Making Components Move: A Separation of Concerns Approach ⋆

Dirk Pattinson, Martin Wirsing

Institut für Informatik, LMU München

Abstract. We present a new calculus for mobile systems, the main feature of which is the separation between dynamic and topological aspects of distributed computations. Our calculus realises the following basic assumptions: (1) every computation executes in a uniquely determined location (2) processes modify the distributed structure by means of predefined operations, and (3) the underlying programming language can be changed easily. This paper introduces our calculus, and shows, that this separation of concerns leads to a perfect match between the logical, syntactical and algebraic theory. On the methodological side, we demonstrate by means of two examples, that the strict distinction between topological and computational aspects allows for an easy integration of features, which are missing in other calculi.

1 Introduction

With the success of the Internet, mobile systems have been promoted as new computational paradigm in which computation can be distributed over the net and highly dynamic, with the network itself changing continuously. In practice, however, such systems are not well accepted since users fear security problems, or more generally, the problems with controlling the behaviour of mobile systems. As a remedy, process calculi, modal logics and other formal techniques have been proposed and studied which provide theoretical foundations for mobile systems and allow one to analyse and verify properties of such systems.

The most well-known example is the π-calculus [8] of Milner which provides an abstract basis for mobility where communicating systems can dynamically change the topology of the channels. The Ambient calculus [5] of Cardelli and Gordon focuses on the handling of administrative domains where mobile processes may enter a domain or exit from a domain and in this way may change the topology of the network. Similarly, the Seal calculus [17] of Vitek and Castagna aims at describing secure mobile computations in a network that is hierarchically partitioned by localities.

In this paper we continue this line of research by proposing a new basic calculus for mobile processes called

⋆ This work has been partially sponsored by the project AGILE, IST-2001-39029.

BasicSail with focus on explicit localities and dynamic reconfiguration of networks. A configuration is a hierarchy of administrative domains, each of which is controlled by a process and which may contain other subconfigurations. Configurations may be dynamically reconfigured by entering another configuration or by exiting from a configuration. This is similar to the Ambient calculus; in contrast to other approaches we aim at a clear separation between processes and configurations: processes show behaviour, whereas the configurations provide the topological structure. BasicSail abstracts from a concrete process calculus: We aim at studying the dynamic reconfiguration of configurations independently of the underlying notion of process. Our approach is centred around three assumptions, which we now briefly discuss:

Assumption 1. Every computation takes place in a uniquely determined location.

This assumption in particular forces a two-sorted approach: We need to distinguish between elements which relate to the spatial structure and those, which drive the computation process. Since our primary interest is the study of mobile computation, we would like to be as independent as possible from the concrete realisation of processes, and therefore make

Assumption 2. The distributed part of the calculus is independent of the underlying programming language or process calculus.

However, a computation needs some means to change the distributed and spatial structure (otherwise our study would end here). That is, we need a clean mechanism, through which the distributed structure can be modified :

Assumption 3. Processes modify the distributed structure of the computation through interfaces only.

Our calculus is modelled after these assumptions. Regarding independence of the underlying programming language, we assume that the processes, which control the computations, already come with a (fixed) operational semantics, in terms of a labelled transition system; this allows us to realise interfaces as a particular set of distinguished labels. As already mentioned before, the separation between processes and locations is taken care of by using a two sorted approach.

The main technical contribution of the paper is the study of the algebraic and logical properties of the basic calculus, and of its extension with local names. We introduce the notion of spatial bisimulation and give an algebraic and a logical characterisation of the induced congruence. Our main result here is, that if one abstracts from the concrete realisation of the computations, we obtain a perfect match between structural congruence, logical equivalence and spatial congruence.

Methodologically, we want to advocate the separation between the concepts "mobility" and "computation" on a foundational basis; we try to make this point by giving two extensions of the calculus, which are missing in other calculi and can be smoothly integrated into BasicSail, thanks to the separation between spatial structure and computation.

We introduce the basic calculus, that is, the calculus without local names, in Section 2. The algebraic theory of he calculus is investigated in Section 3, and Section 4 transfers these results to a logical setting. We then extend the calculus with local names (Section 5). Further extensions, which demonstrate the versatility of our approach, are discussed in Section 6. Finally, Section 7 compares our approach to other calculi found in the literature.

2 Basic Sail: The Basic Calculus

This section introduces BasicSail, our testbed for studying mobile components. In order to ensure independence from the underlying programming language (cf. Assumption 1), BasicSail consists of two layers. The lower layer (which we assume as given) represents the programming language, which is used on the component level. The upper level represents the distributed structure, which is manipulated through programs (residing on the lower level) by means of predefined interfaces. Technically, we assume that the underlying programming language comes with a labelled transition system semantics, which manipulates the distributed structure (on the upper level) by means of a set of distinguished labels.

The distinction between processes (or programs) and the locations, in which they execute (and the structure of which they modify), forces us to work in a two-sorted environment, where we assume the programs (and their operational semantics) as given, and concentrate on the distributed structure. Our basic setup is as follows:

Notation 1. Throughout the paper, we fix a set $\mathcal{N}$ of names and the set $\mathcal{L} = \{\mathtt{in}, \mathtt{out}, \mathtt{open}\} \times \mathcal{N}$ of labels and a transition system $(\mathcal{P}, \longrightarrow)$, where $\mathcal{P}$ is a set (of processes) and $\longrightarrow \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$. We assume that $(P, \longrightarrow)$ is *image finite*, that is, for every $(P, l) \in \mathcal{P} \times \mathcal{L}$, the set $\{P' \mid P \xrightarrow{l} P'\}$ is finite.

We write $\mathtt{in}\, n$ for the pair $(\mathtt{in}, n) \in \mathcal{L}$ and similarly for $\mathtt{out}, \mathtt{open}$ and call the elements of $\mathcal{L}$ *basic labels*. The set $\mathcal{P}$ is the set of *basic processes*.

The prototypical example of transition systems, which can be used to instantiate our framework, are of course process calculi. We present one such calculus, which will also be used in later examples, next.

Example 1. Take $\mathcal{P}$ to be given as the least set according to the following

grammar:

$$\mathcal{P} \ni P, Q ::= \mathbf{0} \mid P \parallel Q \mid \alpha.P \mid !P$$

where $\alpha \in \mathcal{L}$ ranges over the basic labels. The transition relation $\longrightarrow$ is generated by the following rules

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q},$$

modulo structural congruence $\equiv$, given by the axioms $P \parallel Q \equiv Q \parallel P$, $P \parallel 0 \equiv P$, $P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R$ and $!P \equiv P \parallel !P$. For convenience, we often omit the trailing inert process and write α for $\alpha.\mathbf{0}$.

Intuitively, $\alpha.P$ is a process which can perform an α action and continue as P; the term $P \parallel Q$ represents the processes P and Q running concurrently and $!P$ represents a countable number of copies of P.

Note that we use this concrete syntax for processes just in order to illustrate our approach; the general theory is independent of the syntactical presentation and just assumes that processes form a set and come with a transition system over the set $\mathcal{L}$ of labels.

Given such a transition system $(\mathcal{P}, \longrightarrow)$, the distributed structure (which is our primary interest) is built on top of $(\mathcal{P}, \longrightarrow)$ as follows:

Definition 1. *The set $\mathcal{C}$ of basic* configurations *is the least set according to the grammar*

$$\mathcal{C} \ni A, B ::= \mathbf{0} \mid n\langle P\rangle[A] \mid A, B$$

where $P \in \mathcal{P}$ is a process and $n \in \mathcal{N}$ is a name, modulo structural congruence $\equiv$, given by the equations

$$A, B \equiv B, A \qquad A, \mathbf{0} \equiv A$$
$$A, (B, C) \equiv (A, B), C$$

We call the configuration building operator "," spatial composition.

Here, $\mathbf{0}$ is the empty configuration, $n\langle P\rangle[A]$ is a configuration with name n, which is controlled by the process P and has the subconfiguration A. Finally, A, B are two configurations, which execute concurrently. The next definition lays down the formal semantics of our calculus, which is given in terms of the reduction semantics $\longrightarrow$ of the underlying process calculus:

Definition 2. *The operational semantics of BasicSail is the relation given by the following rules the following rules*

$$\frac{P \xrightarrow{\mathtt{in}\, n} P'}{m\langle P\rangle[A], n\langle Q\rangle[B] \Rightarrow n\langle Q\rangle[m\langle P'\rangle[A], B]}$$

$$\frac{P \xrightarrow{\mathtt{out}\, n} P'}{n\langle Q\rangle[m\langle P\rangle[A], B] \Rightarrow m\langle P'\rangle[A], n\langle Q\rangle[B]}$$

$$\frac{P \xrightarrow{\mathtt{open}\, n} P'}{m\langle P\rangle[A], n\langle Q\rangle[B] \Rightarrow m\langle P'\rangle[A], B}$$

together with the congruence rules

$$\frac{A \Longrightarrow A'}{A, B \Longrightarrow A', B}$$

$$\frac{A \Longrightarrow A'}{n\langle P\rangle[A] \Longrightarrow n\langle P\rangle[A']}$$

where we do not distinguish between structurally congruent configurations. The relation $\Longrightarrow$ *is called* spatial reduction.

In the examples, we often omit the empty configuration, and write $n\langle P\rangle[]$ instead of $n\langle P\rangle[\mathbf{0}]$. Using the above definition, we can study phenomena, which arise in a distributed setting, without making a commitment to any kind of underlying language. In particular, we do not have to take internal actions of processes into account; these are assumed to be incorporated into the reduction relation $\longrightarrow$ on the level of processes.

We cannot expect to be able to embed the full ambient calculus [5] into our setting, due to the fact that in the (original) ambient calculus, there are no sorts available. However, we can nevertheless treat many examples:

Example 2. We use the set of basic processes from Example 1.

1. An agent, which has the capability to enter and exit its home location to transport clients inside can be modelled as follows: Put

$$\begin{aligned}\mathrm{agent} &= a\langle P\rangle[]\\ \mathrm{client} &= c\langle Q\rangle[]\\ \mathrm{home} &= h\langle 0\rangle[\mathrm{agent}]\end{aligned}$$

where $P = !(\mathtt{out}\, h.\mathtt{in}\, h.0)$ and $Q = \mathtt{in}\, a.\mathtt{out}\, a.0$. In the configuration $\mathrm{home}, \mathrm{client}$, we have the following chain of reductions (where $P' = \mathtt{in}\, h.0 \parallel P$ and $Q' = \mathtt{out}\, a.0$):

$$\begin{aligned}&\mathrm{home}, \mathrm{client}\\ &\quad\Longrightarrow h\langle 0\rangle[], a\langle P'\rangle[], c\langle Q\rangle[]\\ &\quad\Longrightarrow h\langle 0\rangle[], a\langle P'\rangle[c\langle Q'\rangle[]]\\ &\quad\Longrightarrow h\langle 0\rangle[a\langle P\rangle[c\langle Q'\rangle[]]\\ &\quad\Longrightarrow h\langle 0\rangle[a\langle P\rangle[], c\langle 0\rangle[]].\end{aligned}$$

This sequence of reductions shows a guarded form of entry into h: The client has to enter the mediating agent a, which then transports it into h, where the client then exits. Note that in the basic calculus, c could enter h directly, if c's controlling process were different. This can be made impossible if one adds local names, as we shall do later.

2. We model an agent, which repeatedly visits two network nodes, as follows:

$$\mathrm{agent} \equiv a\langle P\rangle[]$$

with $P = !(\mathtt{in}\, n_1.\mathtt{out}\, n_1.0) \parallel !(\mathtt{in}\, n_2.\mathtt{out}\, n_2.0)$. The activity of a once it is at either n_1 or n_2 is not modelled (but imagine a checks, whether a node has been corrupted or is otherwise non-functional). In the presence of two nodes n_1 and n_2, we have the (spatial) reductions, where we write N_1 and N_2 for the controlling processes of n_1 and n_2:

$$\begin{aligned}&n_1\langle N_1\rangle[], n_2\langle N_2\rangle[], a\langle P\rangle[]\\ \Longrightarrow &n_1\langle N_1\rangle[a\langle P_1\rangle[]], n_2\langle N_2\rangle[]\\ \Longrightarrow &n_1\langle N_1\rangle[], n_2\langle N_2\rangle[], a\langle P\rangle[]\\ \Longrightarrow &n_1\langle N_1\rangle[], n_2\langle N_2\rangle[a\langle P_2\rangle[]]\\ \Longrightarrow &\cdots\end{aligned}$$

In the above, we have abbreviated $P_1 = \mathtt{out}\, n_1.0 \parallel P$ and $P_2 = \mathtt{out}\, n_2.0 \parallel P$. Here, the program P controlling a does not force a to visit n_1 and n_2 in any particular order, and a could for example choose to enter and leave n_1 continuously, without ever setting foot into n_2.

3 Algebraic Theory of the Basic Calculus

This section is devoted to the algebraic theory of the basic calculus; extensions of the calculus, in particular with local names, are deferred until Section 5. In this section, we show that the algebraic and the logical theory of the basic calculus fit together seamlessly. In more detail, we discuss the relationship between three relations on processes: spatial bisimulation (which we introduce shortly), the induced spatial congruence and structural congruence.

3.1 Basic Definitions and Examples

Spatial bisimulation will defined as binary relation on configurations, subject to some closure properties; the precise meaning of which is given as follows:

Terminology 2. Suppose $R \subseteq A \times A$ is a binary relation on a set A and $S \subseteq A \times \cdots \times A$ is $n+1$-ary. We say that R is *closed* under S, if, whenever $(a, b) \in R$ and $(a, a_1, \ldots, a_n) \in S$, there are $b_1, \ldots, b_n \in A$ with $(b, b_1, \ldots, b_n) \in S$ and $(a_i, b_i) \in R$ for $i = 1, \ldots, n$.

If R is closed under S, it is often helpful to think of R as an equivalence on processes and of S as a reduction relation. In this setting, R is closed under S if, whenever a and b are equivalent (i.e. $(a, b) \in R$) and a reduces to a' (i.e. $(a, a') \in S$), there is some b' such that a' and b' are again equivalent (i.e. $(a', b') \in R$) and b reduces to b' (that is, $(b, b') \in R$). So if R is closed under S, we think of R as being some bisimulation relation and R the corresponding notion of reduction.

Definition 3 (Spatial Bisimulation). *Consider the following endorelations on $\mathcal{C}$:*

1. Subtree reduction $\downarrow \subseteq \mathcal{C} \times \mathcal{C}$, *where* $C \downarrow D$ *iff* $C \equiv n\langle P\rangle[D]$ *for some* $n \in \mathcal{N}$ *and* $P \in \mathcal{P}$

2. Forest reduction $\circlearrowright \subseteq \mathcal{C} \times \mathcal{C} \times \mathcal{C}$ *where* $C \circlearrowright (A, B)$ *iff* $C \equiv A, B$ *and* A *is of the form* $A \equiv n\langle P\rangle[D]$ *for some* $n \in \mathcal{N}, P \in \mathcal{P}$ *and* $D \in \mathcal{C}$.

3. Top-level names $@n \subseteq \mathcal{C}$, *where* $n \in \mathcal{N}$ *and* $C \in @n$ *iff* $C \equiv n\langle P\rangle[A]$ *for some* $P \in \mathcal{P}$ *and* $A \in \mathcal{C}$.

The largest relation $\simeq \subseteq \mathcal{C} \times \mathcal{C}$, *which is closed under spatial reduction* $\Longrightarrow$, *subtree reduction* $\downarrow$, *forest reduction* $\circlearrowright$ *and top-level names* $@n$, *for all* $n \in \mathcal{N}$, *is called* spatial bisimulation.

Furthermore, spatial congruence $\cong$ *is the largest spatial bisimulation, which is a congruence with respect to construction of configurations.*

Note that, in the previous definition, we just require the congruence

property wrt. the construction of configurations, that is we require

1. $A_0 \cong A_1,\ B_0 \cong B_1 \implies A_0, A_1 \cong B_0, B_1$ and
2. $A \cong B, n \in \mathcal{N}, P \in \mathcal{P} \implies n\langle P\rangle[A] \cong n\langle P\rangle[B]$.

This not only justifies the name spatial congruence – it furthermore allows us to study the evolution of the tree structure of (a set of) mobile processes without reference to the underlying process calculus. Note that the spatial congruence is not the largest congruence contained in the spatial bisimulation (corresponding to closure under contexts). Our notion of spatial congruence follows the approach of dynamic bisimulation [9].

In a nutshell, two configurations are spatially bisimilar, if they have bisimilar reducts, bisimilar subtrees, and the same top-level names. If two configurations are spatially congruent, one can furthermore substitute them for one another, obtaining spatially congruent processes. Although spatial bisimulation is a very strong notion of bisimilarity, it is not a congruence:

Example 3. Take $n, m \in \mathcal{N}$ with $n \neq m$ and let $A \equiv n\langle \mathtt{in}\, m.0\rangle[]$ and $B \equiv n\langle 0\rangle[]$. Then $A \simeq B$ (since neither A nor B can perform a spatial reduction), but $A \not\cong B$, since $A, m\langle 0\rangle[]$ does reduce, whereas $B, m\langle 0\rangle[]$ does not.

Since we clearly want equivalent configurations to be substitutable for one another (which allows us to build large systems in a compositional way), spatial congruence is the notion of equivalence we are interested in. By definition, spatial congruence involves the closure under all configuration constructing operators, and is therefore not easy to verify.

Our first goal is therefore an alternative characterisation of spatial congruence. As it turns out, we only need to add one closure property to the definition of spatial bisimulation in order to obtain spatial congruence.

3.2 Spatial Congruence and Spatial Bisimulation

We start on our first characterisation of spatial congruence. The approach is as follows: We consider labelled reduction, introduced in the next definition, and show that (i) spatial congruence is closed under labelled reduction, and (ii) that spatial bisimulation + labelled reduction is a congruence. This immediately entails that spatial congruence is spatial bisimulation plus closure under labelled reductions. We begin with the definition of labelled reduction:

Definition 4. *Let $l \in \mathcal{L}$. Define the relation $\overset{l}{\Longrightarrow} \subseteq \mathcal{C} \times \mathcal{C}$ by the rules*

$$\frac{P \xrightarrow{l} P'}{n\langle P\rangle[A] \overset{l}{\Longrightarrow} n\langle P'\rangle[A]}$$

$$\frac{C \overset{l}{\Longrightarrow} C'}{C, D \overset{l}{\Longrightarrow} C', D}$$

and call a relation $B \subseteq \mathcal{C} \times \mathcal{C}$ closed under labelled reduction*, if B is closed under $\overset{l}{\Longrightarrow}$ for all $l \in \mathcal{L}$.*

We use the name "labelled bisimulation" for the closure of spatial bisimulation under labelled reductions.

Definition 5. *We take* labelled bisimulation *to be the largest symmetric relation* $\leftrightarrows \subseteq \mathcal{C} \times \mathcal{C}$ *which is closed under forest reduction, spatial reduction, subtree reduction, labelled reduction and top level names.*

In order to be able to compare spatial congruence and labelled bisimulation, we need a proof principle, which allows us to reason about labelled bisimulation using induction on reductions. This principle works for finitely branching systems only, and is the content of the following two lemmas:

Lemma 1. *Suppose* $(\mathcal{P}, \longrightarrow)$ *is finitely branching. Then the relations* $\Longrightarrow$, $\circlearrowright$, $\downarrow$ *and* $\stackrel{l}{\Longrightarrow}$ *(for all* $l \in \mathcal{L}$*) are image finite.*

Proof. By structural induction using the respective definitions.

Proposition 2. *Assume that* $(\mathcal{P}, \longrightarrow)$ *is image finite and define a sequence of relations* $\sim_i \subseteq \mathcal{C} \times \mathcal{C}$ *inductively as follows:*

1. $\sim_0 = \mathcal{C} \times \mathcal{C}$
2. $C \sim_{i+1} D$ *is the largest symmetric relation s.t.*
   - $C \in @n$ *implies* $D \in @n$
   - $(C, C') \in R$ *implies* $\exists D'.(D, D') \in R$ *and* $C' \sim_i D'$ *where* R *is one of* $\Longrightarrow$ *or* $\downarrow$
   - $C \circlearrowright (C_1 C_2)$ *implies* $\exists D_1, D_2.D \circlearrowright (D_1, D_2)$ *and* $C_1 \sim_i D_1$, $C_2 \sim_i D_2$
   - $C \stackrel{l}{\Longrightarrow} C'$ *implies* $\exists D'.D \stackrel{l}{\Longrightarrow} D'$ *and* $C' \sim_{i+1} D'$ *for* $l \in \mathcal{L}$

Then, if C *and* $D \in \mathcal{C}$*, we have* $C \leftrightarrows D$ *iff* $C \sim_i D$ *for all* $i \in \mathbb{N}$.

Proof. We abbreviate $\sim = \bigcap_{i \in \mathbb{N}} \sim_i$. In order to see that $C \leftrightarrows D$ whenever $C \sim_i D$, one shows that $\sim$ is a spatial bisimulation, which is closed under labelled reduction.

The converse follows from the fact that all relations used in the definition of $\sim_i$ are image finite (Lemma 1).

We note two easy consequences of the above characterisation: in particular, controlling processes, which are bisimilar (in the ordinary sense) do not destroy the relations $\sim_i$ and therefore preserve labelled bisimulation. That is, if we call the largest symmetric relation $B \subseteq \mathcal{P} \times \mathcal{P}$, which is a (strong) labelled bisimulation in the ordinary sense a *process bisimulation*, we have the following:

Lemma 3. *1.* $\sim_{i+1} \subseteq \sim_i$ *for all* $i \in \mathbb{N}$.

2. Let $n \in \mathcal{N}$, $A, B \in \mathcal{C}$ *and* $P, Q \in Prop$*. Then for all* $i \in \mathbb{N}$ $n\langle P \rangle[A] \sim_{i+1} n\langle Q \rangle[B]$ *iff* P, Q *are process-bisimilar and* $A \sim_i B$.

The relationship between labelled bisimulation and process bisimulation can be formalised as follows:

Corollary 4. *Let $n \in \mathcal{N}$, $A, B \in \mathcal{C}$ and $P, Q \in Prop$. Then $n\langle P\rangle[A]$ and $n\langle Q\rangle[B]$ are labelled bisimilar iff P, Q are process-bisimilar and A and B are labelled bisimilar.*

We are now ready to tackle the first step of our comparison between labelled bisimulation and spatial congruence.

Lemma 5. *Spatial congruence is closed under labelled reduction.*

Proof. Suppose $n \in \mathcal{N}$, $C, D \in \mathcal{C}$ are spatially congruent and $C \stackrel{l}{\Longrightarrow} C''$. Then C is of the form $C \equiv C_0, C_1$ with $C_0 \equiv m\langle P\rangle[E]$ and $P \stackrel{l}{\longrightarrow} P'$ for some $P' \in \mathcal{P}$ and $E \in \mathcal{C}$. We proceed by case distinction on $l \in \mathcal{L}$, where we use a fresh name $k \in \mathcal{N}$, i.e. k does not occur as the name of a location either in C or in D, and some arbitrary $R \in \mathcal{P}$.

Case $l = \mathtt{in}\, n$: Consider the context $K[\_] = n\langle R\rangle[k\langle R\rangle[]], \_$. Then $K[C] \Longrightarrow C'$ with $C' \equiv C_1, n\langle R\rangle[m\langle P'\rangle[E], k\langle R\rangle[]]$. Since $C \cong D$, we have $K[D] \Longrightarrow D'$ with $C' \cong D'$. Since spatial congruence is closed under forest reduction and top-level names, we can split $D' \equiv D_1, n\langle R'\rangle[F]$ for some $R' \in \mathcal{P}$ and $F \in \mathcal{C}$, where $D_1 \cong C_1$ and $n\langle R'\rangle[F] \cong n\langle R\rangle[m\langle P'\rangle[E], k\langle R\rangle[]]$. Using closure under subtree reduction, we obtain $F \cong m\langle Q'\rangle[E'], k\langle R\rangle[]$ (since k is fresh) with $m\langle Q'\rangle[E'] \cong m\langle P'\rangle[E]$. Again using that k is fresh, we have $D \equiv D_1, m\langle Q\rangle[E']$ for some $Q \in \mathcal{P}$ with $Q \stackrel{\mathtt{in}\, n}{\longrightarrow} Q'$ with $D_1 \cong C_1$ and $m\langle P'\rangle[E] \cong m\langle Q'\rangle[E']$; since spatial congruence is a congruence we finally obtain $D \stackrel{\mathtt{in}\, n}{\Longrightarrow} D_1, m\langle Q'\rangle[E'] \cong C_1, m\langle P'\rangle[E]$.

Case $l = \mathtt{out}\, n$: Similar, using the context $n\langle R\rangle[\_, k\langle R\rangle[]]$.

Case $l = \mathtt{open}\, n$: Similar, using the context $n\langle R\rangle[k\langle R\rangle[]], \_$.

The converse of Lemma 5 needs the proof principle of Proposition 2.

Lemma 6. *Labelled bisimulation is a congruence.*

Proof. We have to show that labelled bisimulation is a congruence wrt. the construction of configurations, that is, wrt. "putting in a box" and spatial composition.

Congruence wrt. spatial composition: We show that the relation $R_i = \{(C, E), (D, E) | C, D, E \in \mathcal{C} \text{ and } C \sim_i D\}$ is a subset of $\sim_i$ for all $i \in \mathbb{N}$. The case $i = 0$ is trivial; for the inductive step we show that any pair $((C, E), (D, E)) \in R_{i+1}$ satisfies the properties defined in Prop. 2 with $\sim_{i+1}$ replaced by R_i. The cases of top level names, forest reductions and labelled reductions follow directly from the definitions of the R_{i+1} and the fact that $\sim_{i+1} \subseteq \sim_i$. For spatial reduction suppose $C, E \Longrightarrow C'$. If either $C \Longrightarrow C_0$ and $C' \equiv C_0, E$ or $E \Longrightarrow E_0$ and $C' \equiv C, E_0$ the result follows easily from the induction hypothesis. For all other cases we have to show that C, E and D, E have the same spatial reductions, resulting in configurations, which are $\sim_i$ equivalent.

We only consider the `in`-rule; the other cases are similar.

If $C, E \Longrightarrow C'$ by virtue of the `in`-rule, either a component of C enters into a component of E, or vice versa. That is, we have one of the following two cases:

1. $C \equiv C_0, C_1$ with $C_0 \equiv m\langle P\rangle[F]$ and $P \xrightarrow{\texttt{in}\, n} P'$ and $E \equiv E_0, E_1$ with $E_0 \equiv n\langle Q\rangle[G]$, or
2. $E \equiv E_0, E_1$ with $E_0 \equiv m\langle P\rangle[F]$ and $P \xrightarrow{\texttt{in}\, n} P'$ and $C \equiv C_0, C_1$ with $C_0 \equiv n\langle Q\rangle[G]$.

We only treat the first case; the second can be treated along similar lines (using Lemma 3) . From the assumption $C \sim_{i+1} D$ we obtain (using forest reduction and preservation of top level names), that we can split $D \equiv D_0, D_1$ with $D_0 \equiv m\langle R\rangle[H]$ and $C_j \sim_i D_j$ for $j = 0, 1$. Using closure under labelled reduction, we have $R \xrightarrow{\texttt{in}\, n} R'$ with $m\langle P'\rangle[F] \sim_i m\langle R'\rangle[H]$. Since $C, E \Longrightarrow C'$ we obtain $C' \equiv n\langle Q\rangle[m\langle P'\rangle[F], G], C_1, E_1$ and $D, E \Longrightarrow D'$ with $D' \equiv n\langle Q\rangle[m\langle R'\rangle[H], G], D_1, E_1$, from which we obtain $C' \sim_i D'$ using that $\sim_i$ is a congruence.

Congruence wrt. putting in a box: Suppose $C, D \in \mathcal{C}$ with $C \sim_{i+1} D$ and $n \in \mathcal{N}$, $P \in \mathcal{P}$. We have to show that $n\langle P\rangle[C] \simeq_{i+1} n\langle P\rangle[D]$. As before, the only interesting cases arise through spatial reductions. So suppose $n\langle P\rangle[C] \Longrightarrow C'$. If this is because $C \Longrightarrow C''$ and $C' \equiv n\langle P\rangle[C'']$, we find $D'' \sim_i C''$ with $D \Longrightarrow D''$, since $C \sim_{i+1} D$. In this case $n\langle P\rangle[D] \Longrightarrow D'$ with $D' \equiv n\langle P\rangle[D'']$ and by ind. hyp. $C' \sim_i D'$.

Now assume $n\langle P\rangle[C] \Longrightarrow C'$ using the `out`-rule. That is $C \equiv C_0, C_1$ with C_1 of the form $C_1 \equiv m\langle Q\rangle[E]$ and $Q \xrightarrow{\texttt{out}\, n} Q'$. With $C_0' \equiv m\langle Q'\rangle[E]$ we thus have $C_0 \xRightarrow{\texttt{out}\, n} C_0'$. Using forest reduction, we can split $D \equiv D_0, D_1$ with $D_j \sim_i C_j$ for $j = 0, 1$. In particular, $D_0 \xRightarrow{\texttt{out}\, n} D_0'$ and $D_0' \sim_i C_0'$. By assumption, we have $C' \equiv C_0', n\langle P\rangle[C_1]$. Putting $D' \equiv D_0', n\langle P\rangle[D_1]$, we obtain $n\langle P\rangle[D] \Longrightarrow D'$ and $D' \sim_i C'$.

From the previous lemma, we obtain the desired characterisation of spatial congruence:

Corollary 7. *Spatial congruence and labelled bisimulation coincide.*

Proof. By Lemma 5, spatial congruence is contained in spatial bisimulation. Lemma 6 proves the other inclusion.

This result is our first characterisation of spatial congruence in the basic calculus. Spatial congruence allows us to observe the dynamic behaviour of controlling processes plus the tree structure of configurations. One therefore suspects, that spatial congruence is a very intensional notion of equivalence. In the following, we show that spatial congruence is very intensional indeed, by comparing it to the relation of structural congruence on configurations.

3.3 Spatial Congruence vs Structural Congruence

Depending on the underlying labelled transition system $(\mathcal{P}, \longrightarrow)$, which controls the behaviour of processes (which in turn control the evolution of configurations), it is clear that structural congruence is strictly contained in spatial congruence: If $P, Q \in \mathcal{P}$ are bisimilar but not identical, we have that $n\langle P\rangle[]$ and $n\langle Q\rangle[]$ are not structurally congruent, but spatially congruent. This example relies on the existence of equivalent, but non-identical processes in $\mathcal{P}$. In this section, we show, that this is indeed the only possible way in which we can have configurations, which are spatially congruent, but not structurally congruent. We now proceed to show that spatial congruence coincides with structural congruence modulo process bisimilarity. We start with the following:

Definition 6. Weak structural congruence *is the least relation R generated by the rules of Definition 1, plus the rule*

$$\frac{C \equiv D \qquad P, Q \text{ process bisimilar}}{n\langle P\rangle[A] \equiv n\langle Q\rangle[B]}$$

where $n \in \mathcal{N}$, $A, B \in \mathcal{C}$ and $P, Q \in \mathcal{P}$.

Thus weak structural congruence not only identifies structurally congruent configurations, but also configurations with bisimilar controlling processes. We think of weak structural congruence as structural congruence up to process bisimilarity.

Note that – coming back to the example at the beginning of the section – that $n\langle P\rangle[A]$ and $n\langle Q\rangle[A]$ are weakly congruent for P, Q process bisimilar. We have argued that this is an example of a pair of configurations, which are spatially congruent, but not structurally congruent. Extending structural congruence to include those configurations, which only differ in the controlling process, structural and spatial congruence can be shown to coincide:

Proposition 8. *Weak structural congruence and spatial congruence coincide.*

Proof. It follows directly from the definitions that weak structural congruence (which we denote by $\equiv$ for the purpose of this proof) is contained in spatial congruence. We prove the converse inclusion by contradiction: assume that the set $\mathcal{F} = \{(C, D) \in \mathcal{C} \mid C \cong D, C \not\equiv D\}$ of felons is non empty. For $C \in \mathcal{C}$, we define the *height* of C, $\mathrm{ht}(C)$, by induction as follows: $\mathrm{ht}(\mathbf{0}) = 0, \mathrm{ht}(C, D) = \mathrm{ht}(C) + \mathrm{ht}(D)$, $\mathrm{ht}(n\langle P\rangle[C']) = 1 + \mathrm{ht}(C')$.

Since the standard ordering on natural numbers is a well-ordering, there is a pair (C, D) of felons, such that $\mathrm{ht}(C)$ is minimal, that is, for all $(C', D') \in \mathcal{F}$ we have $\mathrm{ht}(C') \geq \mathrm{ht}(C)$. We discuss the different possibilities for C.

Case $C \equiv C_0, C_1$ with $C_0 \not\equiv \mathbf{0} \not\equiv C_1$: Using forest reduction, we can split $D \equiv D_0, D_1$ with $D_j \cong C_j$ for $j = 0, 1$. Since $\mathrm{ht}(C_0) < \mathrm{ht}(C)$ and $\mathrm{ht}(C_1) < \mathrm{ht}(C)$, neither (C_0, D_0) nor

(C_1, D_1) are felons, that is, $C_0 \equiv D_0$ and $C_1 \equiv D_1$, hence $C \equiv C_0, C_1 \equiv D_0, D_1 \equiv D$, contradicting $(C, D) \in \mathcal{F}$.

Case $C \equiv n\langle P\rangle[C_0]$*:* By subtree reduction, $D \equiv m\langle Q\rangle[D_0]$ with $C_0 \cong D_0$. Since $\mathrm{ht}(C_0) < \mathrm{ht}(C)$, the pair (C_0, D_0) is not a felon, hence $C_0 \equiv D_0$.

By closure under top-level names, furthermore $n = m$, and closure under labelled reduction implies that P and Q are process bisimilar. Hence $n\langle P\rangle[C_0]$ and $m\langle Q\rangle[D_0]$ are weakly congruent, contradicting $(C, D) \in \mathcal{F}$.

Case $C \equiv \mathbf{0}$*:* From $C \cong D$ we conclude $D \equiv \mathbf{0}$, contradicting $C \not\equiv D$.

This concludes our investigation of the algebraic properties of BasicSail, which we summarise as follows:

Theorem 9. *Suppose* $C, D \in \mathcal{C}$*. The following are equivalent:*

1. *C and D are spatially congruent*
2. *C and D are labelled bisimilar*
3. *C and D are weakly structurally congruent*

4 The logical theory of BasicSail

In the previous section, we have looked at spatial congruence from an algebraic viewpoint and have given three different characterisations. This section adopts a logical view and gives a further characterisation of spatial bisimulation in terms of a (modal style) logic. Using our setup from the previous section, this task is not overly difficult, we just have to make the (standard) assumption that the underlying processes are finitely branching. Making this assumption, we obtain a logic, which is completely standard except for one binary modal operator, which plays a role similar to the linear implication used in [4,2], except for the fact that linear implication in *loc. cit.* is the logical version of parallel composition, whereas the modal operator we are about to introduce, is the logical dual to "extending a parallel composition with one more process".

As before, our definitions and results are parametric in a set $\mathcal{N}$ of names and the associated set $\mathcal{L}$ of labels (cf. Notation 1). We begin with introducing spatial logic. In essence, this definition is modelled after the characterisation given in Corollary 7.

Definition 7. *The language* L *of* spatial logic *is the least set of formulas according to the grammar*

$$\mathsf{L} \ni \phi, \psi ::= \epsilon \mid @n \mid ff \mid \phi \to \psi \mid \langle R\rangle\phi \mid \langle \circlearrowright\rangle\phi\psi$$

where $n \in \mathcal{N}$*,* $l \in \mathcal{L} \cup \{\tau\}$ *and* R *ranges over the relations* $\downarrow, \Longrightarrow$ *and* $\overset{l}{\Longrightarrow}$ *for* $l \in \mathcal{L}$*.*

Intuitively, the formula ϵ allows us to speak about the empty context and $@n$ allows us to observe the names of locations. Formulas of type $\langle R\rangle\phi$ allow us (as in standard modal logic) to

reason about the behaviour of a process after evolving according to the relation R. In our case, we can specify properties of sub-configurations (using $\downarrow$), transitions (using $\Longrightarrow$) and labelled reductions (using $\overset{l}{\Longrightarrow}$). The most interesting formula is of type $\langle \circlearrowleft \rangle \phi\psi$: it asserts that we can split a process into a single node satisfying ϕ and a remainder, satisfying ψ.

Definition 8. *The semantics of propositional connectives is as usual. For the modal operators, we put, for $C \in \mathcal{C}$:*

$$
\begin{array}{ll}
C \models \epsilon & \textit{iff } C \equiv \mathbf{0} \\
C \models @n & \textit{iff } C \in @n \\
C \models \langle R \rangle \phi & \textit{iff } \exists C'.(C, C') \in R \\
 & \quad \textit{and } C' \models \phi \\
C \models \langle \circlearrowleft \rangle \phi\psi & \textit{iff } \exists C', C''.C \circlearrowleft (C', C'') \\
 & \quad \textit{and } C' \models \phi, C'' \models \psi
\end{array}
$$

where R is as above. As usual, $\mathrm{Th}(C) = \{\phi \in \mathsf{L} \mid C \models \phi\}$ denotes the logical theory of $C \in \mathcal{C}$. Two configurations C, D are logically equivalent, *if $\mathrm{Th}(C) = \mathrm{Th}(D)$.*

Note that we use the expression "$@n$" above both as an atomic formula of the logic and as a unary relation. In this section, we show that logical equivalence gives yet another characterisation of spatial congruence, provided the underlying set of processes is finitely branching. This follows from the characterisation of spatial congruence as spatial bisimulation + labelled reduction by appealing to Proposition 2. We then obtain a characterisation of spatial congruence in the sense of Hennessy and Milner [7].

The main result of this section is as follows:

Theorem 10. *Suppose $(\mathcal{P}, \longrightarrow)$ is image finite. Then spatial congruence and logical equivalence coincide.*

Proof. We use the characterisation of spatial congruence as labelled bisimulation and Proposition 2. It follows directly from the definition of spatial logic, that formulas of spatial logic cannot distinguish states, which are labelled bisimilar, hence labelled bisimilarity is contained in logical equivalence. For the converse, we use the method of Hennessy and Milner [7] and a variant of Proposition 2, replacing "$i + 1$" by "i" in the last clause of the assumption (the meticulous reader is invited to check that the Proposition remains valid).

Suppose for a contradiction that there is a pair of configurations $(C, D) \in \mathcal{C} \times \mathcal{C}$ such that C and D are logically equivalent, but not labelled bisimilar. Let i be minimal such with the property that $C \not\sim_i D$ but $C \sim_k D$ for all $k < i$ (such an n exists because of Proposition 2).

Since C and D are not labelled bisimilar, we have – up to symmetry – one of the following cases:

1. $C \in @m$ but $D \notin @m$ for some $m \in \mathcal{N}$. Then $C \models @m$ but $D \not\models @m$, contradicting $\mathrm{Th}(C) = \mathrm{Th}(D)$.

2. There is $C' \in \mathcal{C}$ such that $(P, P') \in R$ but there is no $D' \in \mathcal{C}$

with $(D, D') \in R$ and $C' \sim_{i-1} D'$, where R is one of $\downarrow, \Longrightarrow$ or $\stackrel{l}{\Longrightarrow}$ (for $l \in \mathcal{L}$).
Since i is minimal, this means that for all D' with $(D, D') \in R$ there is a formula $\phi_{D'}$ such that $D' \not\models \phi_{D'}$ but $C' \models \phi_{D'}$. Take $\phi = \bigwedge_{D':(D,D')\in R} \langle R \rangle \phi_{D'}$, which is well defined by Lemma 1. Then $C \models \phi$ but $D \not\models \phi$, contradicting $\mathrm{Th}(C) = \mathrm{Th}(D)$.

3. There are C_0, C_1 with $C \not\leftrightarrows (C_0, C_1)$ but there is no $(D_0, D_1) \in \mathcal{C} \times \mathcal{C}$ with $D_j \sim_{i-1} C_j$ $(j = 0, 1)$ and $D \not\leftrightarrows (D_0, D_1)$. The argument is as above, using formulas of the form $\langle \circlearrowleft \rangle \phi \, \psi$.

Summing up, we have shown that

Spatial congruence = spatial bisimulation + labelled reduction = structural congruence up to process bisimilarity = logical equivalence

Before extending these correspondences to a more general setting, we give some examples.

Example 4. We use the same setup as in Example 2.

1. Consider the configuration $C \equiv \mathrm{home}, \mathrm{client}$ from Example 2. We have $C \models \langle \circlearrowleft \rangle)(@\mathrm{home}, tt)$, corresponding to the statement that there is a top level node with the name "home".
Also, $C \models \langle \circlearrowleft \rangle(\langle \downarrow \rangle @\mathrm{agent}, tt)$, which expresses that C has a subtree, one node of which has the name "agent".

2. Consider the configuration $C \equiv n_1\langle P \rangle [], n_2 \langle Q \rangle []$, similar to Example 2. Here, $C \models \langle \circlearrowleft \rangle (@n_1, tt)$, i.e. there is a location in C with the name "n_1". Also, $C \models \langle \circlearrowleft \rangle (@n_1, (\langle \circlearrowleft \rangle (@n_2, \epsilon)))$, which says that all top level processes contained in C have either the name n_1 or n_2.

5 Local Names

In the calculus of mobile ambients, local names are essential for many examples. The treatment of local names is derived from the π-calculus, i.e. governed by structural rule of scope extrusion $(\nu n P) \mid Q \equiv \nu n (P \mid Q)$ whenever n is not a freely occurring name of Q. In the ambient calculus, local names cut across dynamics and spatial structure, by adopting a second structural rule: $\nu n (k[P]) \equiv k[\nu n P]$ if $n \neq k$, which allows to move the restriction operator up and down the tree structure, induced by the nesting of the ambient brackets.

If we want to remain independent from the underlying process calculus, we cannot adopt the latter rule. However, we can look at a calculus with local names, where local names obey scope extrusion a la π-calculus.

The next definition extends the syntax as to incorporate local names. In order to deal with scope extrusion, we also have to introduce the concept of free names.

Definition 9. *The set $\mathcal{C}$ of configurations in the calculus with local names*

is given by

$$\mathcal{C} \ni C, D ::= \mathbf{0} \mid n\langle P\rangle[C] \mid C, D \mid (\nu n)C$$

for $n \in \mathcal{N}$ and $P \in \mathcal{P}$. Given $P \in \mathcal{P}$ and $n \in \mathcal{N}$, we say that n is free *in P, if there are $l_1, \ldots, l_k$ and $P_1, \ldots, P_k$ such that $P \xrightarrow{l_1} P_1 \xrightarrow{l_2} \cdots \xrightarrow{l_k} P_k \xrightarrow{l} Q$, where l is one of $\mathtt{in}\, n$, $\mathtt{out}\, n$ and $\mathtt{open}\, n$. We let $\mathrm{fn}(P) = \{n \in \mathcal{N} \mid n$ free in $P\}$.*

For $C \in \mathcal{C}$, the set $\mathrm{fn}(C)$ is defined by induction on the structure of C as follows:

- $\mathrm{fn}(\epsilon) = \emptyset$
- $\mathrm{fn}(C, D) = \mathrm{fn}(C) \cup \mathrm{fn}(D)$
- $\mathrm{fn}(n\langle P\rangle[C]) = \{n\} \cup \mathrm{fn}(P) \cup \mathrm{fn}(C)$
- $\mathrm{fn}(\nu n C) = \mathrm{fn}(C) \setminus \{n\}$

where structural congruence is as in Definition 1, augmented with α-equivalence and the rule $(\nu n)(A, B) \equiv (\nu n A), B$ whenever n does not occur freely in B.

The operational semantics *is given as in Definition 1, augmented with the rule*

$$\frac{C \Longrightarrow C'}{(\nu n)C \Longrightarrow (\nu n)C'}$$

for $C, C' \in \mathcal{C}$ and $n \in \mathcal{N}$.

Note that, in order to be able to state the rule for α-equivalence, we need a notion of substitution on the underlying processes, which we do not make explicit here.

Before investigating the logical and algebraic theory of the calculus with local names, we give a short example. Recall that in Example 2, we had an agent in a home location, the sole purpose of which was to transport clients inside the home-location. However, as we remarked when discussing this example, nothing prevents the client process to enter the home-location directly. This shortcoming can now be remedied in the calculus with local names.

Example 5. We can now model an agent, which has the capability to enter and exit its home location and to transport clients inside with local names as follows: We let "client" and "agent" as in Example 2 and put

$$\mathrm{home} = (\nu h)h\langle 0\rangle[\mathrm{agent}]$$

Using scope extrusion, we have the same chain of reductions as in Example 2. However, since h is a private name now, the client cannot enter "home" without the help of "agent".

The next issue we are going to discuss is the algebraic and the logical theory of the calculus with local names. In order to obtain a similar characterisation as in the calculus without local names, we have to extend the definition of spatial bisimulation, and demand closure under name revelations.

Definition 10. *Suppose $C \in \mathcal{C}$ and $n, k \in \mathcal{N}$. We put*

$$C \overset{\mathtt{rev}\, n}{\Longrightarrow} C' \quad \textit{iff} \quad C \equiv (\nu k)C'' \textit{ and } C' \equiv C[n/k]$$

whenever $n \notin \mathrm{fn}(C)$. *The definition of spatial bisimulation is modified as follows: Spatial bisimulation is the largest symmetric relation which is closed under spatial reduction* $\Longrightarrow$*, forest reduction* $\circlearrowright$*, subtree reduction* $\downarrow$*, top level names* $@n$ *and under revelation* $\stackrel{\mathtt{rev}\, n}{\Longrightarrow}$ *(for all* $n \in \mathcal{N}$*).*

As before, spatial congruence is the largest congruence, which is a spatial bisimulation.

We now turn to the impact of local names on the equivalences, which we have discussed previously. Since we make revelation an explicit part of spatial bisimulation, everything goes through as before, once the equivalences are transferred (without changes) to the calculus with local names. We obtain:

- *labelled bisimulation* is the largest spatial bisimulation, which is closed under labelled reduction
- *weak structural congruence* is the least relation, which contains structural congruence and all pairs of the form $(n\langle P\rangle[C], n\langle Q\rangle[C])$ for $P, Q \in \mathcal{P}$ process bisimilar.

Comparing these equivalences, we obtain

Theorem 11. *In the calculus with local names, spatial congruence coincides with labelled bisimulation and with weak structural congruence.*

Proof. We extend the respective results for the calculus without local names. The arguments used in Lemma 5 remain valid, showing that spatial congruence is closed under labelled reduction, implying that spatial congruence is contained in labelled bisimilarity.

In order to see that labelled bisimulation is a congruence, one has to consider revelation reductions, that is, reductions of the form $\stackrel{\mathtt{rev}\, n}{\Longrightarrow}$ on top of the reductions considered in Lemma 6, but they do not pose any problems.

The comparison of spatial congruence and weak structural congruence is as in Proposition 8.

In order to transfer the characterisation result to a logical setting, we introduce a hidden name quantifier a la Gabbay / Pitts [6]:

Definition 11. *The language of* spatial logic with local names *is the least set according to the following grammar*

$$\mathsf{L} \ni \phi, \psi ::= \epsilon \mid @n \mid ff \mid \phi \to \psi \mid \langle R\rangle\phi \mid \langle\circlearrowright\rangle\phi\psi \mid \mathsf{H}n.\phi$$

Given $C \in \mathcal{C}$ *and* $\phi \in \mathsf{L}$*, satisfaction* $C \models \phi$ *is as in Definition 7, plus the clause*

$$C \models \mathsf{H}n.\phi \text{ iff } C \stackrel{\mathtt{rev}\, n}{\Longrightarrow} C' \text{ and } C' \models \phi$$

for the hidden name quantifier. As before, $\mathrm{Th}(C) = \{\phi \in \mathsf{L} \mid C \models \phi\}$ *for* $C \in \mathcal{C}$*, and* $C, D \in \mathcal{C}$ *are called* logically equivalent*, if* $\mathrm{Th}(C) = \mathrm{Th}(D)$.

Since the relation $\mathtt{rev}\, n$ (for $n \in \mathcal{N}$) is image-finite, Lemma 1 and Proposition 2 remain valid in the calculus with local names. We thus obtain

Theorem 12. *In the calculus with local names, spatial congruence and logical equivalence coincide.*

6 Further Extensions

This section shows, that the separation of dynamic and spatial aspects of mobile components allows for seamless integration of extensions, which are more difficult to model in other calculi. First, we demonstrate that multiple names can easily be handled, since every process runs in precisely one location. It is therefore a straightforward extension to allow the controlling process to change the name of that location. The second extension can be seen as orthogonal: Since the behaviour of every location is governed by precisely one process, new controlling processes can easily be substituted into configurations. This section intends to give an idea regarding extensions of the Basic-Sail calculus; we leave the investigation of the algebraic and logical theory for further work.

6.1 Change of Names and Multiple Names

In the ambient calculus, each ambient has precisely one name, which does not change throughout the reduction process. One can argue that this does not reflect the real world in a faithful manner, since computing devices can also have no names, multiple names or change their names over time. The explicit reference to the enclosing location allows to model the change of names elegantly in the Sail-calculus by extending the set of labels, which influence the spatial reduction relation.

Since we want to keep the separation of the dynamical from the spatial structure, we let the controlling processes change the names of locations through an interface (a set of distinguished labels) as before. This necessitates to extend the set of labels for the underlying process calculus:

Convention 3. We extend the set $\mathcal{L}$ of labels to include primitives for name changing as follows: $\mathcal{L} = \{\texttt{in}, \texttt{out}, \texttt{open}, \texttt{up}, \texttt{down}\} \times \mathcal{N}$; as before, $(\mathcal{P}, \longrightarrow)$ is a labelled transition system with $\longrightarrow \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$.

Definition 12. *In the calculus with multiple names, configurations are given by*

$$\mathcal{C} \ni A, B ::= \mathbf{0} \mid A, B \mid \nu n A \mid (n_1, \ldots, n_k)\langle P \rangle[B]$$

where $k \in \mathbb{N}$ and $n_1, \ldots, n_k \in \mathcal{N}$.

The axioms and rules of structural congruence *are those of Definition 9 augmented with*

$$(n_1, \ldots, n_k)\langle P \rangle[B] \equiv (n_{\sigma(1)}, \ldots, n_{\sigma(k)})\langle P \rangle[B]$$

$$(n, n, n_1, \ldots, n_k)\langle P \rangle[B] \equiv (n, n_1, \ldots, n_k)\langle P \rangle[B]$$

whenever $\sigma : \{1, \ldots, k\} \to \{1, \ldots, k\}$ is a bijection.

The operational semantics *is that of Definition 9, augmented with the*

rules

$$\frac{P \xrightarrow{\mathtt{up}\, n} P'}{\mathfrak{n}\langle P\rangle[A] \longrightarrow \mathfrak{n}^{+n}(P')[A]}$$

$$\frac{P \xrightarrow{\mathtt{down}\, n} P'}{\mathfrak{n}\langle P\rangle[A] \longrightarrow \mathfrak{n}^{-n}(P')[A]}$$

where $\mathfrak{n}^{-n}$ *deletes* n *from the list* $\mathfrak{n}$ *of names;* $\mathfrak{n}^{+n}$ *adds* n *to the list* $\mathfrak{n}$ *of names.*

The idea of a term $(n, m)\langle P\rangle[A]$ is that of a location with two names, n and m, running the programme P and which has A as sub-locations. The additional rule of structural congruence captures the fact that there is no order on the names. The gained expressivity allows us to treat the following:

Example 6. 1. Anonymous locations are modelled by an empty set of names. Take for example $()\langle P\rangle[A]$ for $P \in \mathcal{P}$ and $A \in \mathcal{C}$. Note that anonymous locations are anonymous also for processes from within, that is, the same effect cannot be achieved using local names. Indeed, the processes $\nu n(n)\langle P\rangle[k\langle \mathtt{out}\, n\rangle[]]$ and $()\langle P\rangle[k\langle \mathtt{out}\, n\rangle[]]$ differ in that the former can perform a reduction under the name binder, whereas the latter cannot.

2. Consider the configuration $(n)\langle \mathtt{down}\, n.\mathbf{0}\rangle[A], ()\langle \mathtt{in}\, n.\mathbf{0}\rangle[B]$. First, this shows that unnamed locations can perform movements. Second, this example illustrates that the movement only succeeds, if the unnamed agent is lucky enough to enter into his partner *before* the name disappears.

6.2 Dynamic Reconfiguration

We conclude by demonstrating the strength of our approach by discussing dynamic reconfiguration, another extension of the basic calculus. Here, we use the one-to-one relation between locations and controlling processes to model dynamic reconfiguration, i.e. locations, which dynamically change the programme they run. Sloppily speaking, this allows for downloading a new programme, which is then run in an already existing location. As with multiple names and the change of names, the explicit reference to the enclosing location allows for a concise and elegant formulation of dynamic reconfiguration. Note that this in particular necessitates the transmission of programmes (processes). The extension of the calculus follows the same scheme as the above extension with multiple names: in order to keep dynamic and spatial structure apart, we introduce new labels, which act as an interface, through which the controlling process manipulates the spatial structure.

Convention 4. We extend the set $\mathcal{L}$ of labels to include primitives for dynamic reconfiguration as follows: $\mathcal{L} = \{\mathtt{in}, \mathtt{out}, \mathtt{open}\} \times \mathcal{N} \cup \{\mathtt{send}, \mathtt{rec}, \mathtt{run}\} \times \mathcal{P}$; as before, $(\mathcal{P}, \longrightarrow)$ is a labelled transition system with $\longrightarrow \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$.

Note that this requires the underlying transition system to have processes occurring in the labels, since processes need to be transmitted. Except for the

absence of channel names, this is for example realised in the higher order π-calculus (see [13,16]). For our purposes, it suffices that processes can be transmitted and received; we leave the concrete (syntactical) mechanism abstract.

Definition 13. *In the calculus with dynamic reconfiguration, configurations are given as in the calculus with local names (but recall the extended set of labels). The* operational semantics *for the calculus with dynamic reconfiguration is given by the rules of Definition 9, augmented with*

$$\frac{P \xrightarrow{\mathtt{send}\,R} P' \quad Q \xrightarrow{\mathtt{rec}\,R} Q'}{n\langle P\rangle[C],m\langle Q\rangle[D] \Rightarrow n\langle P'\rangle[C],m\langle Q'\rangle[D]}$$

$$\frac{P \xrightarrow{\mathtt{run}\,R} P'}{n\langle P\rangle[C] \Longrightarrow n\langle R\rangle[C]}$$

Note that in the action $\mathtt{run}\,R$*,* R *is a process and the reduct of* P *after the* $\mathtt{run}\,R$ *reduction is forgotten.*

Using dynamic reconfiguration and communication, we can now model a location, which updates the process it executes:

Example 7. We model an electronic device, which attempts to update the code it is running (its operating system). That is, it tries to replace the programme which it is running by another (newer) version. In order to model this behaviour, we first have to be more precise about the underlying set of processes: We let

$$P, Q \ni \mathcal{P} ::= \mathbf{0} \mid P \parallel Q \mid \alpha.P \mid !P \mid X \mid \mathtt{run}\,Q.P \mid \mathtt{send}\,Q.P \mid \mathtt{rec}\,Q.P$$

where $X \in \mathfrak{X}$ ranges over a set of (process valued) variables. The process level transition relation from Example 1 is augmented with

$$\mathtt{rec}\,X.P \xrightarrow{\mathtt{rec}\,Q} P[Q/X]$$

and the usual rules

$$\mathtt{send}\,Q.P \xrightarrow{\mathtt{send}\,Q} P$$

$$\mathtt{run}\,Q.P \xrightarrow{\mathtt{run}\,Q} P$$

Note that in particular process variables $X \in \mathfrak{X}$ do not generate reductions. Now consider

$$P = (\mathtt{rec}\,X.\mathtt{run}\,X) \parallel O$$

running inside location n, that is, the configuration $C = n\langle P\rangle[B]$, where B are n's sub-locations. In the vicinity of a location which sends updates, e.g.$U = u\langle!(\mathtt{send}\,(\mathtt{rec}\,X.\mathtt{run}\,X \parallel N))\rangle[]$, where N stands for the "new" firmware, we have

$$U, C \Longrightarrow U, n\langle \mathtt{run}\,(\mathtt{rec}\,X.\mathtt{run}\,X \parallel N) \parallel O\rangle[B]$$

which, executing the $\mathtt{run}$-operation, reduces to

$$U, n\langle \mathtt{rec}\,X.\mathtt{run}\,X \parallel N\rangle[B],$$

that is, a process which (again) waits for an update, but now running the new firmware N.

As already mentioned in the introductory remark of this section, both extensions, multiple names and dynamic reconfiguration, are to demonstrate the extensibility of the calculus; the study of the algebraic and logical properties is left for further research.

7 Conclusions and Related Work

As discussed above the first calculus for mobile computation was the π-calculus [8]. Further calculi are the Fusion calculus [12], Nomadic Pict [18] and the distributed coordination language KLAIM [10]. The study of hierarchical re-configurable administrative domains was introduced by the Ambient [5] and the Seal calculus [17]. BasicSail follows these lines but distinguishes processes and configurations in an a priori way and concentrates on a even simpler set of operations for reconfiguration.

The basic calculus and its variations were inspired by the Seal-Calculus. [17]. However, the Seal-Calculus is quite involved syntactically; the present calculus is a simplification in order to study the effect of the separation of dynamics from the underlying topological structure, which is also present in Seal. The second source of inspiration was the calculus of mobile ambients [5]. As we have pointed out before, our principal design decisions do not allow to embed the full ambient calculus into our framework. Spatial logics were studied by Cardelli and Caires [2,3], although to our knowledge not wrt. a clear characterisation of the expressive power. Such a characterisation (called "intensional bisimulation") was considered by Sangiorgi for a variant of the ambient calculus [14,15].

Separation of Concerns in Models of software architecture has also been addressed – albeit not in the context of mobile code – in [1,11]. There the authors differentiate between components, which provide certain services, and an additional layer, which describes the composition of components. In the context of explicit code mobility, this approach can be seen as orthogonal to ours; and it would certainly be interesting to have coordination and mobility in a single framework.

Of course, there remains a wealth of open problems: Most pressingly, we have investigated neither the logical nor the algebraic theory of the calculus with multiple names or the calculus with reconfiguration.

References

1. F. Arbab. Abstract behaviour types: A foundation model for components and their composition. This Volume.
2. L. Caires and L. Cardelli. A spatial logic for concurrency (part i). In N. Kobayashi and B. Pierce, editors, *Proc. TACS 2001*, volume 2215 of *Lecture Notes in Computer Science*, pages 1–37. Springer, 2001.
3. L. Caires and L. Cardelli. A spatial logic for concurrency (part i). In L. Brim, P. Jancˋar, M. Kˋretìnskỳ, and A. Kucˋera, editors, *Proc. CONCUR 2002*, volume

2421 of *Lecture Notes in Computer Science*. Springer, 2002.

4. L. Cardelli and A. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. POPL 2000*, pages 365–377. ACM, 2000.
5. L. Cardelli and A. Gordon. Mobile ambients. *Theor. Comp. Sci.*, 240(1):177–213, 2000.
6. D. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In *14th IEEE Symposium on Logic in Computer Science (LICS 1999)*, pages 214–224. IEEE Computer Society, 1999.
7. M. Hennessy and R. Milner. Algebraic Laws for Non-determinism and Concurrency. *Journal of the ACM*, 32:137–161, 1985.
8. R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.
9. U. Montanari and V. Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae*, 16(2):171–199, 1992.
10. R. De Nicola, G. Ferrari, and R. Pugliese. Klaim: a kernel language for agents interaction and mobility. *IEEE Trans. Software Engineering*, 24(5):315–330, 1998.
11. O. Nierstrasz and F. Achermann. A calculus for modelling software components. This Volume.
12. J. Parrow and B. Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Thirteenth Annual Symposium on Logic in Computer Science (LICS 1998)*, pages 176–185. IEEE, IEEE Computer Society, 1998.
13. D. Sangiorgi. From π-calculus to Higher-Order π-calculus — and back. In M.-C. Gaudel and J.-P. Jouannaud, editors, *Proc. TAPSOFT 93*, volume 668 of *Lect. Notes in Comp. Sci.*, pages 151–166, 1993.
14. D. Sangiorgi. Extensionality and intensionality of the ambient logics. In *Proc. POPL 2001*, pages 4–13. ACM, 2001.
15. D. Sangiorgi. Separability, expressiveness, and decidability in the ambient logic. In *17th IEEE Symposium on Logic in Computer Science (LICS 2002)*. IEEE Computer Society, 2002.
16. Davide Sangiorgi and David Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
17. J. Vitek and G. Castagna. Seal: A framework for secure mobile computation. *Internet Programming*, 1999.
18. P. Wojciechowski and P. Sewell. Nomadic pict: Language and infrastructure design for mobile agents. *IEEE Concurrency*, 8(2):42–52, 2000.