A Coordination Approach to Mobile Components ⋆

Dirk Pattinson, Martin Wirsing

Institut für Informatik, LMU München, Germany

Abstract

We present a calculus for mobile systems, the main novel 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, (3) the underlying programming language can be changed easily, and (4) locations are hierarchically organised. This paper introduces our calculus, and shows, that this separation of concerns leads to a perfect match between the logical, syntactical and algebraic theory. We discuss a core calculus, and extensions with local names and with multiple names.

Key words: mobile components, coordination languages
PACS:

Introduction

With the success of the Internet, mobile computation has presented itself as a new computing paradigm. Over the Internet, distributed computation can be highly dynamic, with a network, which is hierarchically organised into administrative domains and constantly changing.

In practice, distributed systems pose many challenges: users fear security problems or, more generally, problems with controlling the behaviour of mobile systems. The designer of a system has to integrate many different platforms and programming languages.

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

Process calculi and associated formal logics have been studied to deal with problems of the first kind, whereas coordination languages have been proven successful for integrating different computing platforms and languages.

The present paper studies a calculus which allows to address both issues in a uniform framework: we investigate a coordination model for mobile systems, which explicitely allows to model the hierarchical structure of network locations. We study the syntactic, logical and algebraic theory of the calculus and obtain a perfect match between the corresponding equivalences. This shows, that our framework allows to add mobility to software components in a transparent way.

On the side of process calculi, the ambient calculus [16] was the first framework which directly allowed to represent the hierarchical structure of locations. This calculus has been extended in many ways and directions to accommodate different computational mechanism; we mention secure ambients [24], where the communication is synchronous, boxed ambients [10], which adopts a different communication scheme (parents communicate with children), which is similar to the scheme employed by the Seal calculus [44].

On a programming and more practical level, coordination languages [32] have been used to provide the glue needed to build distributed applications from a set of stand alone components. Several such languages have been implemented and are being used in real-world applications [23,3]. On a foundational level, the language KLAIM [29] has been investigated as language where localities are first class citizens, and can be manipulated by a KLAIM programme. The common feature found in all coordination approaches of distributed computing is the *separation of concerns* between the distributed aspects and the computation which are carried out at the individual nodes.

In this paper, we present and study a framework, which allows for direct modelling of hierarchically structured locations while providing a basic separation of concerns between the dynamic and topological aspects of computations in the style of coordination models. We consider the hierarchical structure of locations to be an essential ingredient in any approach to mobile computation: the Internet provides the glue between different sites, which themselves are divided into a hierarchical collection of subnets, each of which with their own administrative policy.

From a coordination point of view, this is the main novelty of our approach: apart from standard coordination principles, we assume that *localities have a hierarchical structure:* Our basic model provides the glue between concurrent computations, each of which performed in distinct locations, which can communicate and actively change the topological structure.

In more detail, we consider locations, where each location has a name, a con-

trolling process and a (possibly empty) set of sub-locations. We abstract from the concrete realisation of the controlling processes by not assuming a particular calculus or language, and instead assume the controlling processes to be given in form of a labelled transition system. The processes interact with their environment via a set of designated labels, which allow them to change the topological structure of the locations to which they are attached.

The hierarchical structure of locations is similar in spirit 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. The separation between processes and locations is also present in KLAIM [29], with the main difference that locations in KLAIM are not nested.

In view of applications of mobility to programming technology, this seems most realistic: keeping the mobility primitives separate from the programming languages which are using them allows for much greater re-use and increases portability of code. The introduction of this additional abstraction layer as provides a programming framework for mobile applications. Existing frameworks typically implement this idea. Our concept of location is very similar to that of a place in Mole [4] and to Jade's containers [6,5]. The assumptions which lead to our particular model are guided by the coordination approach to distributed systems, which we extend by postulating that locations have a hierarchical structure. We briefly discuss our postulates below.

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. That is, we need a clean mechanism, through which the distributed structure can be manipulated:

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

Finally, in order to model hierarchically structured administrative domains, or sub-networks with different administrative or security policies, we postulate

that locations are structured; this is the main novelty of our approach, which is not present in other coordination approaches to mobile systems.

Assumption 4 *Locations are hierarchically structured, that is, each location has a finite (possibly empty) set of sub-locations.*

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. In particular, this enables us to work with strong process equivalences only, since assume that computation steps that do not affect the topological structure are already taken care of on the level of processes.

The main technical contribution of the paper is the study of the algebraic and logical properties of the basic calculus, and its extension with local names and multiple 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 take our results as an indication that our framework transparently allows to incorporate component mobility. Technically, we use closure properties to define bisimulations and bisimulation congruences. Our equivalences account for the spatial structure of computations, and we compare the algebraic, logical and syntactical theory of our calculus. Both in the basic calculus and its extensions, we discuss which observations allow to capture spatial bisimulation congruence: In the basic calculus, we need to observe the transitions of the processes controlling the component movement. Extending the calculus with local names, we have to add name revelation as an observation while a calculus with multiple names requires to observe behaviour after name changes.

Structure of the paper: we introduce the basic calculus, that is, the calculus without local names, in Section 1. The algebraic theory of he calculus is investigated in Section 2 and Section 3, and Section 4 transfers these results to a logical setting. We then extend the calculus with local names (Section 5). We discuss one more extension, the possibility for a location to have multiple names, in Section 6. Finally, Section 7 compares our approach to other calculi found in the literature.

1 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 2), 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 pre-defined 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.

This is similar to the semantics of the coordination language MANIFOLD [8]; the main difference is that our controlling processes already come with a labelled transition system semantics, which allows us to concentrate on the distributed structure. Our basic setup is as follows:

Notation Unless stated otherwise, 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 $\emptyset \neq \mathcal{P}$ 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 call two processes P, Q *process bisimilar*, if they are bisimilar as elements of the labelled transition system $(\mathcal{P}, \longrightarrow)$.

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. Note that we consider only transitions with mobility primitives as labels, that is, we assume that internal transitions are already accounted for in the semantics of processes. One such process calculus, which we use in our example, is the following:

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 a 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.

Although we have included the replication operator in the core calculus above, it is still image finite, since we consider processes only up to structural congruence. 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 2 *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. We consider configurations up to structural equivalence* $\equiv$*, given by the equations*

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

The configuration building operator "," is called spatial composition*, and we refer to* $\langle P \rangle[\,\cdot\,]$ *as* tree construction.

In the above, $\mathbf{0}$ 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 3 *The operational semantics of BasicSail is the relation given by the reaction rules*

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

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

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

together with the congruence rules

$$\frac{A \Longrightarrow A'}{A, B \Longrightarrow A', B} \qquad \frac{A \Longrightarrow A'}{n\langle P\rangle[A] \Longrightarrow n\langle P\rangle[A']} \qquad \frac{A \equiv A' \;\; A' \Longrightarrow B' \;\; B' \equiv B}{A \Longrightarrow B.}$$

The relation $\Longrightarrow$ *is called* spatial reduction.

The last rule captures that we do not distinguish between structurally congruent configurations. 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 [16] into our setting, since we have to distinguish between the computational and the distributed components, whereas the ambient calculus follows an untyped approach. However, we can nevertheless treat many examples:

Example 4 *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

$$\text{agent} = a\langle P\rangle[] \quad \text{client} = c\langle Q\rangle[] \quad \text{home} = h\langle 0\rangle[\text{agent}]$$

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

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

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 will see later.*

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

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

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:

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

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.

2 Spatial Bisimulation and Spatial Congruence

This section introduces spatial bisimulation and spatial congruence, the basic equivalences we will be concerned with for the remainder of the paper. The equivalences are introduced for the basic calculus. We discuss an extension of the equivalences to a calculus with local names in Section 5 and to a calculus which allows multiple names in Section 6.

We introduce spatial bisimulation as binary relation on configurations, subject to a set of closure properties. We take "closure property" as formal term, the meaning of which is given as follows:

Terminology 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, $(b, b') \in S$). So if R is closed under S, we think of R as being some bisimulation relation and S the corresponding notion of reduction.

Definition 5 (Spatial Bisimulation, Spatial Congruence) *Consider the following relations on $\mathcal{C}$:*

(1) Subtree reduction $\downarrow \subseteq \mathcal{C} \times \mathcal{C}$*, given by* $C \downarrow D$ *if* $\exists E \in \mathcal{C}, n \in \mathcal{N}, P \in \mathcal{P}. C \equiv n\langle P \rangle [D], E$.
(2) Forest reduction $\circlearrowleft \subseteq \mathcal{C} \times \mathcal{C} \times \mathcal{C}$*, given by* $C \circlearrowleft (D, E)$ *if* $C \equiv D, E$.
(3) Top-level names $@n \subseteq C$*, given by* $C \in @n$ *if* $\exists D, E \in \mathcal{C}, P \in \mathcal{P}. C \equiv n\langle P \rangle [D], E$.
(4) The inert relation $\mathbf{0} \subseteq C$*, where* $C \in \mathbf{0}$ *if* $C \equiv \mathbf{0}$.

We take spatial bisimulation $\simeq$ *(resp. spatial congruence* $\cong$*) to be the largest symmetric (resp. congruence) relation, which is closed under spatial reduction and the relation (1 - 4).*

Note that forest reduction is the ternary relation associated to spatial composition in the sense of [31,35]. For spatial congruence, we just require the congruence property w.r.t. 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 (this relation is not a bisimulation in general). Our notion of spatial congruence follows the approach of dynamic congruence[28] to ensure that the resulting equivalence retains the bisimulation property.

Remark Our definition of spatial congruence is given in two steps: first we define spatial bisimulation, and then consider the largest congruence, which is a spatial bisimulation at the same time. The original paper [28] suggests the following different definition: spatial congruence is the largest relation, which is *context closed* under the relations (1) - (4) in Definition 5. Here, we call a binary relation $R \subseteq \mathcal{C} \times \mathcal{C}$ context closed under a $n+1$-ary relation $S \subseteq \mathcal{C} \times \cdots \times \mathcal{C}$, if for all contexts $C[\cdot]$ and all $(a, b) \in R$, whenever $(C[a], a_1, \ldots, a_n) \in S$, there are $(b_1, \ldots, b_n) \in \mathcal{C}^n$ with $(C[b], b_1, \ldots, b_n) \in S$. One can obtain the same results using this definition; we prefer to work with Definition 5 as it directly highlights the important features of spatial congruence: being a bisimulation and a congruence at the same time.

In a nutshell, two configurations are spatially bisimilar, if they have bisimilar reducts, bisimilar subtrees, and the same top-level names. Spatial congruence is taken to be the largest spatial bisimulation, which is a congruence w.r.t. spatial composition and tree construction.

These closure properties already allow us to distinguish processes, which consist of two ore more components running concurrently, from processes which

have a single top level location.

Definition 6 *A configuration $C \in \mathcal{C}$ is called a* singleton, *if $C \equiv n\langle P\rangle[D]$ for some $n \in \mathcal{N}, P \in \mathcal{P}$ and $D \in \mathcal{C}$. This is denoted by* $\mathrm{st}(C)$.

In the next lemma, we show that spatial bisimulation is already strong enough to distinguish singleton configurations from configurations which are not.

Lemma 7 *Suppose $C, D \in \mathcal{C}$ with $C \simeq D$ and* $\mathrm{st}(C)$. *Then* $\mathrm{st}(D)$.

Proof: Suppose not. Then either $D \equiv \mathbf{0}$ (in which case $C \equiv \mathbf{0}$, contradiction) or $D \equiv D_1, D_2$ with both $D_1, D_2 \not\equiv \mathbf{0}$. Then $D \circlearrowleft (D_1, D_2)$, and since $C \simeq D$, $C \equiv C_1, C_2$ with both $C_1, C_2 \not\equiv \mathbf{0}$, contradiction. □

If two configurations are spatially congruent, they can be substituted for one another, yielding again spatially congruent configurations. The next example shows, that spatial congruence is properly contained in spatial bisimulation.

Example 8 *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 $A, m\langle 0\rangle[]$ can reduce, whereas $B, m\langle 0\rangle[]$ cannot.*

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 Labelled Bisimulation

In this section, we describe the relation between spatial congruence and labelled bisimulation, a notion which we will introduce shortly.

This addresses the problem of checking that two configurations are spatially bisimilar – the definition of spatial congruence requires closure under all contexts. This can be avoided if one considers labelled bisimulation, which is an extension of spatial bisimulation with respect to one more closure property. In this section we take the first step towards a characterisation of spatial congruence by showing that labelled bisimulation is contained in spatial congruence.

Definition 9 *Let $l \in \mathcal{L}$. Define the relation $\stackrel{l}{\Longrightarrow} \subseteq \mathcal{C} \times \mathcal{C}$ by the rules*

$$\frac{P \stackrel{l}{\longrightarrow} P'}{n\langle P\rangle[A] \stackrel{l}{\Longrightarrow} n\langle P'\rangle[A]} \qquad \frac{C \stackrel{l}{\Longrightarrow} C'}{C, D \stackrel{l}{\Longrightarrow} C', D}$$

and call a relation $B \subseteq \mathcal{C} \times \mathcal{C}$ closed under labelled reduction*, if B is closed under $\stackrel{l}{\Longrightarrow}$ for all $l \in \mathcal{L}$. We take* labelled bisimulation $\leftrightarroweq$ *to be the largest symmetric relation, which is closed under labelled reduction, spatial reduction and the relations (1 - 4) of Definition 5.*

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 image finite relations only:

Lemma 10 *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 using the fact that $(\mathcal{P}, \rightarrow)$ is image finite. □

The last lemma puts us into the position to use induction on the number of (labelled) reduction steps as a proof principle. To make our reasoning explicit, we use a sequence of relations $\sim_i$, each of which capturing the behaviour up to and including i labelled reduction steps.

Definition 11 *Define a sequence of relations $\sim_i \subseteq \mathcal{C} \times \mathcal{C}$ inductively as follows:*

(1) $\sim_0$ is the largest symmetric relation such that for all $C \sim_0 D$
- $\mathrm{st}(C)$ *implies* $\mathrm{st}(D)$.
- $C \in @n$ *implies* $D \in @n$ *for all* $n \in \mathcal{N}$.
- $C \stackrel{l}{\Longrightarrow} C'$ *implies* $\exists D'.D \stackrel{l}{\Longrightarrow} D'$ *and* $C' \sim_0 D'$ *for all* $l \in \mathcal{L}$.

(2) $C \sim_{i+1} D$ is the largest symmetric relation s.t. for all $C \sim_{i+1} D$
- $\mathrm{st}(C)$ *implies* $\mathrm{st}(D)$
- $C \in @n$ *implies* $D \in @n$
- $(C, C') \in R$ *implies* $\exists D'.(D, D') \in R$ *and* $C' \sim_i D'$ *for* $R \in \{\Longrightarrow, \downarrow\}$.
- $C \circlearrowright (C_1 C_2)$ *implies* $\exists D_1, D_2.D \circlearrowright (D_1, D_2)$ *and* $C_j \sim_i D_j$, $j = 1, 2$.
- $C \stackrel{l}{\Longrightarrow} C'$ *implies* $\exists D'.D \stackrel{l}{\Longrightarrow} D'$ *and* $C' \sim_{i+1} D'$ *for* $l \in \mathcal{L}$

Note that in the above definition, the relations $\sim_i$ are required to be closed under labelled reduction; this is expressed in the last clause, where we do not refer to the previously defined relation.The proof principle, which we use to show that labelled bisimulation is a congruence, can now be formulated as follows:

Proposition 12 *(1) $\sim_{i+1} \subseteq \sim_i$ for all $i \in \mathbb{N}$.*

(2) For all $C, D \in \mathcal{C}$, $C \leftrightarrows D$ iff $C \sim_i D$ for all $i \in \mathbb{N}$.

Proof: The first claim is immediate from the definition. For the second, we abbreviate $\sim = \bigcap_{i\in\mathbb{N}} \sim_i$.

To see that $\sim \subseteq \leftrightarrows$ it suffices to show that $\sim$ is a labelled bisimulation. We only treat closure under forest reduction, the remaining cases are even easier. So suppose that $C \sim D$ and $C \circlearrowright (C_1, C_2)$. Since $C \sim D$, there are, for all $i \in \mathbb{N}$, D_1^i and $D_2^i \in \mathcal{C}$ such that $D \circlearrowright (D_1^i, D_2^i)$ and $D_j^i \sim C_j^i$ for $j = 1, 2$. Since $\circlearrowright$ is image finite and $\sim_{i+1} \subseteq \sim_i$ for all i, we can find (D_1, D_2) with $D \circlearrowright (D_1, D_2)$ and $D_j \sim C_j$ for $j = 1, 2$.

The converse inclusion $\leftrightarrows \subseteq \bigcap_{i\in\mathbb{N}} \sim_i$ follows from $\leftrightarrows \subseteq \sim_i$ for all $i \in \mathbb{N}$, which is readily established using induction on i and the fact $\sim_{i+1} \subseteq \sim_i$. Lemma 7 establishes the clause dealing with singleton configurations. □

The next lemma is needed to compare labelled bisimulation and spatial congruence.

Lemma 13 *Suppose $P, Q \in \mathcal{P}$, $C, D \in \mathcal{C}$, $n \in \mathcal{N}$ and $\sim_i$ is defined as Proposition 12.*

(1) If $n\langle P\rangle[C] \sim_i n\langle Q\rangle[D]$ for some $i \in \mathbb{N}$, then P and Q are process bisimilar.
(2) If P and Q are process bisimilar, then $n\langle P\rangle[C] \sim_i n\langle Q\rangle[C]$ for all $i \in \mathbb{N}$.

The last lemma allows us to consider the processes of the underlying labelled transition system $(\mathcal{P}, \longrightarrow)$ up to process bisimilarity. With this, plus the proof principle established in Proposition 12, we can now show that labelled bisimulation is a congruence; this in particular implies that labelled bisimulation is contained in spatial congruence, which establishes a first relationship between labelled bisimulation and spatial congruence. section.

Proposition 14 *Labelled bisimulation is contained in spatial congruence.*

Proof: We show that each $\sim_i$ is a congruence; the claim then follows from Proposition 12 and the definition of spatial congruence. The case $i = 0$ is easy, so suppose $0 < i$. Note that we have to establish the congruence property w.r.t. spatial composition and tree construction

(1) Congruence w.r.t. spatial composition: Suppose $C, D \in \mathcal{C}$ with $C \sim_i D$ and $E \in \mathcal{C}$. We show that $C, E \sim_i D, E$. It is easy to see that C, E have the same labelled reductions, top level names and subtrees. We only show that they have compatible spatial reductions.

Assume $C, E \Longrightarrow C'$. We have to show that $D, E \Longrightarrow D'$ with $C' \sim_{i-1} D'$. We

distinguish the different cases corresponding to the different reduction rules. Throughout, we assume $C \equiv n\langle P\rangle[C_0], C_1$ and $E \equiv m\langle Q\rangle[E_0], E_1$. The cases where $C \Longrightarrow \tilde{C}$ and $C' \equiv \tilde{C}, E$ or $E \Longrightarrow \tilde{E}$ and $C' \equiv C, \tilde{E}$ are trivial, hence omitted.

Case 1: C enters E. Formally $C, E \Longrightarrow C'$ where $P \xrightarrow{\mathtt{in}\, m} P'$ and $C' \equiv C_1, m\langle Q\rangle[n\langle P'\rangle[C_0], E_0], E_1$.

Then $C \circlearrowright (n\langle P\rangle[C_0], C_1)$. Hence $D \circlearrowright (n\langle R\rangle[D_0], D_1)$ with $n\langle R\rangle[D_0] \sim_{i-1} n\langle P\rangle[C_0]$ and $C_1 \sim_{i-1} D_1$. Note that $X \sim_j Y$ only if both are singletons or both are not singletons for all $j \in \mathbb{N}$. Since $n\langle R\rangle[D_0] \sim_{i-1} n\langle P\rangle[C_0]$ and $n\langle P\rangle[C_0] \stackrel{l}{\Longrightarrow} n\langle P'\rangle[C_0]$, there is R' with $n\langle R\rangle[D_0] \stackrel{\mathtt{in}\, m}{\Longrightarrow} n\langle R'\rangle[D_0]$ and $n\langle P'\rangle[C_0] \sim_{i-1} n\langle R'\rangle[D_0]$. By the operational semantics of the basic calculus, we have that $D \longrightarrow D'$ for $D' \equiv D_1, m\langle Q\rangle[n\langle R'\rangle[D_0], E_0], E_1$. Since $\sim_{i-1}$ is a congruence, finally $C' \sim_{i-1} D'$ by Lemma 13.

Case 2: E enters C. Formally $C, E \Longrightarrow C'$ where $Q \xrightarrow{\mathtt{in}\, n} Q'$ and $C' \equiv n\langle P\rangle[C_0, m\langle Q'\rangle[E_0]], E_1$: Similar.

Case 3: C opens E. Formally $C, E \Longrightarrow C'$ and $C' \equiv n\langle P'\rangle[C_0], C_1, E_1$ with $P \xrightarrow{\mathtt{open}\, m} P'$. As above, $D \circlearrowright (n\langle R\rangle[D_0], D_1)$ with $n\langle R\rangle[D_0] \sim_{i-1} n\langle P\rangle[C_0]$ and $D_1 \sim_{i-1} C_1$. Since $n\langle P\rangle[C_0] \stackrel{\mathtt{open}\, m}{\Longrightarrow} n\langle P'\rangle[C_0]$, there is R' such that $n\langle R\rangle[D_0] \stackrel{\mathtt{open}\, m}{\Longrightarrow} n\langle R'\rangle[D_0]$ and $n\langle P'\rangle[C_0] \sim_{i-1} n\langle R'\rangle[D_0]$. By the operational semantics, $D, E \Longrightarrow D'$ with $D' \equiv n\langle R'\rangle[D_0], D_1, E_1$. We have $D' \sim_{i-1} C'$ by Lemma 13, since $\sim_{i-1}$ is a congruence.

Case 4: E opens C. Formally $C, E \Longrightarrow C'$ and $C' \equiv C_1, m\langle Q'\rangle[E_0], E_1$ with $Q \xrightarrow{\mathtt{open}\, n} Q'$: Similar.

The remaining cases, where the reduction has been triggered by the out-rule, are trivial.

(2) Congruence w.r.t. tree construction: Suppose $C \sim_i D$; we show that $k\langle S\rangle[C] \sim_i k\langle S\rangle[D]$ for arbitrary $k \in \mathcal{N}$ and $S \in \mathcal{P}$.

Again, it is straightforward to verify all clauses in the definition of $\sim_i$ save the clause concerning spatial reduction. We treat the following cases:

Case 1: C reduces. Formally $k\langle S\rangle[C] \Longrightarrow k\langle S\rangle[C']$. Then $C \Longrightarrow C'$, hence $D \Longrightarrow D'$ for some $D' \sim_{i-1} C'$. Then $n\langle S\rangle[D] \Longrightarrow D'$ for $D' \equiv k\langle S\rangle[D']$ and $k\langle S\rangle[C'] \sim_{i-1} k\langle S\rangle[D']$ since $\sim_{i-1}$ is a congruence.

Case 2: C leaves k. Formally, for $C \equiv n\langle P\rangle[C_0], C_1$ we have $k\langle S\rangle[C] \Longrightarrow C'$ with $C' \equiv n\langle P'\rangle[C_0], k\langle S\rangle[C_1]$ and $P \xrightarrow{\mathtt{out}\, k} P'$. Then $C \circlearrowright (n\langle P\rangle[C_0], C_1)$, hence $D \circlearrowright (n\langle R\rangle[D_0], D_1)$ with $n\langle P\rangle[C_0] \sim_{i-1} n\langle R\rangle[D_0]$ and $C_1 \sim_{i-1} D_1$.

Hence $n\langle R\rangle[D_0] \stackrel{\mathtt{out}\,k}{\Longrightarrow} n\langle R'\rangle[D_0]$ and $n\langle P'\rangle[C_0] \sim_{i-1} n\langle R'\rangle[D_0]$. By the operational semantics, $k\langle S\rangle[D] \Longrightarrow D'$ for $D' \equiv n\langle R'\rangle[D_0], k\langle S\rangle[D_1]$. Since $\sim_{i-1}$ is a congruence, we conclude $C' \sim_{i-1} D'$.

The remaining cases are straightforward. □

We continue the comparison of equivalences on the set of configuration by relating spatial congruence with structural congruence. Note that it makes no sense to compare spatial congruence and structural congruence directly: if $P, Q \in \mathcal{P}$ are bisimilar but not equal, then $n\langle P\rangle[]$ and $n\langle Q\rangle[]$ are certainly spatially congruent, not structurally congruent. For this reason, we introduce *weak* structural congruence, which extends structural congruence to consider configurations as congruent, whose controlling processes are bisimilar. The formal definition is as follows:

Definition 15 Weak structural congruence *is the least relation R generated by the rules of Definition 2, plus the rule*

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

where $n \in \mathcal{N}$, $C, D \in \mathcal{C}$ 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.

Coming back to the example at the beginning of the section, note that $n\langle P\rangle[]$ and $n\langle Q\rangle[]$ are weakly structurally 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, we can show that spatial congruence implies structural congruence. This result hinges on the following lemma, which demonstrates that spatial congruence is closed under labelled reductions.

Lemma 16 *Spatial congruence is closed under labelled reduction.*

Proof: Suppose $n \in \mathcal{N}$ and $C, D \in \mathcal{C}$ are spatially congruent with $C \stackrel{l}{\Longrightarrow} C''$. 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 $Q \xrightarrow{\mathtt{in}\, n} Q'$ 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 $D \overset{\mathtt{in}\, n}{\Longrightarrow} D_1, m\langle Q'\rangle[E'] \cong C_1, m\langle P'\rangle[E]$.

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 $n\langle R\rangle[k\langle R\rangle[]], \_$. □

We are now ready to state and prove the main result of this section:

Proposition 17 *Spatial congruence and weak structural 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 $\mathcal{F} = \{(C, D) \in \mathcal{C} \times \mathcal{C} \mid C \cong D, C \not\equiv D\}$ of felons is non empty. For $C \in \mathcal{C}$, we define the *size* of C, $\mathrm{size}(C)$, by induction as follows: $\mathrm{size}(\mathbf{0}) = 0, \mathrm{size}(C, D) = \mathrm{size}(C) + \mathrm{size}(D)$, $\mathrm{size}(n\langle P\rangle[C']) = 1 + \mathrm{size}(C')$.

Since the standard ordering on natural numbers is a well-ordering, there is a pair (C, D) of felons, such that $\mathrm{size}(C)$ is minimal, that is, for all $(C', D') \in \mathcal{F}$ we have $\mathrm{size}(C') \geq \mathrm{size}(C)$. 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 $\mathrm{size}(C_0) < \mathrm{size}(C)$ and $\mathrm{size}(C_1) < \mathrm{size}(C)$, 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 $\mathrm{size}(C_0) < \mathrm{size}(C)$, 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 (Lemma 16) 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 $C \not\equiv D$. □

So far, we have shown that labelled bisimulation is contained in spatial congruence, which is in turn contained in structural congruence. In the following

section, we introduce a spatial logic and describe the relationship between structural congruence and logical equivalence.

4 A Spatial Logic for BasicSail

In the previous section, we have shown a chain of implications between different equivalences on the set of configurations: labelled bisimilarity implies spatial congruence, which in turn implies weak structural congruence. This section adopts a logical view and closes the chain of implications by showing that weak structural congruence implies logical equivalence, which is then proven to contain labelled bisimilarity. Using the setup from the previous section, this hinges on the fact that that the underlying processes are image finite. Our logic is very similar in style to modal logics used to reason about the power algebra associated with an algebraic structure: we obtain a hybrid of modal logic and separation logic [31,35]. In style, this logic is very similar the logics discussed in [15,11] except for the absence of linear implication. However, as we shall see later, linear implication can be added at no extra cost.

As before, our definitions and results are parametric in a set $\mathcal{N}$ of names and the associated set $\mathcal{L}$ of labels. We now introduce the logic we are going to work with.

Definition 18 (Spatial Logic: Syntax) *The language* L *of* spatial logic *is the least set of formulas according to the grammar*

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

where $n \in \mathcal{N}$, $l \in \mathcal{L}$ *and* R *ranges over the relations* $\downarrow, \Longrightarrow$ *and* $\stackrel{l}{\Longrightarrow}$ *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 $\stackrel{l}{\Longrightarrow}$). Finally, a formula of type $\langle \circlearrowright \rangle \phi\psi$ asserts that the current configuration can be split into two subconfigurations, the first satisfying ϕ and the second ψ.

Definition 19 (Spatial Logic: Semantics) *The semantics of propositional*

connectives is as usual. For the modal operators, we put, for $C \in \mathcal{C}$*:*

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

where R *ranges over* $\Longrightarrow, \downarrow$ *and* $\stackrel{l}{\Longrightarrow}$ *for* $l \in \mathcal{L}$ *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* $\mathrm{Th}(C) = \mathrm{Th}(D)$*; this is denoted by* $C =_L D$*.*

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 is invariant under structural congruence, adding one more item to our chain of implications:

Lemma 20 *Weak structural congruence is contained in logical equivalence.*

Proof: Straightforward by induction on the definition of weak structural congruence. The case of two bisimilar controlling processes uses the standard fact that bisimulation implies logical equivalence in process calculi (see e.g. [7,43]). □

We now close the chain of relation between the different relation on configurations by showing that logical equivalence implies labelled bisimulation; the proof uses standard techniques in modal logic, see e.g. [7].

Proposition 21 *Logical Equivalence is contained in labelled bisimulation.*

Proof: We show that

$$=_L = \{(C, D) \in \mathcal{C} \mid C,\ D \text{ logically equivalent}\}$$

is a labelled bisimulation. Using Lemma 10, closure under $\Longrightarrow$, $\downarrow$, $\stackrel{l}{\Longrightarrow}$, $\mathbf{0}$ and $@n$ are straightforward, see e.g. [7]. We just demonstrate that $=_L$ is closed under forest reduction.

To this end, suppose that $C, D \in \mathcal{C}$ with $C =_L D$ and $C \circlearrowleft (C_0, C_1)$. Suppose for a contradiction that for all D_0, D_1 with $D \circlearrowleft (D_0, D_1)$ we have $D_0 \neq_L C_0$ or $D_1 \neq_L C_1$.

Thus for all (D_0, D_1) with $D \circlearrowleft (D_0, D_1)$ there is $i = i(D_0, D_1) \in \{0, 1\}$ and $\phi_{i(D_0,D_1)}$ with $C_i \models \phi_i$ but $D_i \not\models \phi_i$. Now, for

$$\phi = \bigwedge\{\phi_{i(D_0,D_1)} \mid i(D_0, D_1) = 0\} \quad \text{and} \quad \psi = \bigwedge\{\phi_{i(D_0,D_1)} \mid i(D_0, D_1) = 1\}$$

we have that $C \models \langle \circlearrowleft \rangle(\phi, \psi)$ but $D \not\models \langle \circlearrowleft \rangle(\phi, \psi)$, contradicting $\mathrm{Th}(C) = \mathrm{Th}(D)$. □

We now conclude the investigation of the basic calculus by a comparison of the different forms of equivalence we have discussed so far.

Theorem 22 *In the BasicSail calculus, labelled bisimilarity, spatial congruence, logical equivalence and weak structural congruence coincide.*

The above equivalences all apply to the basic calculus, that is, the calculus without local names. Before extending our results to the calculus with local names, discuss the impact of adding linear implication to our logic.

Typically, spatial logics for reasoning about mobile processes, for example [15,11,12] contain linear implication $\triangleright$ as further connective. We have chosen not to include linear implication into the spatial logic for the basic calculus, since the main characterisation result, Theorem 22, can be proved without having linear implication available. Our logic is thus more similar in nature to that of [13]. This section shows, that linear implication can be added without destroying invariance under structural congruence.

Definition 23 *The language* $\mathsf{L}^{\triangleright}$ *of* spatial logic with linear implication *is the least set of formulas according to the grammar*

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

where the semantics is given as in Definition 18, plus the clause

$$C \models \phi \triangleright \psi \quad \textit{iff} \quad \forall D. D \models \phi \implies D, C \models \psi$$

for $C \in \mathcal{C}$. *If* $\mathrm{Th}(C) = \mathrm{Th}(D)$ *for* $C, D \in \mathcal{C}$, *we call* C *and* D *logically equivalent, which we denote by* $=^{\triangleright}_{L}$.

The connective $\triangleright$ is called *linear implication*: it stipulates that the formula ψ holds in presence of all configurations satisfying ϕ. It is sometimes helpful to think of ϕ as a property that needs to be guaranteed to hold in presence of all possible attackers which satisfy ψ.

It is immediately clear from the definition of the semantics of $\triangleright$ that linear implication does not allow to distinguish between structurally congruent configurations.

Lemma 24 *Suppose* $C \equiv D$ *are weakly structurally congruent and* $\phi \in \mathsf{L}^{\triangleright}$. *Then* $C \models \phi$ *iff* $D \models \phi$.

Proof: It follows from Theorem 22 that the statement holds for formulas not containing $\triangleright$. It follows directly from the definition of $\triangleright$ that this result carries

over to $\mathsf{L}^{\triangleright}$. $\square$

Using Theorem 22, we immediately obtain that linear implication does not help to distinguish configurations which are labelled bisimilar (or spatially congruent, for that matter).

Corollary 25 *Suppose $C \cong D$ are spatially congruent. Then $C =^{\triangleright}_{L} D$.*

5 LocalSail: A Calculus with 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, as we do not have name restriction avaliable at the process level. 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 26 (LocalSail) *The set $\mathcal{C}$ of configurations in LocalSail 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 $P \in \mathcal{P}$. If $\vec{n} = (n_1, \ldots, n_k)$, we write $(\nu\vec{n})$ for $(\nu n_1) \ldots (\nu n_k)$. 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 $\mathrm{fn}(P) = \{n \in \mathcal{N} \mid n \text{ free in } P\}$.*

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 2, augmented with α*-equivalence and the rules* $(\nu n)(A, B) \equiv ((\nu n)A), B$ *whenever* n *does not occur freely in* B *and the axiom* $(\nu n)\mathbf{0} \equiv \mathbf{0}$.

The operational semantics *is given as in Definition 2, augmented with the rule*

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

for $C, C' \in \mathcal{C}$ *and* $n \in \mathcal{N}$. *The extension of BasicSail with local names is called* LocalSail.

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 needs to assume that the set of processes is closed under substitution. Formally, we have the following coinductive definition:

Definition 27 *Let* $l = \mathtt{op}\, m \in \mathcal{L}$ *with* $\mathtt{op} \in \{\mathtt{in}, \mathtt{open}, \mathtt{out}\}$. *If* $n, k \in \mathcal{N}$, *we put* $l[n/k] = l$, *if* $m \neq k$, *and* $l[n/k] = \mathtt{op}\, n$ *if* $m = k$.

Suppose $P, Q \in \mathcal{P}$ *and* $n, k \in \mathcal{N}$. *We say that* Q *is* $[n/k]$- *bisimilar to* P, *denoted by* $Q \sim P[n/k]$, *if*

- $P \stackrel{\alpha}{\to} P' \implies \exists Q'.Q \stackrel{\alpha[n/k]}{\to} Q'$ *and* $P' \sim Q'[n/k]$.
- $Q \stackrel{\alpha[n/k]}{\to} Q' \implies \exists P'.P \stackrel{\alpha}{\to} P'$ *and* $P' \sim Q'[n/k]$.

We say that $\mathcal{P}$ *is* substitution closed, *if, for all* $P \in \mathcal{P}$ *and all* $n, k \in \mathcal{N}$, *there is* $Q \in \mathcal{P}$ *with* $Q \sim P[k/n]$. *If this is the case, we put*

$$m\langle P\rangle[C][k/n] \equiv \begin{cases} k\langle P[k/n]\rangle[C[k/n]] & \text{if } m = n \text{ and } Q \sim P[k/n] \\ n\langle P[k/n]\rangle[C[k/n]] & \text{if } m \neq n \text{ and } Q \sim P[k/n] \end{cases}$$

and extend this definition to the whole of $\mathcal{C}$ *by putting*

$$\mathbf{0}[k/n] \equiv \mathbf{0} \qquad (C, D)[k/n] \equiv C[k/n], D[k/n]$$

where $C, D \in \mathcal{C}$, $P \in \mathcal{P}$ *and* $n, m, k \in \mathcal{N}$.

In order to be able to deal with α-equivalence, we therefore assume for the remainder of the section that $\mathcal{P}$ is substitution closed; this can always be achieved by adding the missing substitution instances to $\mathcal{P}$. Despite of its name, closure under substitutions is not a syntactic notion: it applies to an arbitrary labelled transition system. Using this notion of substitution on the process level, the inductive extension to configurations is standard.

Before investigating the logical and algebraic theory of the calculus with local names, we continue the discussion of Example 4. Recall that we had an agent

in a home location, whose sole purpose was to transport clients inside home. 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 28 *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 4 and put*

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

Using scope extrusion, we have the same chain of reductions as in Example 4. 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. If we simply transfer the definition of spatial congruence to the setting with local names, we can not expect to obtain the same match between the logical, syntactical and algebraic theory. Consider for example $C \equiv \mathbf{0}$ and $D \equiv (\nu n)n\langle\mathbf{0}\rangle[]$. Clearly $C \not\equiv D$, but it is easy to see that C and D are structurally congruent.

In order to obtain a similar characterisation as in the calculus without local names, we therefore have to extend the definition of spatial bisimulation, and demand closure under name revelations. We think of name revelation as an additional experiment which we can perform on configurations: for two configurations to be equivalent, they have to behave equivalently even if we expose one of their hidden names.

Definition 29 *Suppose $C \in \mathcal{C}$ and $n, k \in \mathcal{N}$. We put*

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

whenever $n \notin \text{fn}(C)$ (free names cannot be revealed – they are not secret). We consider the following equivalences:

- spatial bisimulation *(resp.* spatial congruence*) is the largest symmetric (resp. congruence) 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 $n \in \mathcal{N}$).*
- labelled bisimulation *is the largest spatial bisimulation, which is closed under labelled reduction (Definition 9).*
- 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.*

In spite of the syntactic similarities, our definition is only superficially related

to Sangiorgi's open bisimulation [37] and the equivalences used in the fusion calculus [33]. The use of substitution in the above definition is solely used to consistently rename a hidden name, whereas open bisimilarity uses substitution to deal with the synchronous communication of names. Moreover, open bisimilarity is also meaningful in absence of local names.

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:

Proposition 30 *The following hold in the LocalSail calculus:*

(1) Labelled bisimulation contains spatial congruence
(2) spatial congruence contains weak structural congruence

Proof: To show that labelled bisimulation is contained in spatial congruence, we extend the proof of Proposition 14 and show, that labelled bisimulation is a congruence. We need to deal with three configuration-forming operations: Spatial Composition, tree construction and name restriction. Note that Lemma 10 and Proposition 12 are also valid for the set $\mathcal{L} = \{\mathtt{op}\, n \mid \mathtt{op} \in \{\mathtt{in}, \mathtt{out}, \mathtt{open}, \mathtt{rev}\} \text{ and } n \in \mathcal{N}\}$ of labels; we hence have to show that each relation $\sim_i$, as defined in Definition 11 is a congruence, for every $i \in \mathbb{N}$.

(1) Congruence w.r.t. spatial composition: Suppose $C \sim_i D$ and $E \in \mathcal{C}$. We have to show that $C, E \sim_i D, E$. Again we focus only on the nontrivial clauses in the definition of $\sim_i$ and only treat the case of spatial reductions. The case $C \Longrightarrow \tilde{C}$ and $C' \equiv \tilde{C}, E$ and $E \Longrightarrow \tilde{E}$ with $C' \equiv C, \tilde{E}$ are trivial. For the other cases, we assume that $C \equiv (\nu\vec{n})C_0$ and $E \equiv (\nu\vec{m})E_0$ where there is no occurrence of ν at the top level of either C_0 or E_0. In order for C and D to interact, both must perform scope extrusion, that is, we must have $C, E \equiv (\nu\vec{n})(\nu\vec{m}),\ C_0, E_0 \Longrightarrow C'$ and $C, E \Longrightarrow (\nu\vec{n})(\nu\vec{m})C'$. Since $C \leftrightarrows E$, we have $D \overset{\mathtt{rev}\, n_1}{\Longrightarrow} \ldots \overset{\mathtt{rev}\, n_k}{\Longrightarrow} D_0 \leftrightarrows C_0$ where D_0 has no occurrence of name restriction at the top level. As $C_0 \leftrightarrows E_0$ and both have no name restriction at their top level, we can proceed as in Proposition 12.

(2) Congruence w.r.t. tree construction: As in the proof of Proposition 12.

(3) Congruence w.r.t. name restriction: Suppose $C \sim_i D$; we have to show that $(\nu n)C \sim_i (\nu n)D$. Clearly $(\nu n)C$ and $(\nu n)D$ have the same spatial reductions, as their only reductions can be performed under the ν-binder (Definition 26). The only labelled reductions either C or E can perform are the revelation of the bound name n, which can be matched since $C \sim_i D$.

For the second implication, the minimal witness argument used in Proposition 17 has to me modified as follows: We put $\mathrm{size}(\nu n)C = \mathrm{size}(C)$ and consider

the set $\mathcal{F} = \{(C, D) \in \mathcal{C} \times \mathcal{C} \mid C \cong D, C \not\equiv D\}$ of felons. If $(C, D) \in \mathcal{F}$ such that size(C) is minimal, we have to consider the additional case that $C \equiv (\nu\vec{n})C_0$ with $\vec{n} = (n_0, \ldots, n_k)$. In this case, $C \stackrel{\mathtt{rev}\, n_0}{\Longrightarrow} \ldots \stackrel{\mathtt{rev}\, n_k}{\Longrightarrow} C'$, where $C' \not\equiv (\nu m)C''$ for all m, C''. Hence $D \stackrel{\mathtt{rev}\, n_0}{\Longrightarrow} \ldots \stackrel{\mathtt{rev}\, n_k}{\Longrightarrow} D'$ with $C' \cong D'$. Now $(C', D') \in \mathcal{F}$, which reduces this case to one of the two cases discussed in the proof of Proposition 17. □

In order to transfer the characterisation result to a logical setting, we introduce a hidden name quantifier in the style of Gabbay and Pitts [20]:

Definition 31 *The language of* spatial logic with local names *is the least set according to the following grammar*

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

Given $C \in \mathcal{C}$ and $\phi \in \mathsf{L}$, satisfaction $C \models \phi$ is as in Definition 18, plus the clause

$$C \models \mathsf{H}n.\phi \text{ iff there is } C' \in \mathcal{C} \text{ s.t. } C \stackrel{\mathtt{rev}\, n}{\Longrightarrow} C' \text{ and } C' \models \phi$$

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*, denoted by $C =_L D$, if* $\mathrm{Th}(C) = \mathrm{Th}(D)$.

Since the relation $\stackrel{\mathtt{rev}\, n}{\Longrightarrow}$ (for $n \in \mathcal{N}$) are image-finite, Lemma 10 and Proposition 12 remain valid in the calculus with local names. We thus obtain

Proposition 32 *In the LocalSail calculus:*

(1) weak structural congruence is contained in logical equivalence.
(2) logical equivalence is contained in labelled bisimulation.

Proof: The first claim is immediate from the definition of weak structural congruence in the calculus with local names. Note that the relations $\stackrel{\mathtt{rev}\, n}{\Longrightarrow}$ are image finite for all $n \in \mathcal{N}$. This allows us to preceed as in the proof of Proposition 21 for the second claim, which we extend by showing that logical equivalence $=_L$ is closed under revelation. Assume for a contradiction that $C, D \in \mathcal{C}$ with $C =_L D$, $C \stackrel{\mathtt{rev}\, n}{\Longrightarrow} C'$ but we have $C' \neq_L D'$ for all D' with $D \stackrel{\mathtt{rev}\, n}{\Longrightarrow} D'$. Since the set $R = \{D' \in \mathcal{C} \mid D \stackrel{\mathtt{rev}\, n}{\Longrightarrow} D'\}$ of reducts is finite by assumption, we have a formula $\phi_{D'}$ for every $D' \in R$ s.t. $C' \models \phi_{D'}$ but $D' \not\models \phi_{D'}$. Hence $C \models \bigwedge_{D' \in R} \phi_{D'}$ but $D \not\models \bigwedge_{D' \in R} \phi_{D'}$, which contradicts $C =_L D$. □

As a corollary, we obtain that the characterisation of Theorem 22 carries over to the calculus with local names.

Theorem 33 *The notions of labelled bisimulation, spatial congruence, weak structural congruence and logical equivalence coincide for the LocalSail calculus.*

6 MultipleSail: A Calculus with Multiple Names

In this section, we discuss a second extension of the BasicSail calculus and allow each location to have multiple names. Multiple names can be used to model network devices with more than one network interface; we allow for these interfaces to be switched on or off independently of each other. This feature can be used for example to model firewalls, which have one interface to the outside and a second network connection to a (protected) internal network. While this is a very realistic assumption, it is – to the best of our knowledge – not present in other calculi which can be used to model mobile computation.

Due to the layered structure of BasicSail, every action of a controlling process takes place in a unique location. Therefore, it is straightforward to allow processes to manipulate the names of the locations which they control. We can therefore easily model the addition and deletion of names using two extra primitives $\mathtt{up}\, n$ (add the name n to the names of the present location) and $\mathtt{down}\, n$ (remove the name n from the set of names of the present location). These actions correspond to switching network interfaces on and off, since every network interface comes with a unique name. This in particular covers the case where a location has more than one network interface – or none at all. If we extend the syntax of configurations to include multiple names, a typical singleton configuration has the form $\vec{n}\langle P\rangle[C]$, where $\vec{n} = (n_1, \ldots, n_k)$ is a list of names.

In contrast to the extension of BasicSail with local names, which amounts to adding extra capabilities in the construction of configurations, multiple names require to add new capabilities to the underlying processes. More precisely, we need to assume that $(\mathcal{P}, \longrightarrow)$ is an image finite labelled transition system, where the labels incorporate $\mathtt{up}\, n$ and $\mathtt{down}\, n$. The following convention makes this precise.

Notation Throughout the section, we fix a set $\mathcal{N}$ of names and consider the set $\mathcal{L} = \{\mathtt{op}\, n \mid \mathtt{op} \in \{\mathtt{in}, \mathtt{out}, \mathtt{open}, \mathtt{up}, \mathtt{down}\} \text{ and } n \in \mathcal{N}\}$. Furthermore, we fix a labelled transition system $(\mathcal{P}, \longrightarrow)$, where $\mathcal{P} \neq \emptyset$ and $\longrightarrow \subseteq \mathcal{P} \times \mathcal{L} \times \mathcal{P}$ is image finite.

Based on a set of processes that can exercise control over the names of a location, the MultipleSail calculus is given as follows:

Definition 34 (MultipleSail) *The set of configurations of MultipleSail is the least set according to the grammar*

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

where $P \in \mathcal{P}$ is a process and $\vec{n} = (n_1, \ldots, n_k) \in \mathcal{N}^*$ is a list of names. As

before, configurations are considered up to structural congruence $\equiv$ *given by the axioms of Definition 2, augmented with*

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

where σ *is a permutation of* $\{1, \ldots, k\}$.

In the sequel, we write $n \in (n_1, \ldots, n_k)$ *iff* $n = n_j$ *for some* $1 \leq j \leq k$. *The operational semantics is now given by extending the rules given in Definition 3 with the rules*

$$\frac{P \xrightarrow{\mathtt{up}\, n} P' \qquad n \notin \vec{n}}{\vec{n}\langle P \rangle [A] \longrightarrow \vec{n} \oplus n(P')[A]} \qquad \frac{P \xrightarrow{\mathtt{down}\, n} P' \qquad n \in \vec{n}}{\vec{n}\langle P \rangle [A] \longrightarrow \vec{n} \ominus n(P')[A]}$$

where

$$(n_1, \ldots, n_k) \oplus n = \begin{cases} (n_1, \ldots, n_k, n) & n \neq n_j \text{ for all } j = 1, \ldots, n \\ (n_1, \ldots, n_k) & \text{otherwise} \end{cases}$$

and similarly

$$(n_1, \ldots, n_k) \ominus n = \begin{cases} (n_1, \ldots, n_{j-1}, n_{j+1}, \ldots, n_k) & 1 \leq j \leq k \text{ and } n = n_j \\ (n_1, \ldots, n_k) & \text{otherwise} \end{cases}$$

The resulting extension of BasicSail is called MultipleSail.

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. Note that activating a new name (via $\mathtt{up}\, n$) at a location where the name is already in use has no effect. Similarly, removing a name from a location which is not present will not change the spatial structure.

In particular, a location can have no name at all. The following example contrasts this with hidden names.

Example 35 *(1) The effect of having no name at all cannot be captured with local names, since nameless locations are also nameless for locations from within. 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) Unnamed locations are by no means immobile. Consider the configuration $(n)\langle \mathtt{down}\, n.\mathbf{0} \rangle [A], ()\langle \mathtt{in}\, n.\mathbf{0} \rangle [B]$. *This example also illustrates that the movement only succeeds, if the unnamed agent is lucky enough to enter into his partner* before *the name disappears.*

In the MultipleSail calculus, we cannot expect that spatially congruent processes (in the sense of Definition 5) to contain structural congruence. The reason is that the controlling processes of two spatially congruent processes are not process bisimilar. This occurs for example, if the controlling process allows for de-activating a name, which is not present in the configuration, as e.g. for $()\langle \mathtt{down}\, n.O\rangle[]$ and $\langle \mathtt{down}\, k.0\rangle[]$. In order to achieve a match between the different process equivalences, we therefore have to allow for an additional observation: The changing of names. The formal definition of spatial bisimulation and congruence in MultipleSail is as follows:

Definition 36 *Suppose $C \in \mathcal{C}$ and $n \in \mathcal{N}$. We put*

$$C \stackrel{\oplus n}{\Longrightarrow} C' \quad \textit{iff} \quad C \equiv \vec{n}\langle P\rangle[C_0] \textit{ and } C' \equiv \vec{n} \oplus n\langle P\rangle[C_0]$$

and analogously

$$C \stackrel{\ominus n}{\Longrightarrow} C' \quad \textit{iff} \quad C \equiv \vec{n}\langle P\rangle[C_0] \textit{ and } C' \equiv \vec{n} \ominus n\langle P\rangle[C_0].$$

We say that a relation is closed under name changes*, if it is closed under $\stackrel{\oplus n}{\Longrightarrow}$ and $\stackrel{\ominus n}{\Longrightarrow}$ for all $n \in \mathcal{N}$.*

We consider the following equivalences:

- spatial bisimulation $\simeq$ *(resp.* spatial congruence $\cong$*) is the largest symmetric (resp. congruence) relation that is closed under spatial reduction $\Longrightarrow$, forest reduction $\circlearrowleft$, subtree reduction $\downarrow$, top level names $@n$ and name changes $\stackrel{\oplus n}{\Longrightarrow}, \stackrel{\ominus n}{\Longrightarrow}$ (for all $n \in \mathcal{N}$).*
- labelled bisimulation *is the largest spatial bisimulation which is closed under labelled reduction (see Definition 9).*
- 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.*

We think of closure under name changes as an experiment, which we can perform on a singleton configuration. For two (singleton) configurating to be equivalent, we require that they exhibit the same behaviour even when we change the names of their top level locations. This additional observation ensures that labelled bisimulation is a congruence, and that spatial congruence implies weak structural congruence.

Proposition 37 *The following hold for the MultipleSail calculus:*

(1) Labelled bisimulation contains spatial congruence.
(2) Spatial congruence contains weak structural congruence.

Proof: We extend the corresponding results for the basic calculus. For the first

claim, we extend the proof of Proposition 14 and show, that labelled bisimulation is a congruence. As for the LocalSail calculus, we note that 10 and Proposition 12 remain valid for the set $\mathcal{L} = \{\mathtt{op}\, n \mid \mathtt{op} \in \{\mathtt{in}, \mathtt{out}, \mathtt{open}, \mathtt{up}, \mathtt{down}, \oplus, \ominus\}$ and $n \in \mathcal{N}\}$. We thus have to show that $\sim_i$ (Definition 11) is a congruence w.r.t. spatial composition and tree construction.

(1) Congruence w.r.t. spatial composition: Assume that $C \equiv \vec{n}\langle P\rangle[C_0], C_1$ and $C \sim_i D$. We fix $E \in \mathcal{C}$ and show that $C, E \sim_i D, E$, where we only treat spatial reductions induced by (de)activating a name.

Case 1: C activates a new name. Formally $C, E \Longrightarrow C'$ with $P \xrightarrow{\mathtt{up}\, k} P'$ and $C' \equiv \vec{n} \oplus k\langle P'\rangle[C_0], C_1, E$. Using forest reduction and Lemma 7, we can assume that $D \equiv \vec{n}\langle Q\rangle[D_0], D_1$ where $\vec{n}\langle P\rangle[C_0] \sim_{i-1} \vec{n}\langle Q\rangle[D_0]$ and $C_1 \sim_{i-1} D_1$. Now $\vec{n}\langle P\rangle \stackrel{\mathtt{up}\, k}{\Longrightarrow} \vec{n}\langle P'\rangle[C_0]$. By closure under labelled reduction, we have $\vec{n}\langle Q\rangle[D_0] \stackrel{\mathtt{up}\, n}{\Longrightarrow} \vec{n}\langle Q'\rangle[D_0]$ with $\vec{n}\langle P'\rangle[C_0] \sim_{i-1} \vec{n}\langle Q'\rangle[D_0]$. As $\sim_{i-1}$ is closed under name changes, also $\vec{n} \oplus k\langle P'\rangle[C_0] \sim_{i-1} \vec{n} \oplus k\langle Q'\rangle[D_0]$, and the claim follows, since $\sim_{i-1}$ is a congruence.

Case 2: C de-activates a new name. Formally $C, E \Longrightarrow C'$ with $P \xrightarrow{\mathtt{down}\, k} P'$ and $C' \equiv \vec{n} \ominus k\langle P'\rangle[C_0], C_1, E$: Similar.

The cases, where E activates or de-activates a new name, are straightforward.

(2) Closure w.r.t. tree construction: Immediate.

For the second claim we use the minimal witness argument of Proposition 17, which applies once we have established that spatial congruence is closed under labelled reduction. To this end, assume $C \cong D$ are configurations and $C \stackrel{\mathtt{op}\, n}{\Longrightarrow} C'$. We have to exhibit $D' \cong C'$ such that $D \stackrel{\mathtt{op}\, n}{\Longrightarrow} D'$. We only treat the new cases $\mathtt{op} \in \{\mathtt{up}, \mathtt{down}\}$. So suppose $C \cong D$ and $C \stackrel{\mathtt{op}\, n}{\Longrightarrow} C'$. Then we can assume that $C \equiv \vec{n}\langle P\rangle[C_0], C_1$ and $C' \equiv \vec{n}'\langle P'\rangle[C_0], C_1$. By closure under forest reduction $\circlearrowleft$, we have $D \cong \vec{n}\langle Q\rangle[D_0], D_1$ with $\vec{n}\langle Q\rangle[D_0] \cong \vec{n}\langle P\rangle[C_0]$ and $D_1 \cong C_1$.

Case 1: $\mathtt{op} = \mathtt{up}$. We can assume without loss of generality that $k \notin \vec{n}$; otherwise we use closure under name changes and replace $\vec{n}$ with $\vec{n} \ominus k$. Since $n\langle P\rangle[C_0] \Longrightarrow \vec{n} \oplus k\langle P'\rangle[C_0]$, there is $D_0' \cong \vec{n} \oplus k\langle P'\rangle[C_0]$ with $\vec{n}\langle Q\rangle[D_0] \Longrightarrow D_0'$. Again by closure under forest reduction and top level names, we have $D_0' \equiv \vec{n} \oplus k\langle Q'\rangle[D_0'']$. But this can only happen if $Q \xrightarrow{\mathtt{up}\, k} Q'$ and $D_0'' \in @k$, we have $D_0'' \equiv D_0$. The claim follows, since $\cong$ is a congruence.

Case 2: $\mathtt{op} = \mathtt{down}$: Similar. □

We now take a logical point of view and give a logical characterisation of spatial congruence, which is very similar to the logical characterisation in the

setting with local names (Section 5). Similar to the hidden name quantifier, we introduce a new logical operator, which deals with name changes.

Definition 38 *The language of* spatial logic with multiple names *is the least set according to the following grammar*

$$\mathsf{L} \ni \phi, \psi ::= \epsilon \mid @n \mid \mathit{ff} \mid \phi \to \psi \mid \langle R \rangle \phi \mid \langle \circlearrowleft \rangle \phi\psi \mid \oplus n.\phi \mid \ominus n.\phi$$

Given $C \in \mathcal{C}$ and $\phi \in \mathsf{L}$, satisfaction $C \models \phi$ is as in Definition 18, plus the clause

$$C \models \oplus n.\phi \text{ iff } C \equiv \vec{k}\langle P \rangle [C'] \text{ and } \vec{n} \oplus k \langle P \rangle [C'] \models \phi,$$

and accordingly for $\ominus$. 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, *denoted by $C =_L D$, if* $\mathrm{Th}(C) = \mathrm{Th}(D)$.

As in the calculus with local names, we have the following result:

Proposition 39 *The following hold in the MultipleSail calculus:*

(1) weak structural congruence is contained in logical equivalence.
(2) logical equivalence is contained in labelled bisimulation.

Proof: As for Proposition 32. □

Using name changes as additional observation, we achieve a perfect match between the syntactical, logical and algebraic theory also for the MultipleSail calculus:

Theorem 40 *In the MultipleSail calculus, labelled bisimulation, spatial congruence, weak structural congruence and logical equivalence coincide.*

7 Conclusions and Related Work

We have presented a coordination approach to mobile components: the BasicSail calculus and two extensions LocalSail and MultipleSail. The main novelty of our approach lies in the fact that our calculus strictly distinguishes between the computational and the spatial aspects of distributed computation. Compared to other coordination models, the main novelty of our approach is the spatial structure, where we assume that locations are hierarchically organised. Our main result is that logical equivalence, structural congruence and spatial congruence agree; this indicates that our framework allows to add mobility to components in a transparent way.

The study of mobility goes back to Milner's π-calculus [27]. Further calculi are

the Fusion calculus [33], Nomadic Pict [45] and the distributed coordination language KLAIM [29]. The study of hierarchical re-configurable administrative domains was introduced by the Ambient [16] and the Seal calculus [44]. BasicSail follows these lines but distinguishes processes and configurations in an a priori way and concentrates on a even simpler set of operations for changing the topological structure.

The basic calculus and its variations were inspired by the Seal-Calculus. [44]. 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 [16], from which we have borrowed the primitives in, out and open. As we have pointed out before, our principal design decisions do not allow to embed the full ambient calculus into our framework.

The clear separation of spatial and computational aspects allows for the introduction of an additional layer which monitors the actions performed by the programs controlling the individual components to enforce security policies in the style of edit automata [25]. This allows for even more modularity, as security policies are added independently from the concrete realisation of the components and the mobility layer. The enforcement of security policies by means of an extra layer of control has also been studied in a single calculus; we mention box-π [42] and ambients with guardians [19]. Our notion of controlling process differs from the guardians of [19] as a guardian controls mobility, whereas our controlling process initiate a movement.

Separation of concerns has always been an important aspect of software architecture, and the glue which allows for the inter-operability between different components is studied in the context of coordination languages (see [32,2] for an overview). The language KLAIM [29] provides an integration between coordination and mobility. Based on the notion of tuple spaces in the style of Linda [18,21] it allows for modelling of distributed systems in the style of the π-calculus; in particular, the hierarchical structure of locations cannot be represented directly. Our approach of combining component is similar in spirit to that of [1,30] (which does not cater for mobility). There the authors differentiate between components, which provide certain services, and an additional layer, which describes the composition of components.

Spatial logics were studied by Cardelli and Caires [11,12], although to our knowledge not w.r.t. a clear characterisation of the expressive power. Such a characterisation (called "intensional bisimulation") was considered by Sangiorgi for a variant of the ambient calculus [38,39].

Our work on the comparison between different equivalences on mobile process

has to be seen in the context of the work of Merro and Hennessy [26] and Sangiori, Hirschkoff, Lozes [38,39]. Their results were obtained in an untyped setting, i.e. where one does not distinguish between processes and locations. Our results show, that similar comparisons can also be made in a typed setting, where furthermore one is independent of the underlying process calculus.

The clear distinction between process and location, enforced by our two-layered approach, can of course also be enforced by adding type system *a posteriori*. Existing type systems [9,17,14] do not account for this distinction, as they are designed with a different goal: In general, each type corresponds to a certain property of a process, and the type system allows for the derivation of these properties. Moreover, typing is only meaningful when the source code of the individual components is available, whereas we have assumed independence from the underlying language by just using a labelled transition semantics; see [40] for a discussion of this dichtonomy. We see partial typing [36] and the work on the box-π calculus and local subtyping [42,41], where untrusted components are put inside a wrapper, as steps towards the integration of both aspects. Finally, we remark that our approach of modelling mobile components and their connections is a special case of the bigraph models of Jensen and Milner [22]: The bigraphs corresponding to our setup are quite specialised, as our basic calculus does not directly allow to model connections between the different components.

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 name passing or multiple names, nor a variant which includes dynamic reconfiguration.

A preliminary version of this work has already appeared as [34]. The present paper differs from *loc.cit.* in the choice of the reduction relations which define spatial bisimulation. The present approach, resulting in a hybrid of separation logic and modal logic, is closer to the logics already discussed in an untyped setting. Also, the comparison of the different reduction relations follows a more standard pattern and uses standard proof principles, where available.

References

[1] F. Arbab. Abstract behavior types: A foundation model for components and their composition. In F. de Boer, M. Bonsangue, S. Graf, and W. de Roever, editors, *Proc. FMCO 2002 (Revised Lectures)*, volume 2852 of *Lect. Notes in Comp. Sci.*, pages 33–70. Springer, 2003.

[2] F. Arbab. What do you mean, coordination? *Bulletin of the Dutch Association for Theoretical Computer Science*, March 1998.

[3] F. Arbab, I. Herman, and P. Spilling. An overview of manifold and its implementation. *Concurrencey: Practice and Experience*, 5(1):23–70, 1993.

[4] J. Baumann, F. Hohl, M. Straßer, and K. Rothermel. Mole - concepts of a mobile agent system. *World Wide Web*, 1(3):123–137, 1998.

[5] F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with jade. In Cristiano Castelfranchi and Yves Lespérance, editors, *Proc. ATAL 2000*, volume 1986 of *Lect. Notes in Comp. Sci.*, pages 89–103. Springer, 2001.

[6] F. Bellifemine, A. Poggi, G. Rimassa, and P. Turci. An object oriented framework to realize agent systems. In *Proceedings of WOA 2000 Workshop*, pages 52–57, 2000.

[7] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[8] M. Bonsangue, F. Arbab, J. W. de Bakker, J. J. M. M. Rutten, A. Secutella, and G. Zavattaro. A transition system semantics for the control-driven coordination language MANIFOLD. *Theor. Comp. Sci.*, 240(1):3–34, 2000.

[9] M. Bugliesi and G. Castagna. Behavioural typing for safe ambients. *Computer Languages*, 28(1):61–99, 2002.

[10] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. In N. Kobayashi and B. Pierce, editors, *Proc. TACS 2001*, number 2215 in Lect. Notes in Comp. Sci., pages 38–63. Springer, 2001.

[11] 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.

[12] L. Caires and L. Cardelli. A spatial logic for concurrency (part ii). In L. Brim, P. Janc̀ar, M. Kr̀etìnskỳ, and A. Kuc̀era, editors, *Proc. CONCUR 2002*, volume 2421 of *Lecture Notes in Computer Science*, pages 209–225. Springer, 2002.

[13] L. Cardelli and G. Ghelli. A query language based on the ambient logic. In David Sands, editor, *Proc. ESOP 2001*, volume 2028 of *Lect. Notes in Comp. Sci.*, pages 1–22. Springer, 2001.

[14] L. Cardelli, G. Ghelli, and A. Gordon. Types for the ambient calculus. *Information and Computation*, 177(2):160–194, 2002.

[15] L. Cardelli and A. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. POPL 2000*, pages 365–377. ACM, 2000.

[16] L. Cardelli and A. Gordon. Mobile ambients. *Theor. Comp. Sci.*, 240(1):177–213, 2000.

[17] L. Cardelli, A. Gordon, and G. Ghelli. Mobility types for mobile ambients. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Proc. ICALP 1999*, volume 1644 of *Lect. Notes in Comp. Sci.*, pages 230–239. Springer, 1999.

[18] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, 1989.

[19] G. Ferrari, E. Moggi, and R. Pugliese. Guardians for ambient-based monitoring. In V. Sassone, editor, *Proc. FWAN 2002*, volume 66.3 of *Electr. Notes in Theoret. Comp. Sci.*, 2002.

[20] 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.

[21] D. Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[22] O. Jensen and R. Milner. Bigraph models for mobile processes. Technical Report UCAM-CL-TR-580, University of Cambridge Computer Laboratory, 2004.

[23] T. Lehman, A. Cozzi, Y. Xiong, J. Gottschalk, V. Vasudevan, S. Landis, P. Davis, B. Khavar, and P. Bowman. Hitting the distributed computing sweet spot with tspaces. *Computer Networks*, 35(4), 2001.

[24] F. Levi and D. Sangiorgi. Controlling interference in ambients. In *Proc. POPL 2000*, pages 352–364. ACM, 2000.

[25] J. Ligatti, L. Bauer, and D. Walker. Edit automata: enforcement mechanisms for run-time security policies. *International Journal of Information Security*, 2003. Submitted for Publication.

[26] M. Merro and M. Hennessy. Bisimulation congruences in safe ambients. In *Proc. POPL 2002*, pages 352–364. ACM, 2002.

[27] R. Milner. *Communicating and Mobile Systems: the π-Calculus.* Cambridge University Press, 1999.

[28] U. Montanari and V. Sassone. Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae*, 16(2):171–199, 1992.

[29] 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.

[30] O. Nierstrasz and F. Achermann. A calculus for modeling software components. In F. de Boer, M. Bonsangue, S. Graf, and W. de Roever, editors, *Proc. FMCO 2002 (Revised Lectures)*, volume 2852 of *Lect. Notes in Comp. Sci.*, pages 339–360. Springer, 2003.

[31] P.W. O'Hearn and D.J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2), 1999.

[32] G.A. Papadopoulos and F. Arbab. Coordination models and languages. *Advances in Computers*, 46, 1998.

[33] 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.

[34] D. Pattinson and M. Wirsing. Making components move: A separation of concerns approach. In F. de Boer, M. Bonsangue, S. Graf, and W. de Roever, editors, *Proc. FMCO 2002 (Revised Lectures)*, volume 2852 of *Lect. Notes in Comp. Sci.*, pages 487–507. Springer, 2003.

[35] D. Pym, P. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of BI. *Theoretical Computer Science*, 2003. to appear.

[36] J. Riely and M. Hennessy. Trust and partial typing in open systems of mobile agents. In *Proc. POPL 1999*, pages 93–104. ACM, 1999.

[37] D. Sangiorgi. A theory of bisimulation for π-calculus. *Acta Informatica*, 33, 1996.

[38] D. Sangiorgi. Extensionality and intensionality of the ambient logics. In *Proc. POPL 2001*, pages 4–13. ACM, 2001.

[39] 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.

[40] F. Schneider, G. Morrisett, and R. Harper. A language-based approach to security. In R. Wilhelm, editor, *Informatics – 10 Years Back, 10 Years Ahead*, volume 2000 of *Lect. Notes in Comp. Sci.*, pages 86–101. Springer, 2000.

[41] P. Sewell. Global/local subtyping and capability inference for a distributed pi-calculus. In K. Larsen, S. Skyum, and G. Winskel, editors, *Proc. ICALP 1998*, volume 1443 of *Lect. Notes in Comp. Sci.*, pages 695–706. Springer, 1998.

[42] P. Sewell and J. Vitek. Secure composition of insecure components. In *PCSFW: Proceedings of The 12th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.

[43] C. Stirling. *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer, 2001.

[44] J. Vitek and G. Castagna. Seal: A framework for secure mobile computation. *Internet Programming*, 1999.

[45] P. Wojciechowski and P. Sewell. Nomadic pict: Language and infrastructure design for mobile agents. *IEEE Concurrency*, 8(2):42–52, 2000.