D}$	P2
$\pi_{n+1}(x \trianglelefteq a \trianglerighteq y) = \pi_n(x) \trianglelefteq a \trianglerighteq \pi_n(y)$	P3
$\pi_{n+1}(\mathsf{local}_s^f(x)) = \mathsf{local}_s^f(\pi_{n+1}(x))$	P4

Table 15. Additional axiom for thread forking

$\Vert(\langle x \trianglelefteq \mathsf{nt}(z) \trianglerighteq y\rangle \frown \alpha) = \mathsf{tau} \circ \Vert(\alpha \frown \langle z\rangle \frown \langle x\rangle)$	CSI6
$\mathsf{local}_s^f(x \trianglelefteq \mathsf{nt}(z) \trianglerighteq y) = \mathsf{local}_s^f(x) \trianglelefteq \mathsf{nt}(\mathsf{local}_s^f(z)) \trianglerighteq \mathsf{local}_s^f(y)$	R12
$\pi_{n+1}(x \trianglelefteq \mathsf{nt}(z) \trianglerighteq y) = \pi_n(x) \trianglelefteq \mathsf{nt}(\pi_n(z)) \trianglerighteq \pi_n(y)$	P5

11 Thread Forking and Restriction

In this section, we use restriction to model a form of thread forking found in object-oriented programming languages. For that purpose, we have to adapt the strategic interleaving operator for cyclic interleaving such that it supports a basic form of thread forking. We will do so like in [8].

We add the ternary *forking postconditional composition* operator $\_ \trianglelefteq \mathsf{nt}(\_) \trianglerighteq \_$ to the operators of $\mathrm{TC}_{\mathrm{md}}$. Like action prefixing, we introduce *forking prefixing* as an abbreviation: $\mathsf{nt}(p) \circ q$, where p and q are terms over the signature of $\mathrm{TC}_{\mathrm{md}}$ with thread forking, abbreviates $q \trianglelefteq \mathsf{nt}(p) \trianglerighteq q$. Henceforth, the postconditional composition operators introduced in Section 2 will be called non-forking postconditional composition operators.

The forking postconditional composition operator has the same shape as non-forking postconditional composition operators. Formally, no action is involved in forking postconditional composition. However, for an operational intuition, in $p \trianglelefteq \mathsf{nt}(r) \trianglerighteq q$, $\mathsf{nt}(r)$ can be considered a thread forking action. It represents the act of forking off thread r. Like with real actions, a reply is produced. We consider the case where forking off a thread will never be blocked or fail. In that case, it always produces a positive reply. The action tau arises as a residue of forking off a thread. In [8], we treat several interleaving strategies for threads that support a basic form of thread forking. All of them deal with cases where forking may be blocked and/or may fail. We believe that perfect forking is a suitable abstraction when illustrating the use of restriction. In [8], $\mathsf{nt}(r)$ was formally considered a thread forking action. We experienced afterwards that this leads to unnecessary complications in expressing definitions and results concerning the projective limit model for the thread algebra developed in this paper (see Section 12).

The axioms for $\mathrm{TC}_{\mathrm{md}}$ with thread forking, written $\mathrm{TC}_{\mathrm{md}}^{\mathrm{tf}}$, are the axioms of $\mathrm{TC}_{\mathrm{md}}$ and axioms CSI6 and R12 from Table 15. The axioms for $\mathrm{TC}_{\mathrm{md}}$+AIP

Table 16. Additional transition rules for thread forking

$$\frac{}{\langle x \trianglelefteq \mathsf{nt}(p) \trianglerighteq y, \rho\rangle \xrightarrow{\mathsf{nt}(p)} \langle x, \rho\rangle}$$	
$$\frac{x_1{\downarrow}, \ldots, x_k{\downarrow}, \langle x_{k+1}, \rho\rangle \xrightarrow{\mathsf{nt}(y)} \langle x'_{k+1}, \rho'\rangle}{\langle \|(\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha), \rho\rangle \xrightarrow{\mathsf{tau}} \langle \|(\alpha \frown \langle y\rangle \frown \langle x'_{k+1}\rangle), \rho'\rangle}$$	$(k \geq 0)$
$$\frac{x_1{\updownarrow}, \ldots, x_k{\updownarrow}, x_l{\uparrow}, \langle x_{k+1}, \rho\rangle \xrightarrow{\mathsf{nt}(y)} \langle x'_{k+1}, \rho'\rangle}{\langle \|(\langle x_1\rangle \frown \ldots \frown \langle x_{k+1}\rangle \frown \alpha), \rho\rangle \xrightarrow{\mathsf{tau}} \langle \|(\alpha \frown \langle \mathsf{D}\rangle \frown \langle y\rangle \frown \langle x'_{k+1}\rangle), \rho'\rangle}$$	$(k \geq l > 0)$

with thread forking, written $\mathrm{TC}^{\mathrm{tf}}_{\mathrm{md}}$+AIP, are the axioms of $\mathrm{TC}_{\mathrm{md}}$ and axioms CSI6, R12 and P5 from Table 15.

Not all results concerning the strategic interleaving operator for cyclic interleaving go through if this basic form of thread forking is added. Theorem 3 goes through if we add the following rule to the inductive definition of $\mathcal{B}$ given in Section 9: if $p, q, r \in \mathcal{B}$, then $p \trianglelefteq \mathsf{nt}(r) \trianglerighteq q \in \mathcal{B}$. Proposition 7 and the first part of Proposition 8 go through for closed terms in which the forking postconditional composition operator does not occur only. Proposition 4 goes through for terms in which the forking postconditional composition operator does not occur. It is an open problem whether Proposition 4 goes through for terms in which the forking postconditional composition operator does occur.

The transition rules for cyclic interleaving with thread forking are given in Tables 5 and 16. Here, we use a binary relation $\langle \_, \rho\rangle \xrightarrow{\alpha} \langle \_, \rho'\rangle$ for each $\alpha \in \mathcal{A}_{\mathsf{tau}} \cup \{\mathsf{nt}(p) \mid p \text{ closed term over signature of } \mathrm{TC}^{\mathrm{tf}}_{\mathrm{md}}\}$ and $\rho, \rho' \in \mathcal{E}$. Bisimulation equivalence is a congruence with respect to cyclic interleaving with thread forking. The transition labels containing terms do not complicate matters because there are no volatile operators involved (see e.g. [26]).

We introduce expressions of the form $p \trianglelefteq \mathsf{nt}(s, s', r) \trianglerighteq q$, where p, q and r are terms over the signature of $\mathrm{TC}^{\mathrm{tf}}_{\mathrm{md}}$ such that $s \notin \mathrm{fn}^{\mathsf{md}}(p) \cup \mathrm{fn}^{\mathsf{md}}(q)$.

The intuition is that $p \trianglelefteq \mathsf{nt}(s, s', r) \trianglerighteq q$ will not only fork off thread r, like $p \trianglelefteq \mathsf{nt}(r) \trianglerighteq q$, but will also have the following side-effect: a new atom is created which is made accessible by means of spot s to the thread being forked off and by means of spot s' to the thread forking off. The new atom serves as a unique identity object associated with the thread being forked off. The spots s and s' serve as the names available in the thread being forked off and the thread forking off, respectively, to refer to that identity object. The spot s corresponds to the self-reference this from Java. The important issue is that s is meant to be locally available only.

An expression of the form $p \trianglelefteq \mathsf{nt}(s, s', r) \trianglerighteq q$, where p, q and r are as above, can be considered an abbreviation for the following term over the signature of $\mathrm{TC}^{\mathrm{tf}}_{\mathrm{md}}$:

$$\mathsf{local}^{\mathsf{tf}}_{s}(\mathsf{tf}(s\,!) \circ \mathsf{tf}(s' = s) \circ (p \trianglelefteq \mathsf{nt}(r) \trianglerighteq q))\ ,$$

where tf is the focus of the molecular dynamics service that is used in thread forking. Restriction is used here to see to it that s does not become globally available.

12 Projective Limit Model for $\mathbf{TC_{md}}$

In this section, we construct a projective limit model for $\mathrm{TC_{md}}$. In this model, which covers finite and infinite threads, threads are represented by infinite sequences of finite approximations.

To express definitions more concisely, the interpretations of the constants and operators from the signature of $\mathrm{TC_{md}}$ in the initial model of $\mathrm{TC_{md}}$ and the projective limit model of $\mathrm{TC_{md}}$ are denoted by the constants and operators themselves. The ambiguity thus introduced could be obviated by decorating the symbols, with different decorations for different models, when they are used to denote their interpretation in a model. However, in this paper, it is always immediately clear from the context how the symbols are used. Moreover, we believe that the decorations are more often than not distracting. Therefore, we leave it to the reader to mentally decorate the symbols wherever appropriate.

The projective limit construction is known as the inverse limit construction in domain theory, the theory underlying the approach of denotational semantics for programming languages (see e.g. [28]). In process algebra, this construction has been applied for the first time by Bergstra and Klop [5].

We will write A_ω for the domain of the initial model of $\mathrm{TC_{md}}$. A_ω consists of the equivalence classes of terms from $\mathcal{B}$ with respect to the equivalence induced by the axioms of $\mathrm{TC_{md}}$. In other words, modulo equivalence, A_ω is $\mathcal{B}$. Henceforth, we will identify terms from $\mathcal{B}$ with their equivalence class where elements of A_ω are concerned.

Each element of A_ω represents a finite thread, i.e. a thread of which the length of the sequences of actions that it can perform is bounded. Below, we will construct a model that covers infinite threads as well. In preparation for that, we define for all n a function that cuts off finite threads from A_ω after performing a sequence of actions of length n.

For all $n \in \mathbb{N}$, we have the *projection* function $\pi_n : A_\omega \to A_\omega$, inductively defined by

$$
\begin{aligned}
&\pi_0(p) = \mathsf{D}\ ,\\
&\pi_{n+1}(\mathsf{S}) = \mathsf{S}\ ,\\
&\pi_{n+1}(\mathsf{D}) = \mathsf{D}\ ,\\
&\pi_{n+1}(p \trianglelefteq a \trianglerighteq q) = \pi_n(p) \trianglelefteq a \trianglerighteq \pi_n(q)\ ,\\
&\pi_{n+1}(\mathsf{local}_s^f(p)) = \mathsf{local}_s^f(\pi_{n+1}(p))\ .
\end{aligned}
$$

For $p \in A_\omega$, $\pi_n(p)$ is called the n-th projection of p. It can be thought of as an approximation of p. If $\pi_n(p) \neq p$, then $\pi_{n+1}(p)$ can be thought of as the closest better approximation of p. If $\pi_n(p) = p$, then $\pi_{n+1}(p) = p$ as well. For all $n \in \mathbb{N}$, we will write A_n for $\{\pi_n(p) \mid p \in A_\omega\}$.

The semantic equations given above to define the projection functions have the same shape as the axioms for the projection operators introduced in Section 10. We will come back to this at the end of Section 14.

The properties of the projection operations stated in the following two lemmas will be used frequently in the sequel.

Lemma 1. *For all $p \in A_\omega$ and $n, m \in \mathbb{N}$, $\pi_n(\pi_m(p)) = \pi_{\min\{n,m\}}(p)$.*

Proof. This is easily proved by induction on the structure of p. □

Lemma 2. *For all $p_1, \ldots, p_m \in A_\omega$ and $n \in \mathbb{N}$:*

$$\pi_n(\|(\langle p_1\rangle \curvearrowright \ldots \curvearrowright \langle p_m\rangle)) = \|(\langle \pi_n(p_1)\rangle \curvearrowright \ldots \curvearrowright \langle \pi_n(p_m)\rangle)\ , \tag{1}$$

$$\pi_n(\mathsf{S}_\mathsf{D}(p_1)) = \mathsf{S}_\mathsf{D}(\pi_n(p_1))\ , \tag{2}$$

$$\pi_n(p_1 /_f H) = \pi_n(p_1) /_f H\ . \tag{3}$$

Proof. Equation 1 is straightforwardly proved by induction on $n + m$ and case distinction on the structure of p_1. Equations 2 and 3 are easily proved by induction on the structure of p_1. □

In the projective limit model, which covers finite and infinite threads, threads are represented by *projective sequences*, i.e. infinite sequences $(p_n)_{n\in\mathbb{N}}$ of elements of A_ω such that $p_n \in A_n$ and $p_n = \pi_n(p_{n+1})$ for all $n \in \mathbb{N}$. In other words, a projective sequence is a sequence of which successive components are successive projections of the same thread. The idea is that any infinite thread is fully characterized by the infinite sequence of all its finite approximations. We will write A^∞ for $\{(p_n)_{n\in\mathbb{N}} \mid \bigwedge_{n\in\mathbb{N}}(p_n \in A_n \wedge p_n = \pi_n(p_{n+1}))\}$.

The *projective limit model* of $\mathrm{TC}_{\mathrm{md}}$ consists of the following:

- the set A^∞, the domain of the projective limit model;
- an element of A^∞ for each constant of $\mathrm{TC}_{\mathrm{md}}$;
- an operation on A^∞ for each operator of $\mathrm{TC}_{\mathrm{md}}$;

where those elements of A^∞ and operations on A^∞ are defined as follows:

$$\begin{aligned}
&\mathsf{S} && = (\pi_n(\mathsf{S}))_{n\in\mathbb{N}}\ ,\\
&\mathsf{D} && = (\pi_n(\mathsf{D}))_{n\in\mathbb{N}}\ ,\\
&(p_n)_{n\in\mathbb{N}} \trianglelefteq a \trianglerighteq (q_n)_{n\in\mathbb{N}} && = (\pi_n(p_n \trianglelefteq a \trianglerighteq q_n))_{n\in\mathbb{N}}\ ,\\
&\|(\langle (p_{1n})_{n\in\mathbb{N}}\rangle \curvearrowright \ldots \curvearrowright \langle (p_{mn})_{n\in\mathbb{N}}\rangle) && = (\pi_n(\|(\langle p_{1n}\rangle \curvearrowright \ldots \curvearrowright \langle p_{mn}\rangle)))_{n\in\mathbb{N}}\ ,\\
&\mathsf{S}_\mathsf{D}((p_n)_{n\in\mathbb{N}}) && = (\pi_n(\mathsf{S}_\mathsf{D}(p_n)))_{n\in\mathbb{N}}\ ,\\
&(p_n)_{n\in\mathbb{N}} /_f H && = (\pi_n(p_n /_f H))_{n\in\mathbb{N}}\ ,\\
&\mathsf{local}^f_s((p_n)_{n\in\mathbb{N}}) && = (\pi_n(\mathsf{local}^f_s(p_n)))_{n\in\mathbb{N}}\ .
\end{aligned}$$

Using Lemmas 1 and 2, we easily prove for $(p_n)_{n\in\mathbb{N}}, (q_n)_{n\in\mathbb{N}} \in A^\infty$ and $(p_{1n})_{n\in\mathbb{N}}, \ldots, (p_{mn})_{n\in\mathbb{N}} \in A^\infty$:

– $\pi_n(\pi_{n+1}(p_{n+1} \trianglelefteq a \trianglerighteq q_{n+1})) = \pi_n(p_n \trianglelefteq a \trianglerighteq q_n)$;
– $\pi_n(\pi_{n+1}(\|(\langle {p_1}_{n+1}\rangle \curvearrowright \ldots \curvearrowright \langle {p_m}_{n+1}\rangle))) = \pi_n(\|(\langle {p_1}_n\rangle \curvearrowright \ldots \curvearrowright \langle {p_m}_n\rangle))$;
– $\pi_n(\pi_{n+1}(\mathsf{S}_\mathsf{D}(p_{n+1}))) = \pi_n(\mathsf{S}_\mathsf{D}(p_n))$;
– $\pi_n(\pi_{n+1}(p_{n+1} /_f H)) = \pi_n(p_n /_f H)$;
– $\pi_n(\pi_{n+1}(\mathsf{local}_s^f(p_{n+1}))) = \pi_n(\mathsf{local}_s^f(p_n))$.

From this and the definition of A_n, it follows immediately that the operations defined above are well-defined, i.e. they always yield elements of A^∞.

The initial model can be embedded in a natural way in the projective limit model: each $p \in A_\omega$ corresponds to $(\pi_n(p))_{n\in\mathbb{N}} \in A^\infty$. We extend projection to an operation on A^∞ by defining $\pi_m((p_n)_{n\in\mathbb{N}}) = (p'_n)_{n\in\mathbb{N}}$, where $p'_n = p_n$ if $n < m$ and $p'_n = p_m$ if $n \geq m$. That is, $\pi_m((p_n)_{n\in\mathbb{N}})$ is p_m embedded in A^∞ as described above. Henceforth, we will identify elements of A_ω with their embedding in A^∞ where elements of A^∞ are concerned.

13 Metric Space Structure for Projective Limit Model

Following [23] to some extent, we make A^∞ into a metric space to establish, using Banach's fixed point theorem, that every guarded operation $\phi : A^\infty \to A^\infty$ has a unique fixed point. This is relevant to the expansion of the projective limit model of $\mathrm{TC_{md}}$ to the projective limit model of $\mathrm{TC_{md}}$+REC in Section 14.

An m-ary operation ϕ on A^∞ is a *guarded* operation if for all $p_1, \ldots, p_m$, $p'_1, \ldots, p'_m \in A^\infty$ and $n \in \mathbb{N}$:

$$\pi_n(p_1) = \pi_n(p'_1) \land \ldots \land \pi_n(p_m) = \pi_n(p'_m) \\ \Rightarrow \pi_{n+1}(\phi(p_1, \ldots, p_m)) = \pi_{n+1}(\phi(p'_1, \ldots, p'_m)) \;.$$

We say that ϕ is an *unguarded* operation if ϕ is not a guarded operation.

The notion of guarded operation, which originates from [29], supersedes the notion of guard used in [23].

In the remainder of this section, as well as in Sections 14 and 15, we assume known the notions of metric space, completion of a metric space, dense subset in a metric space, continuous function on a metric space, limit in a metric space and contracting function on a metric space, and Banach's fixed point theorem. The definitions of the above-mentioned notions concerning metric spaces and Banach's fixed point theorem can, for example, be found in [15]. In this paper, we will consider ultrametric spaces only. A metric space (M, d) is an *ultrametric space* if for all $p, p', p'' \in M$, $d(p, p') \leq \max\{d(p, p''), d(p'', p')\}$.

We define a distance function $d : A^\infty \times A^\infty \to \mathbb{R}$ by

$$d(p, p') = 2^{-\min\{n\in\mathbb{N} \mid \pi_n(p) \neq \pi_n(p')\}} \quad \text{if } p \neq p' \;, \\ d(p, p') = 0 \quad \text{if } p = p' \;.$$

It is easy to verify that (A^∞, d) is a metric space. The following theorem summarizes the basic properties of this metric space.

Theorem 4.

1. *(A^∞, d) is an ultrametric space;*
2. *(A^∞, d) is the metric completion of the metric space (A_ω, d'), where d' is the restriction of d to A_ω;*
3. *A_ω is dense in A^∞;*
4. *the operations $\pi_n : A^\infty \to A_n$ are continuous;*
5. *for all $p \in A^\infty$ and $n \in \mathbb{N}$, $d(\pi_n(p), p) < 2^{-n}$, hence $\lim_{n\to\infty} \pi_n(p) = p$.*

Proof. These properties are general properties of metric spaces constructed in the way pursued here. Proofs of Properties 1–3 can be found in [29]. A proof of Property 4 can be found in [18]. Property 5 is proved as follows. It follows from Lemma 1, by passing to the limit and using that the projection operations are continuous and A_ω is dense in A^∞, that $\pi_n(\pi_m(p)) = \pi_{\min\{n,m\}}(p)$ for $p \in A^\infty$ as well. Hence, $\min\{m \in \mathbb{N} \mid \pi_m(\pi_n(p)) \neq \pi_m(p)\} > n$, and consequently $d(\pi_n(p), p) < 2^{-n}$. □

The basic properties given above are used in coming proofs.

The properties of the projection operations stated in the following two lemmas will be used in the proofs of Theorems 5 and 6 given below.

Lemma 3. *For all $p \in A^\infty$ and $n, m \in \mathbb{N}$, $\pi_n(\pi_m(p)) = \pi_{\min\{n,m\}}(p)$.*

Proof. As mentioned above in the proof of Theorem 4, this lemma follows from Lemma 1 by passing to the limit and using that the projection operations are continuous and A_ω is dense in A^∞. □

Lemma 4. *For all $p_1, \ldots, p_m \in A^\infty$ and $n \in \mathbb{N}$:*

$$\pi_n(p_1 \trianglelefteq a \trianglerighteq p_2) = \pi_n(\pi_n(p_1) \trianglelefteq a \trianglerighteq \pi_n(p_2)) \;, \tag{1}$$

$$\pi_n(\|(\langle p_1\rangle \frown \ldots \frown \langle p_m\rangle)) = \pi_n(\|(\langle \pi_n(p_1)\rangle \frown \ldots \frown \langle \pi_n(p_m)\rangle)) \;, \tag{2}$$

$$\pi_n(\mathsf{S_D}(p_1)) = \pi_n(\mathsf{S_D}(\pi_n(p_1))) \;, \tag{3}$$

$$\pi_n(p_1 /_f H) = \pi_n(\pi_n(p_1) /_f H) \;, \tag{4}$$

$$\pi_n(\mathsf{local}_s^f(p_1)) = \pi_n(\mathsf{local}_s^f(\pi_n(p_1))) \;. \tag{5}$$

Proof. It is enough to prove Equations 1–5 for $p_1, \ldots, p_m \in A_\omega$. The lemma will then follow by passing to the limit and using that π_n is continuous and A_ω is dense in A^∞. Equations 1 and 5 follow immediately from Lemma 1 and the definition of π_n. Equations 2–4 follow immediately from Lemmas 1 and 2. □

In the terminology of metric topology, the following theorem states that all operations in the projective limit model of $\mathrm{TC}_{\mathrm{md}}$ are non-expansive. This implies that they are continuous, with respect to the metric topology induced by d, in all arguments.

Theorem 5. *For all $p_1, \ldots, p_m, p'_1, \ldots, p'_m \in A^\infty$:*

$$d(p_1 \trianglelefteq a \trianglerighteq p_2, p'_1 \trianglelefteq a \trianglerighteq p'_2) \leq \max\{d(p_1, p'_1), d(p_2, p'_2)\} , \tag{1}$$

$$d(\|(\langle p_1 \rangle \curvearrowright \ldots \curvearrowright \langle p_m \rangle), \|(\langle p'_1 \rangle \curvearrowright \ldots \curvearrowright \langle p'_m \rangle)) \\ \leq \max\{d(p_1, p'_1), \ldots, d(p_m, p'_m)\} , \tag{2}$$

$$d(\mathsf{S_D}(p_1), \mathsf{S_D}(p'_1)) \leq d(p_1, p'_1) , \tag{3}$$

$$d(p_1 /_f H, p'_1 /_f H) \leq d(p_1, p'_1) , \tag{4}$$

$$d(\mathsf{local}^f_s(p_1), \mathsf{local}^f_s(p'_1)) \leq d(p_1, p'_1) . \tag{5}$$

Proof. Let $k_i = \min\{n \in \mathbb{N} \mid \pi_n(p_i) \neq \pi_n(p'_i)\}$ for $i = 1, 2$, and let $k = \min\{k_1, k_2\}$. Then for all $n \in \mathbb{N}$, we have $n < k$ iff $\pi_n(p_1) = \pi_n(p'_1)$ and $\pi_n(p_2) = \pi_n(p'_2)$. From this and Lemma 4, it follows immediately that $\pi_{k-1}(p_1 \trianglelefteq a \trianglerighteq p_2) = \pi_{k-1}(p'_1 \trianglelefteq a \trianglerighteq p'_2)$. Hence, $k \leq \min\{n \in \mathbb{N} \mid \pi_n(p_1 \trianglelefteq a \trianglerighteq p_2) \neq \pi_n(p'_1 \trianglelefteq a \trianglerighteq p'_2)\}$, which completes the proof for the postconditional composition operators. The proof for the other operators go analogously. □

The notion of guarded operation is defined without reference to metric properties. However, being a guarded operation coincides with having a metric property that is highly relevant to the issue of unique fixed points: an operation on A^∞ is a guarded operation iff it is contracting. This is stated in the following lemma.

Lemma 5. *An m-ary operation ϕ on A^∞ is a guarded operation iff for all $p_1, \ldots, p_m, p'_1, \ldots, p'_m \in A^\infty$:*

$$d(\phi(p_1, \ldots, p_m), \phi(p'_1, \ldots, p'_m)) \leq \tfrac{1}{2} \cdot \max\{d(p_1, p'_1), \ldots, d(p_m, p'_m)\} .$$

Proof. Let $k_i = \min\{n \in \mathbb{N} \mid \pi_n(p_i) \neq \pi_n(p'_i)\}$ for $i = 1, \ldots, m$, and let $k = \min\{k_1, \ldots, k_m\}$. Then for all $n \in \mathbb{N}$, $n < k$ iff $\pi_n(p_1) = \pi_n(p'_1)$ and $\ldots$ and $\pi_n(p_m) = \pi_n(p'_m)$. From this, the definition of a guarded operation and the definition of π_0, it follows immediately that ϕ is a guarded operation iff for all $n < k + 1$, $\pi_n(\phi(p_1, \ldots, p_m)) = \pi_n(\phi(p'_1, \ldots, p'_m))$. Hence, ϕ is a guarded operation iff $k + 1 \leq \min\{n \in \mathbb{N} \mid \pi_n(\phi(p_1, \ldots, p_m)) \neq \pi_n(\phi(p'_1, \ldots, p'_m))\}$, which completes the proof. □

We write ϕ^n, where ϕ is a unary operation on A^∞, for the unary operation on A^∞ that is defined by induction on n as follows: $\phi^0(p) = p$ and $\phi^{n+1}(p) = \phi(\phi^n(p))$.

We have the following important result about guarded operations.

Theorem 6. *Let $\phi{:}A^\infty \to A^\infty$ be a guarded operation. Then ϕ has a unique fixed point, i.e. there exists a unique $p \in A^\infty$ such that $\phi(p) = p$, and $(\pi_n(\phi^n(\mathsf{D})))_{n \in \mathbb{N}}$ is the unique fixed point of ϕ.*

Proof. We have from Theorem 4.2 that (A^∞, d) is a complete metric space and from Lemma 5 that ϕ is contracting. From this, we conclude by Banach's fixed point theorem that ϕ has a unique fixed point. It is easily proved by induction on n, using Lemma 3 and the definition of guarded operation, that

$\pi_n(\pi_{n+1}(\phi^{n+1}(\mathsf{D}))) = \pi_n(\phi^n(\mathsf{D}))$. From this and the definition of A_n, it follows that $(\pi_n(\phi^n(\mathsf{D})))_{n\in\mathbb{N}}$ is an element of A^∞. Moreover, it is easily proved by case distinction between $n = 0$ and $n > 0$, using this equation, Lemma 3 and the definition of guarded operation, that $\pi_n(\phi(\pi_n(\phi^n(\mathsf{D})))) = \pi_n(\pi_n(\phi^n(\mathsf{D})))$. From this, it follows that $(\pi_n(\phi^n(\mathsf{D})))_{n\in\mathbb{N}}$ is a fixed point of ϕ by passing to the limit and using that ϕ is continuous and A_ω is dense in A^∞ (recall that contracting operations are continuous). Because ϕ has a unique fixed point, $(\pi_n(\phi^n(\mathsf{D})))_{n\in\mathbb{N}}$ must be the unique fixed point of ϕ. □

14 Projective Limit Model for $\mathbf{TC_{md}}$+REC

The projective limit model for $\mathrm{TC_{md}}$+REC is obtained by expansion of the projective limit model for $\mathrm{TC_{md}}$ with a single operation $\mathsf{fix}:(A^\infty \to^1 A^\infty) \to A^\infty$ for all the recursion operators.[5]

The operation fix differs from the other operations by taking functions from A^∞ to A^∞ as argument. In agreement with that, for a given assignment in A^∞ for variables, the operand of a recursion operator is interpreted as a function from A^∞ to A^∞. If the recursion operator fix_x is used, then variable x is taken as the variable representing the argument of the function concerned. The interpretation of terms over the signature of $\mathrm{TC_{md}}$+REC will be formally defined in Section 15.

The operation fix is defined as follows:

$$\mathsf{fix}(\phi) = (\pi_n(\phi^n(\mathsf{D})))_{n\in\mathbb{N}} \quad \text{if } \phi \text{ is a guarded operation,}$$
$$\mathsf{fix}(\phi) = (\pi_n(\mathsf{D}))_{n\in\mathbb{N}} \quad \text{if } \phi \text{ is an unguarded operation.}$$

From Theorem 6, we know that every guarded operation $\phi : A^\infty \to A^\infty$ has only one fixed point and that $(\pi_n(\phi^n(\mathsf{D})))_{n\in\mathbb{N}}$ is that fixed point. The justification for the definition of fix for unguarded operations is twofold:

- a function ϕ from A^∞ to A^∞ that is representable by a term over the signature of $\mathrm{TC_{md}}$+REC is an unguarded operation only if D is one of the fixed points of ϕ;
- if D is a fixed point of a function ϕ from A^∞ to A^∞, then $(\pi_n(\mathsf{D}))_{n\in\mathbb{N}} = (\pi_n(\phi^n(\mathsf{D})))_{n\in\mathbb{N}}$.

This implies that, for all function ϕ from A^∞ to A^∞ that are representable by a term over the signature of $\mathrm{TC_{md}}$+REC, fix yields a fixed point. Actually, it is the least fixed point with respect to the approximation relation $\sqsubseteq$ that is introduced in Appendix B. There may be unguarded operations in $A^\infty \to^1 A^\infty$ for which D is not a fixed point. However, those operations are not representable by a term over the signature of $\mathrm{TC_{md}}$+REC.

It is straightforward to verify that, for every guarded operation $\phi : A^\infty \to A^\infty$, $(\pi_n(\phi^n(\mathsf{D})))_{n\in\mathbb{N}} = (\pi_n(\phi^{k(n)}(\mathsf{D})))_{n\in\mathbb{N}}$, where $k(n) = \min\{k \mid \pi_n(\phi^k(\mathsf{D})) =$

[5] Given metric spaces (D, d) and (D', d'), we write $D \to^1 D'$ for the set of all non-expansive functions from (D, d) to (D', d').

$\pi_n(\phi^{k+1}(\mathsf{D}))\}$. The right-hand side of this equation is reminiscent of the definition of the operation introduced in [4] for the selection of a fixed point in a projective limit model for PA, a subtheory of ACP [5] without communication.

We define a distance function $\delta : (A^\infty \to^1 A^\infty) \times (A^\infty \to^1 A^\infty) \to \mathbb{R}$ by

$$\delta(\phi, \psi) = \bigsqcup\{d(\phi(p), \psi(p)) \mid p \in A^\infty\} .$$

The distance function δ is well-defined because for all $p, p' \in A^\infty$, $\delta(p, p') \leq 2^{-1}$. It is easy to verify that $(A^\infty \to^1 A^\infty, \delta)$ is an ultrametric space.

The following theorem states that fix is non-expansive for guarded operations.

Theorem 7. *For all $\phi, \psi \in A^\infty \to^1 A^\infty$ that are guarded operations:*

$$d(\mathsf{fix}(\phi), \mathsf{fix}(\psi)) \leq \delta(\phi, \psi) .$$

Proof. Let $p = \mathsf{fix}(\phi)$ and $q = \mathsf{fix}(\psi)$. Then $\phi(p) = p$, $\psi(q) = q$ and also $d(\phi(p), \psi(q)) = d(p, q)$. We have $d(\phi(p), \phi(q)) \leq \frac{1}{2} \cdot d(p, q)$ by Lemma 5 and $d(\phi(q), \psi(q)) \leq \delta(\phi, \psi)$ by the definition of δ. It follows that $d(\phi(q), \psi(q)) \leq \max\{\frac{1}{2} \cdot d(p, q), \delta(\phi, \psi)\}$. Hence, because $d(\phi(p), \psi(q)) = d(p, q)$, we have $d(p, q) \leq \delta(\phi, \psi)$. That is, $d(\mathsf{fix}(\phi), \mathsf{fix}(\psi)) \leq \delta(\phi, \psi)$. □

Projective limit models of $\mathrm{TC_{md}}$+AIP and $\mathrm{TC_{md}}$+REC+AIP are simply obtained by expanding the projective limit models of $\mathrm{TC_{md}}$ and $\mathrm{TC_{md}}$+REC with the projection operations $\pi_n : A^\infty \to A^\infty$ defined at the end of Section 12.

15 Guarded Recursion Equations

In this section, following [23] to some extent, we introduce the notions of guarded term and guarded recursion equation and show that every guarded recursion equation has a unique solution in A^∞.

Supplementary, in Appendix B, we make A^∞ into a complete partial ordered set and show, using Tarski's fixed point theorem, that every recursion equation has a least solution in A^∞ with respect to the partial order relation concerned.

It is assumed that there is a fixed but arbitrary set of variables $\mathcal{X}$.

Let $P \subseteq A^\infty$ and let $X \subseteq \mathcal{X}$. Then we will write $\mathcal{T}_P$ for the set of all terms over the signature of $\mathrm{TC_{md}}$+REC with parameters from P and $\mathcal{T}_P^X$ for the set of all terms from $\mathcal{T}_P$ in which no other variables than the ones in X have free occurrences.[6]

The interpretation function $[\![\_]\!] : \mathcal{T}_P \to ((\mathcal{X} \to A^\infty) \to A^\infty)$ of terms with parameters from $P \subseteq A^\infty$ is defined as follows:

[6] A term with parameters is a term in which elements of the domain of a model are used as constants naming themselves. For a justification of this mix-up of syntax and semantics in case only one model is under consideration, see e.g. [21].

$$
\begin{aligned}
[\![x]\!](\rho) &= \rho(x)\ ,\\
[\![p]\!](\rho) &= p\ ,\\
[\![\mathsf{S}]\!](\rho) &= \mathsf{S}\ ,\\
[\![\mathsf{D}]\!](\rho) &= \mathsf{D}\ ,\\
[\![t_1 \trianglelefteq a \trianglerighteq t_2]\!](\rho) &= [\![t_1]\!](\rho) \trianglelefteq a \trianglerighteq [\![t_2]\!](\rho)\ ,\\
[\![\|(\langle t_1\rangle \frown \ldots \frown \langle t_n\rangle)]\!](\rho) &= \|(\langle [\![t_1]\!](\rho)\rangle \frown \ldots \frown \langle [\![t_n]\!](\rho)\rangle)\ ,\\
[\![\mathsf{S}_{\mathsf{D}}(t)]\!](\rho) &= \mathsf{S}_{\mathsf{D}}([\![t]\!](\rho))\ ,\\
[\![t \,/_f\, H]\!](\rho) &= [\![t]\!](\rho) \,/_f\, H\ ,\\
[\![\mathsf{local}^s_f(t)]\!](\rho) &= \mathsf{local}^s_f([\![t]\!](\rho))\ ,\\
[\![\mathsf{fix}_x(t)]\!](\rho) &= \mathsf{fix}(\phi)\ ,\\
&\quad \text{where } \phi : A^\infty \to A^\infty \text{ is defined by } \phi(p) = [\![t]\!](\rho \oplus [x \mapsto p])\ .
\end{aligned}
$$

The property stated in the following lemma will be used in the proof of Lemma 7 given below.

Lemma 6. *Let $P \subseteq A^\infty$, let $t \in \mathcal{T}_P$, let $x \in \mathcal{X}$, let $p \in P$, and let $\rho : \mathcal{X} \to A^\infty$. Then $[\![t]\!](\rho \oplus [x \mapsto p]) = [\![t[p/x]]\!](\rho)$.*

Proof. This is easily proved by induction on the structure of t. □

Let $x_1, \ldots, x_n \in \mathcal{X}$, let $X \subseteq \{x_1, \ldots, x_n\}$, let $P \subseteq A^\infty$, and let $t \in \mathcal{T}^X_P$. Moreover, let $\rho : \mathcal{X} \to A^\infty$. Then the *interpretation of t with respect to $x_1, \ldots, x_n$*, written $[\![t]\!]^{x_1,\ldots,x_n}$, is the unique function $\phi : A^{\infty n} \to A^\infty$ such that for all $p_1, \ldots, p_n \in A^\infty$, $\phi(p_1, \ldots, p_n) = [\![t]\!](\rho \oplus [x_1 \mapsto p_1] \oplus \ldots \oplus [x_n \mapsto p_n])$.

The interpretation of t with respect to $x_1, \ldots, x_n$ is well-defined because it is independent of the choice of ρ.

The notion of guarded term defined below is suggested by the fact, stated in Lemma 5 above, that an operation on A^∞ is a guarded operation iff it is contracting. The only guarded operations, and consequently contracting operations, in the projective limit model of $\mathrm{TC}_{\mathrm{md}}$+REC are the postconditional composition operations. Based upon this, we define the notion of guarded term as follows.

Let $P \subseteq A^\infty$. Then the set $\mathcal{G}_P$ of *guarded* terms with parameters from P is inductively defined as follows:

- if $p \in P$, then $p \in \mathcal{G}_P$;
- $\mathsf{S}, \mathsf{D} \in \mathcal{G}_P$;
- if $a \in \mathcal{A}$ and $t_1, t_2 \in \mathcal{T}_P$, then $t_1 \trianglelefteq a \trianglerighteq t_2 \in \mathcal{G}_P$;
- if $t_1, \ldots, t_l \in \mathcal{G}_P$, then $\|(\langle t_1\rangle \frown \ldots \frown \langle t_l\rangle) \in \mathcal{G}_P$;
- if $t \in \mathcal{G}_P$, then $\mathsf{S}_{\mathsf{D}}(t) \in \mathcal{G}_P$;
- if $f \in \mathcal{F}$, $H \in \mathcal{RF}$ and $t \in \mathcal{G}_P$, then $t \,/_f\, H \in \mathcal{G}_P$;
- if $f \in \mathcal{F}$, $s \in \mathsf{Spot}$ and $t \in \mathcal{G}_P$, then $\mathsf{local}^s_f(t) \in \mathcal{G}_P$;
- if $x \in \mathcal{X}$ and $t \in \mathcal{G}_P$, then $\mathsf{fix}_x(t) \in \mathcal{G}_P$.

It is easy to show that $t \in \mathcal{G}_P$ iff all variables that have occurrences in t are guarded in t. The inductive definition of guarded terms given above is more convenient in proofs.

The following lemma states that guarded terms represent operations on A^∞ that are contracting.

Lemma 7. *Let $x_1, \ldots, x_n \in \mathcal{X}$, let $X \subseteq \{x_1, \ldots, x_n\}$, let $P \subseteq A^\infty$, and let $t \in \mathcal{T}_P^X$. Then $t \in \mathcal{G}_P$ only if for all $p_1, \ldots, p_n, p'_1, \ldots, p'_n \in A^\infty$:*

$$d([\![t]\!]^{x_1,\ldots,x_n}(p_1, \ldots, p_n), [\![t]\!]^{x_1,\ldots,x_n}(p'_1, \ldots, p'_n)) \\ \leq \tfrac{1}{2} \cdot \max\{d(p_1, p'_1), \ldots, d(p_n, p'_n)\} \;.$$

Proof. This is easily proved by induction on the structure of t using Theorems 5 and 7, Lemmas 5 and 6, and the fact that the postconditional composition operations are guarded operations. □

A *recursion equation* is an equation $x = t$, where $x \in \mathcal{X}$ and $t \in \mathcal{T}_P^{\{x\}}$ for some $P \subseteq A^\infty$. A recursion equation $x = t$ is a *guarded* recursion equation if $t \in \mathcal{G}_P$ for some $P \subseteq A^\infty$. Let $x = t$ be a recursion equation. Then $p \in A^\infty$ is a *solution* of $x = t$ if $[\![t]\!]^x(p) = p$.

We have the following important result about guarded recursion equations.

Theorem 8. *Every guarded recursion equation has a unique solution in the projective limit model for* $\mathrm{TC}_{\mathrm{md}}$+REC*.*

Proof. Let $x \in \mathcal{X}$, let $P \subseteq A^\infty$, and let $t \in \mathcal{T}_P^{\{x\}}$ be such that $t \in \mathcal{G}_P$. We have from Theorem 4.2 that (A^∞, d) is a complete metric space and from Lemma 7 that $[\![t]\!]^x$ is contracting. From this, we conclude by Banach's fixed point theorem that $[\![t]\!]^x$ has a unique fixed point. Hence, the guarded recursion equation $x = t$ has a unique solution. □

The projection operations and the distance function as defined in this paper match well with our intuitive ideas about finite approximations of threads and closeness of threads, respectively. The suitability of the definitions given in this paper is supported by the fact that guarded operations coincide with contracting operations. However, it is not at all clear whether adaptations of the definitions are feasible and will lead to different uniqueness results.

16 Projective Limit Model for $\mathrm{TC}_{\mathrm{md}}$ with Thread Forking

The construction of the projective limit model for $\mathrm{TC}_{\mathrm{md}}^{\mathrm{tf}}$ follows the same line as the construction of the projective limit model for $\mathrm{TC}_{\mathrm{md}}$. In this section, the construction of the projective limit model for $\mathrm{TC}_{\mathrm{md}}^{\mathrm{tf}}$ is outlined.

Recall that the basic terms of $\mathrm{TC}_{\mathrm{md}}^{\mathrm{tf}}$ include closed terms $p \trianglelefteq \mathsf{nt}(r) \trianglerighteq q$, where p, q and r are basic terms (see Section 11). The domain A'_ω of the initial model of $\mathrm{TC}_{\mathrm{md}}^{\mathrm{tf}}$ consists of the equivalence classes of basic terms of $\mathrm{TC}_{\mathrm{md}}^{\mathrm{tf}}$.

The projection functions $\pi_n : A'_\omega \to A'_\omega$ are the extensions of the projection functions $\pi_n : A_\omega \to A_\omega$ inductively defined by the equations given for $\pi_n : A_\omega \to A_\omega$ in Section 12 and the following equation:

$$\pi_{n+1}(p \trianglelefteq \mathsf{nt}(r) \trianglerighteq q) = \pi_n(p) \trianglelefteq \mathsf{nt}(\pi_n(r)) \trianglerighteq \pi_n(q) \; .$$

For all $n \in \mathbb{N}$, we will write A'_n for $\{\pi_n(p) \mid p \in A'_\omega\}$. Moreover, we will write A'^∞ for $\{(p_n)_{n\in\mathbb{N}} \mid \bigwedge_{n\in\mathbb{N}}(p_n \in A'_n \land p_n = \pi_n(p_{n+1}))\}$.

Lemmas 1 and 2 go through for A'_ω.

The projective limit model of $\mathrm{TC}^{\mathrm{tf}}_{\mathrm{md}}$ consists of the following:

- the set A'^∞, the domain of the projective limit model;
- an element of A'^∞ for each constant of $\mathrm{TC}^{\mathrm{tf}}_{\mathrm{md}}$;
- an operation on A'^∞ for each operator of $\mathrm{TC}^{\mathrm{tf}}_{\mathrm{md}}$.

Those elements of A'^∞ and operations on A'^∞, with the exception of the operation associated with the forking postconditional composition operator, are defined as in the case of the projective limit model for $\mathrm{TC}_{\mathrm{md}}$. The ternary operation on A'^∞ associated with the forking postconditional composition operator is defined as follows:

$$(p_n)_{n\in\mathbb{N}} \trianglelefteq \mathsf{nt}((r_n)_{n\in\mathbb{N}}) \trianglerighteq (q_n)_{n\in\mathbb{N}} = (\pi_n(p_n \trianglelefteq \mathsf{nt}(r_n) \trianglerighteq q_n))_{n\in\mathbb{N}} \; .$$

Using Lemma 1, we easily prove that, for $(p_n)_{n\in\mathbb{N}}, (q_n)_{n\in\mathbb{N}}, (r_n)_{n\in\mathbb{N}} \in A'^\infty$, $\pi_n(\pi_{n+1}(p_{n+1} \trianglelefteq \mathsf{nt}(r_{n+1}) \trianglerighteq q_{n+1})) = \pi_n(p_n \trianglelefteq \mathsf{nt}(r_n) \trianglerighteq q_n)$. From this and the definition of A'_n, it follows immediately that the operation defined above always yield elements of A'^∞.

Lemma 3 goes through for A'^∞. Lemma 4 goes through for A'^∞ as well; and we have in addition that for all $p_1, p_2, p_3 \in A'^\infty$ and $n \in \mathbb{N}$:

$$\pi_n(p_1 \trianglelefteq \mathsf{nt}(p_3) \trianglerighteq p_2) = \pi_n(\pi_n(p_1) \trianglelefteq \mathsf{nt}(\pi_n(p_3)) \trianglerighteq \pi_n(p_2)) \; .$$

Theorem 5 goes through for A'^∞; and we have in addition that for all $p_1, p_2, p_3, p'_1, p'_2, p'_3 \in A'^\infty$:

$$d(p_1 \trianglelefteq \mathsf{nt}(p_3) \trianglerighteq p_2, p'_1 \trianglelefteq \mathsf{nt}(p'_3) \trianglerighteq p'_2) \leq \max\{d(p_1, p'_1), d(p_2, p'_2), d(p_3, p'_3)\} \; .$$

Lemma 5 and Theorem 6 go through for A'^∞. Theorem 7 goes through for A'^∞ as well.

The interpretation function $[\![\_]\!]$ of terms with parameters from P is now defined by the equations given for $[\![\_]\!]$ in Section 15 and the following equation:

$$[\![t_1 \trianglelefteq \mathsf{nt}(t_3) \trianglerighteq t_2]\!](\rho) = [\![t_1]\!](\rho) \trianglelefteq \mathsf{nt}([\![t_3]\!](\rho)) \trianglerighteq [\![t_2]\!](\rho) \; .$$

The set $\mathcal{G}_P$ of guarded terms with parameters from P is now inductively defined by the rules given for $\mathcal{G}_P$ in Section 15 and the following rule:

- if $t_1, t_2, t_3 \in \mathcal{T}_P$, then $t_1 \trianglelefteq \mathsf{nt}(t_3) \trianglerighteq t_2 \in \mathcal{G}_P$.

Lemmas 6 and 7 and Theorem 8 go through for A'^∞.

It is easily proved that the projective limit model for $\mathrm{TC}_{\mathrm{md}}$ is a substructure of the restriction of the projective limit model for $\mathrm{TC}^{\mathrm{tf}}_{\mathrm{md}}$ to the signature of $\mathrm{TC}_{\mathrm{md}}$.

17 Conclusions

In this paper, we have carried on the line of research with which we made a start in [8]. We pursue with this line of research the object to develop a theory about threads, multi-threading and interaction of threads with services that is useful for (a) gaining insight into the semantic issues concerning the multi-threading related features found in contemporary programming languages such as Java and C#, and (b) simplified formal description and analysis of programs in which multi-threading is involved. In this paper, we have extended the theory with features that allow for details of multi-threading that come up where it is intertwined with object-orientation to be dealt with. We regard this extension as just a step towards attaining the above-mentioned object. It is likely that applications of the theory developed so far will make clear that further developments are needed.

There is another line of research that emanated from the work presented in [8]. That line of research concerns the development of a formal approach to design new micro-architectures. The approach should allow for the correctness of new micro-architectures and their anticipated speed-up results to be verified. In [9, 10], we demonstrate the feasibility of an approach that involves the use of thread algebra. The line of research concerned is carried out in the framework of a project investigating micro-threading [14, 22], a technique for speeding up instruction processing on a computer which requires that programs are parallelized by judicious use of thread forking.

The work presented in this paper, was partly carried out in the framework of that project as well. For programs written in programming languages such as Java and C#, compilers will have to take care of the parallelization. In ongoing work, we are investigating parallelization for simple programs, which are close to machine language programs. That work has convinced us that it is desirable to have available an extension of thread algebra like the one presented in this paper when developing parallelization techniques for the compilers referred to above.

It is worth mentioning that the applications of thread algebra exceed the domain of single multi-threaded programs. In [12], we extend the theory with features to cover systems that consist of several multi-threaded programs on various hosts in different networks. To demonstrate its usefulness, we employ the extended theory to develop a simplified, formal representation schema of the design of such systems and to verify a property of all systems designed according to that schema.

A Free and Bound Names, Substitution

In this appendix, we define $\mathrm{fn}^f(p)$, the set of free names of term p with respect to focus f, $\mathrm{bn}^f(p)$, the set of bound names of term p with respect to focus f, and $p[s'/s]^f$, the substitution of name s' for free occurrences of name s with respect to focus f in term p. In Table 17, $\mathrm{fn}^f(p)$ and $\mathrm{bn}^f(p)$ are defined, and in

Table 17. Definition of $\mathrm{fn}^f(p)$ and $\mathrm{bn}^f(p)$

$\mathrm{fn}^f(\mathsf{S}) = \emptyset$		$\mathrm{bn}^f(\mathsf{S}) = \emptyset$	
$\mathrm{fn}^f(\mathsf{D}) = \emptyset$		$\mathrm{bn}^f(\mathsf{D}) = \emptyset$	
$\mathrm{fn}^f(\mathsf{tau} \circ t) = \mathrm{fn}^f(t)$		$\mathrm{bn}^f(\mathsf{tau} \circ t) = \mathrm{bn}^f(t)$	
$\mathrm{fn}^f(t \trianglelefteq g.m \trianglerighteq t') = \mathrm{fn}^f(t) \cup \mathrm{fn}^f(t')$	if $f \neq g$	$\mathrm{bn}^f(t \trianglelefteq g.m \trianglerighteq t') = \mathrm{bn}^f(t) \cup \mathrm{bn}^f(t')$	
$\mathrm{fn}^f(t \trianglelefteq f.m \trianglerighteq t') = \mathrm{fn}^f(t) \cup \mathrm{fn}^f(t') \cup \mathrm{n}(m)$			
$\mathrm{fn}^f(\Vert(\alpha)) = \mathrm{fn}^f(\alpha)$		$\mathrm{bn}^f(\Vert(\alpha)) = \mathrm{bn}^f(\alpha)$	
$\mathrm{fn}^f(\mathsf{S}_\mathsf{D}(t)) = \mathrm{fn}^f(t)$		$\mathrm{bn}^f(\mathsf{S}_\mathsf{D}(t)) = \mathrm{bn}^f(t)$	
$\mathrm{fn}^f(t \,/_g\, H) = \mathrm{fn}^f(t)$		$\mathrm{bn}^f(t \,/_g\, H) = \mathrm{bn}^f(t)$	
$\mathrm{fn}^f(\mathsf{local}_s^g(t)) = \mathrm{fn}^f(t)$	if $f \neq g$	$\mathrm{bn}^f(\mathsf{local}_s^g(t)) = \mathrm{bn}^f(t)$	if $f \neq g$
$\mathrm{fn}^f(\mathsf{local}_s^f(t)) = \mathrm{fn}^f(t) \setminus \{s\}$		$\mathrm{bn}^f(\mathsf{local}_s^f(t)) = \mathrm{bn}^f(t) \cup \{s\}$	
$\mathrm{fn}^f(\langle\,\rangle) = \emptyset$		$\mathrm{bn}^f(\langle\,\rangle) = \emptyset$	
$\mathrm{fn}^f(\langle t\rangle \frown \alpha) = \mathrm{fn}^f(t) \cup \mathrm{fn}^f(\alpha)$		$\mathrm{bn}^f(\langle t\rangle \frown \alpha) = \mathrm{bn}^f(t) \cup \mathrm{bn}^f(\alpha)$	

Table 18. Definition of $p[s'/s]^f$

$\mathsf{S}[s'/s]^f = \mathsf{S}$	
$\mathsf{D}[s'/s]^f = \mathsf{D}$	
$(\mathsf{tau} \circ t)[s'/s]^f = \mathsf{tau} \circ (t[s'/s]^f)$	
$(t \trianglelefteq g.m \trianglerighteq t')[s'/s]^f = (t[s'/s]^f) \trianglelefteq g.m \trianglerighteq (t'[s'/s]^f)$	if $f \neq g$
$(t \trianglelefteq f.m \trianglerighteq t')[s'/s]^f = (t[s'/s]^f) \trianglelefteq f.m[s'/s] \trianglerighteq (t'[s'/s]^f)$	
$\Vert(\alpha)[s'/s]^f = \Vert(\alpha[s'/s]^f)$	
$\mathsf{S}_\mathsf{D}(t)[s'/s]^f = \mathsf{S}_\mathsf{D}(t[s'/s]^f)$	
$(t \,/_g\, H)[s'/s]^f = (t[s'/s]^f) \,/_g\, H$	
$\mathsf{local}_{s''}^g(t)[s'/s]^f = \mathsf{local}_{s''}^g(t[s'/s]^f)$	if $f = g \Rightarrow (s \neq s'' \wedge s' \neq s'')$
$\mathsf{local}_{s''}^f(t)[s'/s]^f = \mathsf{local}_{s'''}^f((t[s'''/s'']^f)[s'/s]^f)$	if $(s \neq s'' \wedge s' = s'')$ $(s''' \notin \mathrm{fn}^f(t) \cup \mathrm{bn}^f(t) \cup \{s, s'\})$
$\mathsf{local}_s^f(t)[s'/s]^f = \mathsf{local}_s^f(t)$	
$\langle\,\rangle[s'/s]^f = \langle\,\rangle$	
$(\langle t\rangle \frown \alpha)[s'/s]^f = \langle t[s'/s]^f\rangle \frown (\alpha[s'/s]^f)$	

Table 18, $p[s'/s]^f$ is defined. We write $m[s'/s]$, where $m \in \mathcal{M}$, for the result of replacing in m all occurrences of s by s'.

B CPO Structure for Projective Limit Model

In this appendix, we make A^∞ into a complete partial ordering (cpo) to establish the existence of least solutions of recursion equations using Tarski's fixed point theorem.

The *approximation* relation $\sqsubseteq \subseteq A_\omega \times A_\omega$ is the smallest partial ordering such that for all $p, p', q, q' \in A_\omega$:

- $\mathsf{D} \sqsubseteq p$;
- $p \sqsubseteq p' \Rightarrow \mathsf{tau} \circ p \sqsubseteq \mathsf{tau} \circ p'$;
- for all $f \in \mathcal{F}$ and $m \in \mathcal{M}$, $p \sqsubseteq p' \wedge q \sqsubseteq q' \Rightarrow p \trianglelefteq f.m \trianglerighteq q \sqsubseteq p' \trianglelefteq f.m \trianglerighteq q'$;
- for all $f \in \mathcal{F}$, $m \in \mathcal{M}$ and $s \in \mathrm{n}(m)$, $p \sqsubseteq p' \wedge q \sqsubseteq q' \Rightarrow \mathsf{local}_s^f(p \trianglelefteq f.m \trianglerighteq q) \sqsubseteq \mathsf{local}_s^f(p' \trianglelefteq f.m \trianglerighteq q')$.

The *approximation* relation $\sqsubseteq \subseteq A^\infty \times A^\infty$ is defined component-wise:

$$(p_n)_{n\in\mathbb{N}} \sqsubseteq (q_n)_{n\in\mathbb{N}} \Leftrightarrow \forall n \in \mathbb{N} \bullet p_n \sqsubseteq q_n \; .$$

The approximation relation $\sqsubseteq$ on A_n is simply the restriction of $\sqsubseteq$ on A_ω to A_n.

The following proposition states that any $p \in A_\omega$ is finitely approximated by projection.

Proposition 10. *For all $p \in A_\omega$:*

$$\exists n \in \mathbb{N} \bullet (\forall k < n \bullet \pi_k(p) \sqsubseteq \pi_{k+1}(p) \wedge \forall l \geq n \bullet \pi_l(p) = p) \; .$$

Proof. The proof follows the same line as the proof of Proposition 1 from [3]. This means that it is a rather trivial proof by induction on the structure of p. Here, we have to consider the additional case $p \equiv \mathsf{local}_s^f(p' \trianglelefteq a \trianglerighteq p'')$ with $s \in \mathrm{n}(m)$. This case goes analogous to the case $p \equiv p' \trianglelefteq a \trianglerighteq p''$. $\square$

The properties stated in the following lemma will be used in the proof of Theorem 9 given below.

Lemma 8. *For all $n \in \mathbb{N}$:*

1. *$(A_n, \sqsubseteq)$ is a cpo;*
2. *π_n is continuous;*
3. *for all $p \in A_\omega$:*
   (a) *$\pi_n(p) \sqsubseteq p$;*
   (b) *$\pi_n(\pi_n(p)) = \pi_n(p)$;*
   (c) *$\pi_{n+1}(\pi_n(p)) = \pi_n(p)$.*

Proof. The proof follows similar lines as the proof of Proposition 2 from [3]. For property 1, we now have to consider directed sets that consist of D, postconditional compositions and restrictions of postconditional compositions instead of D and postconditional compositions. However, the same reasoning applies. For property 2, we now have to use induction on the structure of the elements of A_ω and distinction between the cases $n = 0$ and $n > 0$ for postconditional compositions. Due to the presence of restrictions, we cannot use induction on n and case distinction on the structure of the elements of A_ω like in [3]. However, the crucial details of the proof remain the same. Like in [3], property 3a follows immediately from Proposition 10. Properties 3b and 3c follow immediately from Lemma 1. □

The following theorem states some basic properties of the approximation relation $\sqsubseteq$ on A^∞.

Theorem 9. *$(A^\infty, \sqsubseteq)$ is a cpo with $\bigsqcup P = (\bigsqcup\{\pi_n(p) \mid p \in P\})_{n\in\mathbb{N}}$ for all directed sets $P \subseteq A^\infty$. Moreover, up to (order) isomorphism $A_\omega \subseteq A^\infty$.*

Proof. The proof follows the same line as the proof of Theorem 1 from [3]. That is, using general properties of the projective limit construction on cpos, the first part follows immediately from Lemmas 8.1 and 8.2, and the second part follows easily from Proposition 10 and Lemma 8.3. □

Another important property of the approximation relation $\sqsubseteq$ on A^∞ is stated in the following theorem.

Theorem 10. *The operations from the projective limit model of* $\mathrm{TC}_{\mathrm{md}}$ *are continuous with respect to $\sqsubseteq$.*

Proof. The proof begins by establishing the monotonicity of the operations on A_ω. For the postconditional composition operations, this follows immediately from the definition of $\sqsubseteq$ on A_ω. For the cyclic interleaving operation, it is straightforwardly proved by induction on the sum of the depths plus one of the threads in the thread vector and case distinction on the structure of the first thread in the thread vector. For the deadlock at termination operation, the thread-service composition operations and the restriction operations, it is easily proved by structural induction. Then the monotonicity of the operations on A^∞ follows from their monotonicity on A_ω, the monotonicity of the projection operations and the definition of $\sqsubseteq$ on A^∞.

What remains to be proved is that least upper bounds of directed sets are preserved by the operations. We will show how the proof goes for the postconditional composition operations. The proofs for the other kinds of operations go similarly. Let $P, Q \subseteq A^\infty$ be directed sets. Then, for all $n \in \mathbb{N}$, $\{\pi_n(p) \mid p \in P\}$, $\{\pi_n(q) \mid q \in Q\}, \{\pi_n(p) \trianglelefteq a \trianglerighteq \pi_n(q) \mid p \in P \wedge q \in Q\} \subseteq A_n$ are directed sets by the monotonicity of π_n. Moreover, it is easily proved by induction on n, using the definition of $\sqsubseteq$ on A_n, that these directed sets are finite. This implies that they have maximal elements. From this, it follows by the monotonicity of $\_ \trianglelefteq a \trianglerighteq \_$ that, for all $n \in \mathbb{N}$, $(\bigsqcup\{\pi_n(p) \mid p \in P\}) \trianglelefteq a \trianglerighteq (\bigsqcup\{\pi_n(q) \mid q \in Q\}) = \bigsqcup\{\pi_n(p) \trianglelefteq a \trianglerighteq \pi_n(q) \mid p \in P \wedge q \in Q\}$. From this, it follows by the property of lubs of directed sets stated in Theorem 9 and the definition of π_{n+1} that, for all $n \in \mathbb{N}$, $\pi_{n+1}((\bigsqcup P) \trianglelefteq a \trianglerighteq (\bigsqcup Q)) = \pi_{n+1}(\bigsqcup\{p \trianglelefteq a \trianglerighteq q \mid p \in P \wedge q \in Q\})$. Because $\pi_0((\bigsqcup P) \trianglelefteq a \trianglerighteq (\bigsqcup Q)) = \mathsf{D} = \pi_0(\bigsqcup\{p \trianglelefteq a \trianglerighteq q \mid p \in P \wedge q \in Q\})$, also for all $n \in \mathbb{N}$, $\pi_n((\bigsqcup P) \trianglelefteq a \trianglerighteq (\bigsqcup Q)) = \pi_n(\bigsqcup\{p \trianglelefteq a \trianglerighteq q \mid p \in P \wedge q \in Q\})$. From this, it follows by the definition of $\sqsubseteq$ on A^∞ that $(\bigsqcup P) \trianglelefteq a \trianglerighteq (\bigsqcup Q) = \bigsqcup\{p \trianglelefteq a \trianglerighteq q \mid p \in P \wedge q \in Q\}$. □

We have the following result about fixed points.

Theorem 11. *Let x be a variable, and let t be a term over the signature of* $\mathrm{TC}_{\mathrm{md}}$ *in which no other variables than x have free occurrences. Then $[\![t]\!]^x$ has a least fixed point with respect to $\sqsubseteq$, i.e. there exists a $p \in A^\infty$ such that $[\![t]\!]^x(p) = p$ and, for all $q \in A^\infty$, $[\![t]\!]^x(q) = q$ implies $p \sqsubseteq q$.*

Proof. We have from Theorem 9 that $(A^\infty, \sqsubseteq)$ is a cpo and, using Theorem 10, it is easily proved by induction on the structure of t that $[\![t]\!]^x$ is continuous. From this, we conclude by Tarski's fixed point theorem that $[\![t]\!]^x$ has a least fixed point with respect to $\sqsubseteq$. □

Hence, every recursion equation in which no recursion operator occurs has a least solution in the projective limit model for $\mathrm{TC}_{\mathrm{md}}$.

According to Tarski's fixed point theorem, the least fixed point of a continuous operation $\phi : A^\infty \to A^\infty$ is $\bigsqcup\{\phi^n(\mathsf{D}) \mid n \in \mathbb{N}\}$. It is well-known that the restriction to continuous functions of the operation $\mathsf{fix_l} :(A^\infty \to A^\infty) \to A^\infty$ defined by $\mathsf{fix_l}(\phi) = \bigsqcup\{\phi^n(\mathsf{D}) \mid n \in \mathbb{N}\}$ is continuous. Moreover, for all functions $\phi{:}A^\infty \to A^\infty$ that are representable by a term over the signature of $\mathrm{TC}_{\mathrm{md}}$+REC, $\mathsf{fix}(\phi) = \mathsf{fix_l}(\phi)$. This brings us to the following corollary of Theorem 11.

Corollary 1. *Let x be a variable, and let t be a term over the signature of* $\mathrm{TC}_{\mathrm{md}}$+REC *in which no other variables than x have free occurrences. Then $[\![t]\!]^x$ has a least fixed point with respect to $\sqsubseteq$.*

Hence, every recursion equation has a least solution in the projective limit model for $\mathrm{TC}_{\mathrm{md}}$+REC.

References

1. L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. Elsevier, Amsterdam, 2001.
2. J. A. Bergstra and I. Bethke. Molecular dynamics. *Journal of Logic and Algebraic Programming*, 51:193–214, 2002.
3. J. A. Bergstra and I. Bethke. Polarized process algebra and program equivalence. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *Proceedings 30th ICALP*, volume 2719 of *Lecture Notes in Computer Science*, pages 1–21. Springer-Verlag, 2003.
4. J. A. Bergstra and J. W. Klop. Fixed point semantics in process algebra. CWI Report IW 206/82, Centre for Mathematics and Computer Science, August 1982.
5. J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1/3):109–137, 1984.
6. J. A. Bergstra and M. E. Loots. Program algebra for sequential code. *Journal of Logic and Algebraic Programming*, 51(2):125–156, 2002.
7. J. A. Bergstra and C. A. Middelburg. A thread algebra with multi-level strategic interleaving. To appear in *Theory of Computing Systems*, 2006.
8. J. A. Bergstra and C. A. Middelburg. Thread algebra for strategic interleaving. Computer Science Report 04-35, Department of Mathematics and Computer Science, Eindhoven University of Technology, November 2004.

9. J. A. Bergstra and C. A. Middelburg. Maurer computers with single-thread control. Computer Science Report 05-17, Department of Mathematics and Computer Science, Eindhoven University of Technology, June 2005.
10. J. A. Bergstra and C. A. Middelburg. Maurer computers for pipelined instruction processing. Computer Science Report 06-12, Department of Mathematics and Computer Science, Eindhoven University of Technology, March 2006.
11. J. A. Bergstra and C. A. Middelburg. Splitting bisimulations and retrospective conditions. *Information and Computation*, 204:1083–1138, 2006.
12. J. A. Bergstra and C. A. Middelburg. Thread algebra with multi-level strategies. *Fundamenta Informaticae*, 71(2/3):153–182, 2006.
13. J. A. Bergstra and A. Ponse. Combining programs and state machines. *Journal of Logic and Algebraic Programming*, 51(2):175–192, 2002.
14. A. Bolychevsky, C. R. Jesshope, and V. Muchnick. Dynamic scheduling in RISC architectures. *IEE Proceedings Computers and Digital Techniques*, 143(5):309–317, 1996.
15. F. H. Croom. *Principles of Topology*. Saunders College Publishing, Philadelphia, 1989.
16. J. W. de Bakker, J. A. Bergstra, J. W. Klop, and J.-J. Ch. Meyer. Linear time and branching time semantics for recursion with merge. *Theoretical Computer Science*, 34:135–156, 1984.
17. J. W. de Bakker and J. I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54(1/2):70–120, 1982.
18. J. Dugundji. *Topology*. Allyn and Bacon, Boston, 1966.
19. J. Gosling, B. Joy, G. Steele, and G. Bracha. *The Java Language Specification*. Addison-Wesley, Reading, MA, second edition, 2000.
20. A. Hejlsberg, S. Wiltamuth, and P. Golde. *C# Language Specification*. Addison-Wesley, Reading, MA, 2003.
21. W. A. Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1993.
22. C. R. Jesshope and B. Luo. Micro-threading: A new approach to future RISC. In *Australian Computer Architecture Conference 2000*, pages 34–41. IEEE Computer Society Press, 2000.
23. E. Kranakis. Fixed point equations with parameters in the projective model. *Information and Computation*, 75(3):264–288, 1987.
24. C. A. Middelburg. An alternative formulation of operational conservativity with binding terms. *Journal of Logic and Algebraic Programming*, 55(1/2):1–19, 2003.
25. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part II. *Information and Computation*, 100:41–77, 1992.
26. M. R. Mousavi, M. J. Gabbay, and M. A. Reniers. SOS for higher order processes. In M. Abadi and L. de Alfaro, editors, *CONCUR 2005*, volume 3653 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, 2005.
27. A. Ponse and Y. S. Usenko. Equivalence of recursive specifications in process algebra. *Information Processing Letters*, 80:59–65, 2001.
28. D. A. Schmidt. *Denotational Semantics: A Methodology for Language Development*. Allyn and Bacon, Boston, 1986.
29. V. Stoltenberg-Hansen and J. V. Tucker. Algebraic and fixed point equations over inverse limits of algebras. *Theoretical Computer Science*, 87:1–24, 1991.