Elimination of Quantifiers and Undecidability in Spatial Logics for Concurrency

Luís Caires [a], Etienne Lozes [b,*]

[a] *Departamento de Informática, FCT/UNL, Portugal*

[b] *LIP, Ecole Normal Superieure de Lyon, France*

Abstract

The introduction of spatial logics in concurrency is motivated by a shift of focus from concurrent systems towards distributed systems. Aiming at a deeper understanding of the essence of dynamic spatial logics, we study a minimal spatial logic without quantifiers or any operators talking about names. The logic just includes the basic spatial operators void, composition and its adjunct, and the next step modality; for the model we consider a tiny fragment of CCS. We show that this core logic can already encode its own extension with quantification over actions, and modalities for actions. From this result, we derive several consequences. Firstly, we establish the intensionality of the logic, we characterize the equivalence it induces on processes, and we derive characteristic formulas. Secondly, we show that, unlike in static spatial logics, the composition adjunct adds to the expressiveness of the logic, so that adjunct elimination is not possible for dynamic spatial logics, even quantifier-free. Finally, we prove that both model-checking and satisfiability problems are undecidable in our logic. We also conclude that our results extend to other calculi, namely the π-calculus and the ambient calculus.

Introduction

The introduction of spatial logics in concurrency has been motivated by a recent shift of focus from monolithic concurrent systems towards distributed computing systems. Such systems are by nature both concurrent and spatially distributed, in the sense that they are composed from a number of separate and independently observable units of behavior and computation. The central

* Corresponding author.
Email addresses: Luis.Caires@di.fct.unl.pt (Luís Caires), elozes@ens-lyon.fr (Etienne Lozes).

idea behind spatial logics is that for specifying distributed computations there is a need to talk in a precise way not just about pure behaviors, as is the case with traditional logics for concurrency, but about a richer model able to represent computation in a space. Such an increased degree of expressiveness is necessary if we want to specify with and reason about notions like locations, resources, independence, distribution, connectivity, and freshness. Spatial logics have been proposed for π-calculi [4,3], and for the ambient calculus [11,10]. Spatial logics for manipulating and querying semi-structured data have also been developed [9,8]. Closely related are the separation logics [21,20], introduced with the aim of supporting local reasoning about imperative programs.

The simplest spatial logic for concurrency, we may argue, is the one obtained by adding to boolean logic the very basic spatial connectives, namely void (0), composition $(- \mid -)$ and its logical adjunct $(- \triangleright -)$, and then the dynamic modality next step $(\Diamond -)$. This logic, based essentially on spatial observations, will be referred here by $\mathcal{L}_{spat}$. The basic spatial connectives can be used to specify the distribution of processes, 0 specifies the empty system (not to be confused with the inactive system), and $\mathcal{A} \mid \mathcal{B}$ specifies the systems that can be partitioned in two parts, one satisfying property $\mathcal{A}$ and the other satisfying property $\mathcal{B}$.

A typical spatial property expressible in this logic is

$$1 \triangleq \neg 0 \wedge \neg (\neg 0 \mid \neg 0)$$

A process satisfies 1 if and only if it is non void, and cannot be split as a composition of two separate non void processes, in other words, if it is single-threaded. A simple example of a property combining spatial and dynamic operators is the one expressed by the formula

$$\mathsf{Annihilate} \triangleq (\neg 0 \mid \neg 0) \wedge \Diamond 0$$

This formula specifies those processes that have (at least) two separate components and may reduce (in one step) to the void system.

For the composition adjunct (sometimes also called *guarantee*), we have that $\mathcal{A} \triangleright \mathcal{B}$ is satisfied by those processes that, whenever composed with a process satisfying property $\mathcal{A}$, are guaranteed to satisfy property $\mathcal{B}$.

Adjuncts allow the specification of contextual properties of systems. For instance, consider the formula

$$\mathsf{Erasable} \triangleq 1 \wedge (1 \blacktriangleright \Diamond 0)$$

Here, we use the De Morgan dual (existential version) of the composition adjunct, defined $\mathcal{A} \blacktriangleright \mathcal{B} \triangleq \neg (\mathcal{A} \triangleright \neg \mathcal{B})$. This formula specifies the single-threaded processes that can be composed with some other process to yield a system

that may evolve to the empty system, after a single reduction step. The composition adjunct is a powerful operation, allowing the logic to perform quite strong observations on processes. With the composition adjunct, logical validity can be internally defined in the sense that a process P satisfies the formula $(\neg\, \mathcal{A}) \triangleright \bot$ if and only if every process satisfies the formula $\mathcal{A}$ [11]: a consequence is that the validity-checking problem in logics containing $\mathcal{L}_{spat}$ is subsumed by the model-checking problem. The composition adjunct also supports certain forms of specification akin to a comprehension principle: for example, we may specify the set of all processes that have exactly an even number of parallel components. This is perhaps not surprising since the semantics of the composition adjunct involves a conditional quantification over all processes.

Adjunct-free spatial logics with modalities for behavioral observations (*e.g.,* [2]) are also able to render many interesting contextual properties. For example, the property Erasable just presented can be expressed by the formula

$$1 \wedge \exists x.\, \langle x \rangle 0$$

using an action modality. For another example, consider the formula

$$\mathsf{NoRace} \triangleq \neg\, \exists x.\, (\langle \overline{x} \rangle \top \mid \langle \overline{x} \rangle \top \mid \langle x \rangle \top)$$

A process satisfies NoRace if it does not have an immediate race condition on some communication channel.

Thus, one of the motivations for this work is to achieve a deeper understanding about the relative expressiveness of these two approaches, the purely spatial one, that builds on composition adjunct, and the spatial - behavioral one, that restricts contextual observations to those expressible by behavioral modalities.

For the sake of simplicity and generality, we interpret $\mathcal{L}_{spat}$ in a rather small fragment of choice-free CCS, a minimal calculus defined from the void process $\mathbf{0}$, parallel composition $- \mid -$, and action prefixing $\alpha.\, -$. This calculus turns out to conveniently abstract the kind of concurrent behavior present in both π- and ambient calculi, in the broad sense that interactions are local, and triggered by the presence of named capabilities.

At first, the logic $\mathcal{L}_{spat}$ seems quite weak, as far as expressiveness is concerned, when compared to other spatial logics [11,4,2]. For instance, it provides no constructs referring explicitly to names or actions, such as *e.g.*, the action modality $\langle n \rangle \mathcal{A}$ of behavioral logics, or the ambient match construction $n[\mathcal{A}]$ in the ambient logic, therefore formulas of $\mathcal{L}_{spat}$ are always closed (in the sense that they do not have free names or variables). As a consequence, satisfaction of $\mathcal{L}_{spat}$ formulas is invariant under swapping of any pair of actions in processes (a property usually called equivariance) because formulas cannot single out specific actions or names in processes.

Still, due to the presence of the $\Diamond$ operator, the logic is able to make some distinctions between actions, and substitution of actions (which are not mentioned in formulas) does not in general preserve satisfaction. For instance, let $P \triangleq \alpha \mid \overline{\beta}$. Then $P \models \neg \Diamond \top$ for $\beta \neq \alpha$, but $P\{\beta \leftarrow \alpha\} \not\models \neg \Diamond \top$.

These considerations lead to the general question of what is the largest relation between processes which are indistinguishable by the logical equivalence: answering this question crucially contributes to our understanding of the spatial model induced on processes by the simplest combination of logical observations.

The study of expressiveness for spatial logics usually goes through the definition of an adequate spatial bisimilarity $\approx$ along the lines of [17], such that $P \approx Q$ implies that P and Q are logically equivalent. However, this question turns out to be a rather difficult one to answer for the case of dynamic spatial logics, due to the presence of the composition adjunct operator $\triangleright$ together with the dynamic modality. Establishing the congruence of $\approx$ is key to ensure correctness of $\approx$, so that from $P \approx Q$ we conclude $P \mid R \approx Q \mid R$. For our logic however, such a property does not hold, due to equivariance. For instance, the processes $\alpha.\mathbf{0}$ and $\beta.\mathbf{0}$ are logically equivalent, but $\alpha.\mathbf{0} \mid \overline{\alpha}.\mathbf{0}$ and $\beta.\mathbf{0} \mid \overline{\alpha}.\mathbf{0}$ are not. Hence, this general approach does not seem to work well in this setting.

Despite many works about decidability of spatial logics, the question of model-checking spatial logics for concurrency with adjunct has not been fully settled yet. Results are known for some particular cases, where the logic includes just $\triangleright$ or $\Diamond$ [11,2], but there seems to be no work about the interesting combination of $\triangleright$ and $\Diamond$, as far as decidability is concerned. However, we believe that this issue lies at the heart and novelty of a purely spatial approach to verification of distributed systems. On the one hand, image-finiteness of the reduction relation gives a model-checking algorithm for adjunct-free logics [2]. On the other hand, in the absence of name quantifiers and name revelation it is also known that static fragments of spatial logics, that is spatial logics without quantifiers and dynamic operators, are decidable [6], so there could be some hope in obtaining decidability of model-checking for the whole of $\mathcal{L}_{spat}$.

We may attempt to answer the questions about expressiveness and decidability posed above by considering the extension $\mathcal{L}_{mod}$ of $\mathcal{L}_{spat}$ with the existential quantifier and quantified action modalities; for $\mathcal{L}_{mod}$, logical equivalence is much clearly intensional (close to structural congruence), and one may adapt the results of [13] to derive the undecidability of model-checking. But even if the composition adjunct $\triangleright$ induces undecidability, we need to raise the question of what is its actual contribution to the expressiveness of the logic. In previous work [19], Lozes has shown that in static spatial logics the adjunct connective can be eliminated in behalf of the remaining connectives, in the sense that

for any formula of such a logic there is a (possibly hard to find) logically equivalent adjunct-free formula. For example, for the formula $1 \triangleright (1 \mid 1)$ we have the equivalent adjunct free formula 1. An interesting question is then whether something similar happens in $\mathcal{L}_{mod}$: we could conceive that the expressive power of the composition adjunct could be somehow recovered by the presence of action modalities, given that both kinds of constructs allow some contextual observations to be made.

The logics $\mathcal{L}_{mod}$ and $\mathcal{L}_{spat}$ seem quite different as far as expressive power is concerned. The first one seems clearly intensional (in the technical sense that logical equivalence coincides with structural congruence), and undecidable. But for the second, as discussed above, it would be reasonable to hope for decidability, and expect a separation power coarser than structural congruence. All this turns out not to be the case.

The main result of this paper is that $\mathcal{L}_{mod}$ admits the elimination of quantifiers and action modalities in a precise sense (Theorem 2.1); on the way we also show that equality is internally definable. Our quantifier elimination result builds on techniques which are quite specific to spatial logics, and relies on a not obvious encoding of environments and valuations as processes, not on more traditional skolemization techniques: recall that target language of the encoding is the logic $\mathcal{L}_{spat}$ that does not contain variables or constants. So, we actually have to faithful encode action modalities and quantifications on actions by appropriate quantification on processes (using the composition adjunct), interactions on processes (using the next step dynamic modality $\Diamond$), and suitable structural observations on processes (*e.g.*, counting) relying on the basic spatial operators. Building on this surprising result, we then show that $\mathcal{L}_{spat}$ and $\mathcal{L}_{mod}$ have the same separation power (Theorem 3.3), and the same expressive power in a certain sense. As a consequence, we also characterize the separation power of $\mathcal{L}_{spat}$, showing that it coincides with structural congruence modulo permutation of actions (Theorem 3.3). Quantifier elimination is compositional and effective, allowing us to conclude that model-checking of both $\mathcal{L}_{spat}$ and $\mathcal{L}_{mod}$ is undecidable (Theorem 5.4). A counterexample inspired by a suggestion of Yang allows us then to prove that composition adjunct contributes in a non-trivial way to the expressiveness of both logics, thus settling a conjecture formulated in [19] about whether this connective could also be eliminated in spatial logics for concurrency. We conclude with a generalization of our results to the π-calculus and to the calculus of Mobile Ambients.

Related Work

Sangiorgi first showed [22] that observation of capabilities in the ambient calculus can be expressed inside spatial logics making use of the $\triangleright$ and $\Diamond$ operators.

This result has since then been generalized to other calculi [4,18]. However, in all such encodings, the use of quantifiers, and references to (some times fresh) names using the revelation connective seems to be essential. From this point of view, our work gives a tighter bound on the level of expressiveness really needed to embed action modalities, since it does not use operators beyond those expected in every pure spatial logic. A related effort addressing minimality was developed by Hirschkoff, aiming at a characterization of π-calculus behavioral equivalences using a logic with composition adjunct but no composition [16].

Adjunct elimination for a static spatial logic was first proved by Lozes in [19], where a counterexample to adjunct elimination in the presence of quantifiers was also presented. However, the particular counterexample given there makes an essential use of name revelation, and thus only applies to calculi with restricted names and related logical connectives. The counterexample presented in this paper is much more general to spatial logics, since it does not rely on such constructs.

Concerning decidability and model-checking of spatial logics, decidability of model-checking for the adjunct-free ambient logic against the replication free calculus was settled by Cardelli and Gordon in [11]. Validity and model-checking of ambient calculus against spatial logics with existential quantifiers was shown undecidable by Charatonik and Talbot [13]. The same authors also extended the results of [11] to logics with constructs for restricted names, and then with Gordon to the finite-control ambient-calculus [12].

Model-checking the π-calculus against full adjunct-free spatial logic with behavioral modalities, hidden and fresh name quantifiers, and recursive operators was shown to be decidable by Caires in [2], where it is also presented an equational characterization of logical equivalence for such logic. Decidability of validity in a static spatial logic for trees with adjunct was first shown by Calcagno, Cardelli and Gordon in [6], building on techniques developed by Calcagno, Yang and O'Hearn in [7]. More recently, Conforti and Ghelli proved that similar results do not hold in spatial logics with operators for restricted names [14].

No results about expressiveness and decidability of dynamic spatial logics so crisp as the ones developed in this paper have been presented elsewhere, in the sense that they apply to a minimal spatial logic for concurrency, and focus on the crucial combination of the composition adjunct with the dynamic modality. The elimination of quantifiers (although not of variables, as we also achieve here) is an important topic of interest in classical logic, related to decidability and complexity issues (*e.g.*, see [1]). However, we believe that our work lies completely out of this scope, as on the contrary we derive undecidability of our logic from the elimination of quantifiers.

1 Preliminaries

In this section, we introduce the process calculus and the spatial logics considered in this work. For the process calculus, we pick a fairly small fragment of CCS.

Definition 1.1 *Assume given an infinite set* A *of* actions*, ranged over by* α, β*.* Processes *are defined by the following grammar:*

$$\begin{array}{rcl} \alpha, \beta, \gamma & \in & \mathsf{A} \\ P, Q, R & ::= & \mathbf{0} \;\Big|\; P \mid P \;\Big|\; \alpha.\, P \end{array}$$

Actions are given in pairs of distinct (co)actions, characterized by the involution $\mathbf{co} : \mathsf{A} \to \mathsf{A}$ sending α into $\overline{\alpha}$, and such that $\overline{\overline{\alpha}} = \alpha$. The relation of *structural congruence* is defined as the least congruence $\equiv$ on processes such that $P \mid \mathbf{0} \equiv P$, $P \mid Q \equiv Q \mid P$, and $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$. Structural congruence represents identity of the static spatial structure of processes. Dynamics of processes is captured by labeled transitions.

Definition 1.2 *Given the set* $\mathsf{L} \triangleq \{\tau\} \cup \mathsf{A}$ *of* labels*, the relation of* labeled transition *is defined by the rules*

$$\alpha.\, P \xrightarrow{\alpha} P \qquad\qquad P \xrightarrow{\ell} P' \Rightarrow P \mid Q \xrightarrow{\ell} P' \mid Q$$

$$P \xrightarrow{\alpha} P', Q \xrightarrow{\overline{\alpha}} Q' \Rightarrow P \mid Q \xrightarrow{\tau} P' \mid Q'$$

Notice that $\xrightarrow{\alpha}$ is closed under $\equiv$, and that $\xrightarrow{\tau}$ corresponds to the usual relation of *reduction*, noted $\longrightarrow$. A few technical notions will be useful. We define the *depth* of a process P (maximal nesting of actions in a process P) as follows:

$$\mathsf{ds}(\mathbf{0}) = 0 \qquad \mathsf{ds}(\alpha.\, P) = 1 + \mathsf{ds}(P) \qquad \mathsf{ds}(P \mid Q) = \max(\mathsf{ds}(P), \mathsf{ds}(Q))$$

For any natural number K, let M_K denote the set of all processes whose depth does not exceed K: $M_K \triangleq \{P \mid \mathsf{ds}(P) \leq K\}$. Notice that $M_\infty \triangleq \bigcup_{k \in \mathbb{N}} M_k$ coincides with the set of all processes. We also define the projection (by truncation) $\pi_k : M_\infty \to M_k$, by induction on k by letting $\pi_0(P) = \mathbf{0}$, $\pi_{k+1}(\mathbf{0}) \triangleq \mathbf{0}$, $\pi_k(P \mid Q) \triangleq \pi_k(P) \mid \pi_k(Q)$, and $\pi_{k+1}(\alpha.\, P) \triangleq \alpha.\, \pi_k(P)$.

Having defined the intended process model, we turn to logics. The basic logic we consider includes the basic spatial operators found in all spatial logics namely: the composition operator $|$, the void operator 0, and the composition adjunct operator $\triangleright$ (guarantee). To these connectives, we add the temporal

$$
\begin{array}{ll}
P, v \models_M \neg \mathcal{A} & \textit{if not } P, v \models_M \mathcal{A} \\
P, v \models_M \mathcal{A} \wedge \mathcal{B} & \textit{if } P, v \models_M \mathcal{A} \textit{ and } P, v \models_M \mathcal{B} \\
P, v \models_M 0 & \textit{if } P \equiv \mathbf{0} \\
P, v \models_M \mathcal{A} \mid \mathcal{B} & \textit{if } \exists Q, R.\ P \equiv Q \mid R \textit{ and } Q, v \models_M \mathcal{A} \textit{ and } R, v \models_M \mathcal{B} \\
P, v \models_M \mathcal{A} \triangleright \mathcal{B} & \textit{if } \forall Q \in M,\ Q, v \models_M \mathcal{A} \textit{ implies } P \mid Q, v \models_M \mathcal{B} \\
P, v \models_M \exists x. \mathcal{A} & \textit{if } \exists \alpha \in \mathsf{A}.\ P, (v\{x \leftarrow \alpha\}) \models_M \mathcal{A} \\
P, v \models_M \Diamond \mathcal{A} & \textit{if } \exists P'.\ P \longrightarrow P' \textit{ and } P', v \models_M \mathcal{A} \\
P, v \models_M \langle x \rangle \mathcal{A} & \textit{if } \exists P'.\ P \xrightarrow{v(x)} P' \textit{ and } P', v \models_M \mathcal{A} \\
P, v \models_M \langle \overline{x} \rangle \mathcal{A} & \textit{if } \exists P'.\ P \xrightarrow{\overline{v(x)}} P' \text{ and } P', v \models_M \mathcal{A}
\end{array}
$$

Fig. 1. Semantics of logical formulas

operator $\Diamond$ (next step), to capture the dynamic behavior of processes. These operators may be considered the core connectives for spatial logics for concurrency. We then consider the extension of the core with modalities for actions (*cf.* Hennessy-Milner logic), and first-order quantifiers ranging over actions.

Definition 1.3 *Given an infinite set* X *of variables,* $(x, y \in \mathsf{X})$ *formulas are given by:*

$$
\begin{array}{rll}
\mathcal{A}, \mathcal{B} ::= & \mathcal{A} \wedge \mathcal{B} \mid \mathcal{A} \mid \mathcal{B} \mid \neg \mathcal{A} \mid \mathcal{A} \triangleright \mathcal{B} \mid 0 \mid \Diamond \mathcal{A} & (\mathcal{L}_{spat}) \\
\mid & \langle x \rangle \mathcal{A} \mid \langle \overline{x} \rangle \mathcal{A} \mid \exists x. \mathcal{A} & (\mathcal{L}_{mod})
\end{array}
$$

We write $\mathcal{L}_{spat}$ for the set of formulas in the pure spatial fragment, and $\mathcal{L}_{mod}$ for the set of all formulas. *Free variables* of formulas are defined as usual; we say a formula is *closed* if it has no free variables.

Semantics is defined by a relation of satisfaction as shown in Fig. 1. *Satisfaction* is expressed by the judgment form $P, v \models_M \mathcal{A}$ where P is a process, M is a set of processes, $\mathcal{A}$ a formula, and v is a valuation for the free variables of $\mathcal{A}$. A *valuation* is a mapping from a finite subset of X to A. For any valuation v, we write $v\{x \leftarrow \alpha\}$ for the valuation v' such that $v'(x) = \alpha$, and $v'(y) = v(y)$ if $y \neq x$. By $\emptyset$ we denote the empty valuation. Notice that this definition of satisfaction matches the usual one except for the presence of the index M, which specifies the range of quantification for interpreting the adjunct (see the clause for $\triangleright$ in Fig. 1). This generalization is only a convenience for our technical development; it is clear that $\models_{M_\infty}$ corresponds to the standard

non-relativized relation of satisfaction. So, we abbreviate $P, v \models_{M_\infty} \mathcal{A}$ by $P, v \models \mathcal{A}$, moreover, when the formula $\mathcal{A}$ is closed we abbreviate $P, \emptyset \models_M \mathcal{A}$ by $P \models_M \mathcal{A}$. By default, the set of processes M is M_∞, so that we may also abbreviate $P \models_{M_\infty} \mathcal{A}$ by $P \models \mathcal{A}$.

An *action permutation* is a bijection $\sigma : \mathsf{A} \to \mathsf{A}$ such that $\sigma(\overline{\alpha}) = \overline{\sigma(\alpha)}$ and the *domain* $D(\sigma) \triangleq \{\alpha \mid \sigma(\alpha) \neq \alpha\}$ is finite. We write $\{\alpha \leftrightarrow \beta\}$ for the action permutation that swaps α and β (and thus $\overline{\alpha}$ with $\overline{\beta}$).

Definition 1.4 (Action of permutations) *Action permutations act on processes as follows:*

$$\sigma(\mathbf{0}) \triangleq \mathbf{0} \quad \sigma(\gamma.\, P) \triangleq \sigma(\gamma).\, \sigma(P) \quad \sigma(P \mid Q) \triangleq \sigma(P) \mid \sigma(Q)$$

By $\sigma(v)$ we denote the valuation such that $(\sigma(v))(x) = \sigma(v(x))$ for all $x \in \mathsf{X}$.

Definition 1.5 *Let $\equiv_s$ be the binary relation on processes defined by $P \equiv_s Q$ if and only if there is an action permutation σ such that $P \equiv \sigma(Q)$.*

Satisfaction verifies the fundamental property of equivariance, which in our present setting is formulated as follows.

Proposition 1.6 (Equivariance) *Let $P, v \models_M \mathcal{A}$. For every action permutation σ, if $P \equiv \sigma(Q)$ then $Q, \sigma(v) \models_M \mathcal{A}$.*

Proof: Induction on the structure of the formula $\mathcal{A}$. □

We frequently refer to the logical equivalence relation $=_{\mathrm{L}}$ induced on processes by the logic L (where L is one of the logics $\mathcal{L}_{spat}$ or $\mathcal{L}_{mod}$). The relation $=_{\mathrm{L}}$ is defined in the standard way by asserting $P =_{\mathrm{L}} Q$ whenever for all closed formulas $\mathcal{A}$, we have $P, \emptyset \models \mathcal{A}$ if and only if $Q, \emptyset \models \mathcal{A}$.

Besides the basic stock of primitive connectives, we also use a few derived ones: we list their definition and formal meaning in Fig. 2. Notice in particular the definition of equality and symmetry of actions, the remaining operators are fairly standard (see [4]). The following technical notions of formula width and depth are useful.

Definition 1.7 (Formula width) *By $w(\mathcal{A})$ we denote the maximal level of nesting of composition connectives $- \mid -$ in the formula $\mathcal{A}$, defined by*

$$w(0) \triangleq 0$$

$$w(\mathcal{A} \wedge \mathcal{B}) = w(\mathcal{A} \triangleright \mathcal{B}) \triangleq \max\left(w(\mathcal{A}), w(\mathcal{B})\right)$$

$$w(\mathcal{A} \mid \mathcal{B}) \triangleq 1 + \max\left(w(\mathcal{A}), w(\mathcal{B})\right)$$

$$w(\Diamond\mathcal{A}) = w(\neg\,\mathcal{A}) = w(\exists x.\, \mathcal{A}) = w(\langle x \rangle \mathcal{A}) = w(\langle \overline{x} \rangle \mathcal{A}) \triangleq w(\mathcal{A})$$

$$
\begin{array}{lll}
\top & \triangleq 0 \vee \neg\, 0 & \bot \quad \triangleq \neg\, \top \\
\mathcal{A} \vee \mathcal{B} & \triangleq \neg\,(\neg\, \mathcal{A} \wedge \neg\, \mathcal{B}) & \mathcal{A} \Rightarrow \mathcal{B} \triangleq \neg\, \mathcal{A} \vee \mathcal{B} \\
\forall x.\, \mathcal{A} & \triangleq \neg\, \exists x.\, \neg\, \mathcal{A} & \mathcal{A} \| \mathcal{B} \quad \triangleq \neg\,(\neg\, \mathcal{A} \mid \neg\, \mathcal{B}) \\
\mathcal{A} \blacktriangleright \mathcal{B} & \triangleq \neg\,(\mathcal{A} \triangleright \neg\, \mathcal{B}) & \mathcal{A}^{\forall} \quad \triangleq \mathcal{A} \| \bot \\
\mathcal{A}^{\exists} & \triangleq \mathcal{A} \mid \top & \mathcal{A}^{\vdash} \quad \triangleq (\neg\, \mathcal{A}) \triangleright \bot \\
x = y & \triangleq \Big((\langle x \rangle 0 \mid \langle \overline{y} \rangle 0) \Rightarrow \Diamond 0\Big)^{\vdash} & x = \overline{y} \quad \triangleq \Big((\langle x \rangle 0 \mid \langle y \rangle 0) \Rightarrow \Diamond 0\Big)^{\vdash}
\end{array}
$$

$$
\begin{array}{lll}
P, v \models_M \top & \textit{if} & \textit{always} \\
P, v \models_M \bot & \textit{if} & \textit{never} \\
P, v \models_M \mathcal{A} \vee \mathcal{B} & \textit{if} & P, v \models_M \mathcal{A} \text{ or } P, v \models_M \mathcal{B} \\
P, v \models_M \mathcal{A} \Rightarrow \mathcal{B} & \textit{if} & P, v \models_M \mathcal{A} \textit{ implies } P, v \models_M \mathcal{B} \\
P, v \models_M \forall x.\, \mathcal{A} & \textit{if} & \forall \alpha \in \mathsf{A}.\ P, (v\{x \leftarrow \alpha\}) \models_M \mathcal{A} \\
P, v \models_M \mathcal{A} \| \mathcal{B} & \textit{if} & \forall Q, R.\ P \equiv Q \mid R \textit{ implies } Q, v \models_M \mathcal{A} \text{ or } R, v \models_M \mathcal{B} \\
P, v \models_M \mathcal{A} \blacktriangleright \mathcal{B} & \textit{if} & \exists Q \in M.\ Q \models_M \mathcal{A} \textit{ and } P \mid Q \models_M \mathcal{B} \\
P, v \models_M \mathcal{A}^{\forall} & \textit{if} & \forall Q, R.\ P \equiv Q \mid R \textit{ implies } Q, v \models_M \mathcal{A} \\
P, v \models_M \mathcal{A}^{\exists} & \textit{if} & \exists Q, R.\ P \equiv Q \mid R \textit{ and } Q, v \models_M \mathcal{A} \\
P, v \models_M \mathcal{A}^{\vdash} & \textit{if} & \forall Q \in M.\ Q, v \models_M \mathcal{A} \\
P, v \models_M x = y & \textit{if} & v(x) = v(y) \quad (\textit{for all } M \textit{ such that } M_1 \subseteq M) \\
P, v \models_M x = \overline{y} & \textit{if} & v(x) = \overline{v(y)} \quad (\textit{for all } M \textit{ such that } M_1 \subseteq M)
\end{array}
$$

Fig. 2. Definition and semantics of derived operators.

Definition 1.8 (Formula depth) *We denote by* $\mathsf{ds}(\mathcal{A})$ *the maximal nesting of dynamic modalities in the formula* $\mathcal{A}$*, defined by*

$$
\begin{array}{l}
\mathsf{ds}(0) = 0 \\
\mathsf{ds}(\mathcal{A} \wedge \mathcal{B}) = \mathsf{ds}(\mathcal{A} \mid \mathcal{B}) = \mathsf{ds}(\mathcal{A} \triangleright \mathcal{B}) \triangleq \max\left(\mathsf{ds}(\mathcal{A}), \mathsf{ds}(\mathcal{B})\right) \\
\mathsf{ds}(\Diamond \mathcal{A}) = \mathsf{ds}(\langle x \rangle \mathcal{A}) = \mathsf{ds}(\langle \overline{x} \rangle \mathcal{A}) \triangleq \mathsf{ds}(\mathcal{A}) + 1 \\
\mathsf{ds}(\neg\, \mathcal{A}) = \mathsf{ds}(\exists x.\, \mathcal{A}) \triangleq \mathsf{ds}(\mathcal{A})
\end{array}
$$

It is easy to see that a formula $\mathcal{A}$ cannot observe any property of a process that can only manifests itself after a temporal horizon of $\mathsf{ds}(\mathcal{A})$ dynamic steps. As

a consequence, the restriction to M_k of the denotation of a formula of depth k completely characterizes such denotation, in the precise sense of:

Proposition 1.9 (Depth finiteness) *For all formulas $\mathcal{A} \in \mathcal{L}_{mod}$, for all $k > \mathsf{ds}(\mathcal{A})$, for all processes P, and for all valuations v,*

$P, v \models_{M_\infty} \mathcal{A}$ *if and only if* $\pi_k(P), v \models_{M_\infty} \mathcal{A}$ *if and only if* $\pi_k(P), v \models_{M_k} \mathcal{A}$.

This result is a consequence of the following technical lemma:

Lemma 1.10 (Reduction mimick) *Let $\mathcal{R}$ be either $\xrightarrow{\alpha}$ or $\longrightarrow$. Then for any $k > 0$, for any processes P, P'*

- *if $P \mathcal{R} P'$, then there is P'' such that $\pi_k(P) \mathcal{R} P''$ and $\pi_{k-1}(P') \equiv \pi_{k-1}(P'')$;*
- *if $\pi_k(P) \mathcal{R} P'$, then there is P'' such that $P \mathcal{R} P''$ and $\pi_{k-1}(P') \equiv \pi_{k-1}(P'')$.*

Proof: Case $\mathcal{R} = \xrightarrow{\alpha}$: assume first that $P \xrightarrow{\alpha} P'$. Then there are P_1, P_2 such that $P \equiv \alpha.P_1 \mid P_2$ and $P' \equiv P_1 \mid P_2$. We set $P'' = \pi_{k-1}(P_1) \mid \pi_k P_2$. Since $\pi_k(P) \equiv \alpha.\pi_{k-1}(P_1) \mid \pi_k(P_2)$, we have $\pi_k(P) \xrightarrow{\alpha} P''$. Moreover, $\pi_{k-1)(P'') \equiv \pi_{k-1}(P_1) | \pi_{k-1}(P_2} \equiv \pi_{k-1}(P')$. Assume now that $\pi_k(P) \xrightarrow{\alpha} P'$. Then there are P_1, P_2 such that $\pi_k(P) \equiv \alpha.\pi_{k-1}(P_1) \mid \pi_k(P_2)$ and $P \equiv \alpha.P_1 \mid P_2$. We set $P'' \equiv P_1 \mid P_2$, which gives both $P \xrightarrow{\alpha} P''$ and $\pi_{k-1}(P') \equiv \pi_{k-1}(P'')$.
Case of $\mathcal{R} = \longrightarrow$: assume first that $P \longrightarrow P'$. Then there are $\alpha_1, \alpha_2, P_1, P_2, P_1', P_2'$ such that $\alpha_1 = \overline{\alpha_2}$, $P \equiv P_1 \mid P_2$, $P' \equiv P_1' \mid P_2'$, and $P_i \xrightarrow{\alpha_i} P_i'$. By the previous result, there are P_i'' such that $\pi_k(P_i) \xrightarrow{\alpha_i} P_i''$ and $\pi_{k-1}(P_i') \equiv \pi_{k-1}(P_i'')$. Set $P'' = P_1'' \mid P_2''$, then $\pi_k(P) \longrightarrow P''$ and $\pi_{k-1}(P'') \equiv \pi_{k-1}(P')$. The converse implication is proved exactly in the same way. □

Proof:(of Proposition 1.9) The second equivalence is established by a straightforward induction on $\mathcal{A}$ using the first equivalence. We sketch the proof by induction on $\mathcal{A}$ for the first equivalence:
Case of $(|)$: if $P, v \models \mathcal{A}_1 \mid \mathcal{A}_2$, then there are P_1, P_2 such that $P \equiv P_1 \mid P_2$ and $P_i, v \models \mathcal{A}_i$. Then $\mathsf{ds}(\mathcal{A}_1) = \mathsf{ds}(A_2) = \mathsf{ds}(A) < k$, so we may apply induction hypothesis and $\pi_k(P_i), v \models \mathcal{A}_i$. Since $\pi_k(P) \equiv \pi_k(P_1) \mid \pi_k(P_2)$, $\pi_k(P), v \models \mathcal{A}_1 \mid \mathcal{A}_2$. Conversely, if $\pi_k(P), v \models \mathcal{A}_1 \mid \mathcal{A}_2$, then there are P_1, P_2 such that $P \equiv P_1 \mid P_2$, $\pi_k(P) \equiv \pi_k(P_1) \mid \pi_k(P_2)$, and $\pi_k(P_i), v \models \mathcal{A}_i$. By induction hypothesis, $P_i, v \models \mathcal{A}_i$, so $P \models, v \models \mathcal{A}_1 \mid \mathcal{A}_2$.
Case of $(\Diamond)$: if $P, v \models \Diamond\mathcal{A}$, then there is P' such that $P \longrightarrow P'$ and $P', v \models \mathcal{A}$. Then $\mathsf{ds}(\mathcal{A}) = \mathsf{ds}(\Diamond\mathcal{A}) - 1$, so $\mathsf{ds}(\mathcal{A}) < k - 1$ and by induction hypothesis $\pi_{k-1}(P'), v \models \mathcal{A}$. Let P'' be the process obtained mimicking the reduction $P \longrightarrow P'$ starting from $\pi_k(P)$, so that $\pi_k(P) \longrightarrow P''$ and $\pi_{k-1}(P'') \equiv \pi_{k-1}(P')$. Now by induction hypothesis $P', v \models \mathcal{A}$ implies $\pi_{k-1}(P'), v \models \mathcal{A}$, which is $\pi_{k-1}(P''), v \models \mathcal{A}$, which implies $P'', v \models \mathcal{A}$. This finally shows that $\pi_k(P), v \models \Diamond\mathcal{A}$. Conversely, if $\pi_k(P), v \models \Diamond\mathcal{A}$, then there is P'' such that $\pi_k(P) \longrightarrow P''$ and $P'', v \models \mathcal{A}$. Mimicking the same reduction, we have P'

such that $P \longrightarrow P'$ and $\pi_{k-1}(P') \equiv \pi_{k-1}(P'')$. Now by induction hypothesis, $P'', v \models \mathcal{A}$ implies $\pi_{k-1}(P''), v \models \mathcal{A}$, which is $\pi_{k-1}(P'), v \models \mathcal{A}$, which implies $P', v \models \mathcal{A}$. This finally shows that $P, v \models \Diamond\mathcal{A}$. □

The process $P_1 \mid \ldots \mid P_n$ is abbreviated by $\prod_{i=1\ldots n} P_i$, and by P^n we denote the process $\prod_{i=1\ldots n} P$. In the same way, we abbreviate the formula $\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ by $\prod_{i=1\ldots n} \mathcal{A}_i$, and $\mathcal{A}^n$ then denotes $\prod_{i=1\ldots n} \mathcal{A}$.

2 Elimination of quantifiers and action modalities

In this section, we prove the main result of the paper, stating that the spatial logic $\mathcal{L}_{mod}$, which contains quantifiers, variables and action modalities, can be embedded into the core spatial logic $\mathcal{L}_{spat}$, which does not seem to contain related constructs. The embedding will be defined by a recursive map $[\![-]\!]_K$ that assigns to each $\mathcal{L}_{mod}$ formula $\mathcal{A}$ and natural number K a formula of $[\![\mathcal{A}]\!]_K$ of $\mathcal{L}_{spat}$. This result is expressed in a precise way the statement of the following theorem.

Theorem 2.1 *For any natural number K there is a recursively defined map $[\![-]\!]_K : \mathcal{L}_{mod} \to \mathcal{L}_{spat}$ such that for any closed formula $\mathcal{A} \in \mathcal{L}_{mod}$ with $\mathsf{ds}(\mathcal{A}) < K$, we have:*

$$\forall P.\ \ P \models \mathcal{A} \ \ \textit{if and only if} \ \ \pi_K(P) \models [\![\mathcal{A}]\!]_K$$

It is important to note that this result does not state that $\mathcal{L}_{spat}$ and $\mathcal{L}_{mod}$ have exactly the same expressiveness, at least in the strict technical sense of "same expressiveness". However, we must remark that the denotation of any formula $\mathcal{A}$ is completely characterized by its denotation on some subset of the models M_k, following the depth finiteness property stated in Proposition 1.9. Hence, the denotation of $[\![\mathcal{A}]\!]_K$ completely characterizes the denotation of $\mathcal{A}$; this close correspondence will be enough to show the undecidability and separation power of $\mathcal{L}_{spat}$ (Theorem 3.3 and Theorem 5.4), and the independence of the composition adjunct from the remaining logical connectives (Theorem 4.4).

The proof of Theorem 2.1 requires considerable build up. In particular, we need to define $\mathcal{L}_{spat}$ formulas to characterize processes of several quite specific structural forms, to be used for various purposes in our encoding of $\mathcal{A}$ into $[\![\mathcal{A}]\!]_K$. This exercise turns out to be quite interesting: by going through it one gets a better understanding about what can be expressed in $\mathcal{L}_{spat}$, in a not obvious and certainly not trivial way.

We seek a reduction of a satisfaction judgment $P, v \models_{M_K} \mathcal{A}$, where $\mathcal{A}$ is any $\mathcal{L}_{mod}$ formula, into a satisfaction judgment for a formula $[\![\mathcal{A}]\!]_K$ of $\mathcal{L}_{spat}$ that

neither contains quantifiers, nor action modalities (and thus no occurrences of variables whatsoever). The key idea is to represent the valuation v appearing in $P, v \models_{M_K} \mathcal{A}$ by a special process $\mathbf{val}(e, \nu, w)_K$, to be composed with the process P being tested for satisfaction. Here, the data e, ν and w are a technically more convenient representation of the valuation v, further explained below. With this device, the addition to a valuation v of a (new) entry $\{x \leftarrow \alpha\}$ for the variable x as introduced in the satisfaction clause for $\exists x.\, \mathcal{A}$

$$P, v \models_M \exists x.\, \mathcal{A} \quad \textit{if} \;\; \exists \alpha \in \mathsf{A}.\;\; P, (v\{x \leftarrow \alpha\}) \;\models_M\; \mathcal{A}$$

can be mimicked in the encoding $[\![\exists x.\, \mathcal{A}]\!]_K$ by the addition to the context of a certain process $\mathbf{val}(e, \nu, w)_K$, using the existential adjunct $- \blacktriangleright -$. The intent is to obtain a correspondence property of the form

$$P, v \models_{M_K} \mathcal{A} \textit{ if and only if } P \mid \mathbf{val}(e, \nu, w)_K \models_{M_\infty} [\![\mathcal{A}]\!]_K$$

With relation to a valuation v and the process $\mathbf{val}(e, \nu, w)_K$, ν and e are maps $e : \mathsf{X} \to \mathbb{N}$ and $\nu : \mathbb{N} \to \mathsf{A}$, respectively called the *environment* and the *naming*. These maps express a decomposition of the valuation v so that $\nu \circ e = v$. The general idea is that for a valuation such as $v = \{x_1 \to \alpha_1, \ldots, x_n \to \alpha_n\}$, we will have $e = \{x_1 \to 1, \ldots, x_n \to n\}$ and $\nu = \{1 \to \alpha_1, \ldots, n \to \alpha_n\}$.

The encoding of a valuation into the process $\mathbf{val}(e, \nu, w)_K$ uses the notion of *row* process. A row process $\mathbf{row}(n, \alpha)$ (on the action α) is a thread of the form $\alpha.\, \alpha \ldots \alpha.\, \mathbf{0}$, where the action α occurs precisely n times (so that $\mathsf{ds}(\mathbf{row}(n, \alpha)) = n$). This process is interesting since it can be characterized by a $\mathcal{L}_{spat}$ formula. We will use row processes on α to represent each binding of a valuation: bindings relative to different variables x_i will be represented by rows of different length $e(i)$. In fact, we will not use a single row for each binding, but a collection of 2^w rows: this is related to the fact that we need to have multiple copies of the valuation to deal with the interpretation of $\mid$.

Notice that, by imposing a bound K on the depth of the process P one considers, and encoding valuations by rows whose depth is strictly greater that K, we can easily separate the valuation part from the process part that represents the "real" model, in the "soup" $P \mid \mathbf{val}(e, \nu, w)_K$.

We now proceed to define several interesting sets of processes by means of appropriate logical formulas. We start by introducing, for each natural number $n > 0$, $\mathcal{L}_{spat}$ formulas $\mathsf{Thread}(n)$ whose models are precisely the sequential threads with the given number n of actions, in the way we also define the derived modality $?.\, \mathcal{A}$.

$$1 \quad \triangleq \neg 0 \;\wedge\; (0 \mid\mid 0) \qquad \mathsf{Thread}(1) \quad \triangleq 1 \;\wedge\; (1 \blacktriangleright \Diamond 0)$$

$$?.\, \mathcal{A} \triangleq 1 \;\wedge\; (\mathsf{Thread}(1) \blacktriangleright \Diamond \mathcal{A}) \quad \mathsf{Thread}(n+1) \triangleq ?.\, \mathsf{Thread}(n)$$

We have

Lemma 2.2 *For all processes P, and sets of processes M such that $M_1 \subseteq M$*

$$\begin{array}{lll} P \models_M 1 & \textit{iff} & \exists \alpha \in \mathsf{A}.\, \exists Q.\, P \equiv \alpha.\, Q \\ P \models_M ?.\, \mathcal{A} & \textit{iff} & \exists \alpha \in \mathsf{A}.\, \exists Q.\ P \equiv \alpha.\, Q \textit{ and } Q \models \mathcal{A} \\ P \models_M \mathsf{Thread}(1) & \textit{iff} & \exists \alpha \in \mathsf{A}.\ P \equiv \alpha.\, \mathbf{0} \\ P \models_M \mathsf{Thread}(k) & \textit{iff} & \exists \alpha_1 \in \mathsf{A}. \ldots \exists \alpha_k \in \mathsf{A}.\ P \equiv \alpha_1. \cdots . \alpha_k.\, \mathbf{0} \end{array}$$

We now define (for each $k \geq 0$) a formula $\mathcal{M}_k$ that characterizes the model M_k, that is, such that we have $P \models \mathcal{M}_k$ if and only if $P \in M_k$.

$$\mathcal{M}_0 \triangleq 0 \qquad \mathcal{M}_{k+1} \triangleq (1 \Rightarrow ?.\, \mathcal{M}_k)^{\forall}$$

Using the $\Diamond$ modality as an equality tester, we can then define a formula $\mathsf{Equals}(k)$ that is satisfied by the processes which belong to M_k, and are compositions of guarded processes all with the *same* first action. From $\mathsf{Equals}(k)$ we can then specify row processes as shown below

$$\begin{array}{ll} \mathsf{Equals}(k) & \triangleq \mathcal{M}_k \ \wedge \ \Big(\mathsf{Thread}(k+1) \blacktriangleright \Big((\mathsf{Thread}(k+1) \mid 1) \Rightarrow \Diamond \top\Big)^{\forall}\Big) \\ \mathsf{RowCol}(0) & \triangleq 0 \\ \mathsf{RowCol}(n+1) & \triangleq \Big(\mathsf{Thread}(n+1) \mid \mathsf{Equals}(1)\Big) \wedge \Diamond \mathsf{RowCol}(n) \\ \mathsf{Row}(n) & \triangleq \mathsf{Thread}(n) \wedge (\top \blacktriangleright \mathsf{RowCol}(n)) \end{array}$$

We now prove

Lemma 2.3 *For all k, and process P, we have:*

$$\begin{array}{lll} P \models \mathcal{M}_k & \textit{iff} & P \in M_k \\ P \models \mathsf{Equals}(k) & \textit{iff} & P \in M_k \textit{ and } \exists \alpha \in \mathsf{A}.\, \exists n \geq 0. \\ & & \quad \exists P_1, \ldots, P_n.\ \ P \ \equiv \ \alpha.\, P_1 \mid \ldots \mid \alpha.\, P_n \\ P \models \mathsf{Row}(k) & \textit{iff} & \exists \alpha \in \mathsf{A}.\ P \equiv \mathbf{row}(k, \alpha) \end{array}$$

Building on these ingredients, we can now introduce our encoding of a valuation v into the process $\mathbf{val}(e, \nu, w)_K$. Given a valuation $v = \{x_1 \to \alpha_1, \ldots, x_n \to \alpha_n\}$ with environment $e = \{x_1 \to 1, \ldots, x_n \to n\}$ and naming $\nu = \{1 \to \alpha_1, \ldots, n \to \alpha_n\}$, we define

$$\mathbf{val}(e, \nu, w)_K \ \triangleq \ \textstyle\prod_{i=1 \ldots |e|} \mathbf{row}(K+i, \nu(i))^{2^w}$$

The parameter w specifies the number of rows of the appropriate length that are needed to represent the environment entry for a variable x, and is related to the number of occurrences of the $-\mid-$ connective in the source formula. Since interpreting $-\mid-$ also splits the (encoding of the) valuation, we have to provide enough copies (2^w, where w is related to $w(\mathcal{A})$). We can also verify the following important fact

Lemma 2.4 *Let v, v' and v'' be valuations with the same domain, with decompositions (ν, e), (ν', e) and (ν'', e). Then, we have*

$$\mathbf{val}(e, \nu, w+1)_K \equiv \mathbf{val}(e, \nu', w)_K \mid \mathbf{val}(e, \nu'', w)_K \text{ if and only if } \nu = \nu' = \nu''$$

Notice moreover that we can always filter out any undesirable interference of the process $\mathbf{val}(e, \nu, w)_K$ with the parallel process P, since for any labeled-transition reduct Q of $\mathbf{val}(e, \nu, w)_K$, Q is not the proper encoding of any valuation, since it does not have the right number of rows for each depth. Using already defined properties, we define for each K, e and w the formulas

$$\begin{array}{rcl} \mathsf{Val}(e, w)_K & \triangleq & \prod_{i=1\ldots|e|} \left(\mathsf{Row}(K+i)^{2^w} \wedge \mathsf{Equals}(K+i)\right) \\ \mathsf{ProcVal}(e, w)_K & \triangleq & \mathcal{M}_K \mid \mathsf{Val}(e, w)_K \end{array}$$

We have the following characterization

Lemma 2.5 *For any process P, environment e and naturals $K, w \geq 1$*

$$\begin{array}{lll} P \models \mathsf{Val}(e, w)_K & \text{iff} & \exists \nu.\ P \equiv \mathbf{val}(e, \nu, w)_K \\ P \models \mathsf{ProcVal}(e, w)_K & \text{iff} & \exists Q \in M_K, \exists \nu.\ P \equiv Q \mid \mathbf{val}(e, \nu, w)_K \end{array}$$

Hence, the formula $\mathsf{ProcVal}(e, w)_K$ specifies a pair process-valuation, where the process belongs to M_K. Now we introduce formulas for querying specific entries of the (encoding of the) valuation: selection of the action α associated to the variable x is achieved by selecting the group of row processes of depth $e(x)$.

$$\begin{array}{rcl} \mathsf{XRow}(x, e)_K & \triangleq & \mathsf{Row}(K + e(x)) \\ \mathsf{UsedXRow}(x, e)_K & \triangleq & \mathsf{Row}(K + e(x) - 1) \\ \mathsf{EnvX}(x, e, w)_K & \triangleq & \mathsf{Equals}(K + \mid e \mid) \wedge (\mathsf{XRow}(x, e)_K)^{2^w} \end{array}$$

The formula $\mathsf{XRow}(x, e)_K$ matches one of the rows that represents the environment entry of the variable x. Then, the formula $\mathsf{UsedXRow}(x, e)_K$ can be used to check that such a row has lost exactly one action prefix (after a reduction

$$
\begin{array}{lcl}
[\![\mathcal{A} \wedge \mathcal{B}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge [\![\mathcal{A}]\!]_{(e,w)} \ \wedge \ [\![\mathcal{B}]\!]_{(e,w)} \\
[\![\neg\, \mathcal{A}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge \neg\, [\![\mathcal{A}]\!]_{(e,w)} \\
[\![0]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge \mathsf{Val}(e,w)_K \\
[\![\mathcal{A} \mid \mathcal{B}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge ([\![\mathcal{A}]\!]_{(e,w-1)} \ \big|\ [\![\mathcal{B}]\!]_{(e,w-1)}) \\
[\![\mathcal{A} \triangleright \mathcal{B}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge \\
 & & \quad \Big([\![\mathcal{A}]\!]_{(e,w)} \ \triangleright \ \Big(\mathsf{ProcVal}(e,w+1)_K \Rightarrow [\![\mathcal{B}]\!]_{(e,w+1)}\Big)\Big) \\
[\![\Diamond \mathcal{A}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge \Diamond [\![\mathcal{A}]\!]_{(e,w)} \\
[\![\exists x.\, \mathcal{A}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge \Big(\mathsf{EnvX}(x,e',w)_K \blacktriangleright [\![\mathcal{A}]\!]_{(e',w)}\Big) \\
 & & \quad \text{where } e' = e\{x\leftarrow \mid e \mid +1\} \\
[\![\langle x \rangle \mathcal{A}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge \\
 & & \quad \mathsf{Test}(e)_K \blacktriangleright \Big((\mathsf{TestMatchesX}(x,e,w)_K \mid \top) \ \wedge \\
 & & \qquad \Diamond(\mathsf{UsedTest}(e)_K \mid [\![\mathcal{A}]\!]_{(e,w)})\Big) \\
[\![\langle \overline{x} \rangle \mathcal{A}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K \ \wedge \Diamond\Big(\mathsf{UsedXRow}(x,e)_K \ \mid \\
 & & \qquad (\mathsf{XRow}(x,e)_K \blacktriangleright [\![\mathcal{A}]\!]_{(e,w)})\Big)
\end{array}
$$

Fig. 3. Encoding of $\mathcal{L}_{mod}$ into $\mathcal{L}_{spat}$.

step takes place). It is easy to see that the formula $\mathsf{EnvX}(x,e,w)_K$ can be used to match the 2^w rows that encode the environment entry for the variable x.

The action modalities $\langle x \rangle \mathcal{A}$ and $\langle \overline{x} \rangle \mathcal{A}$ are interpreted in the encoding by formulas that detect interactions between the process P being tested and the valuation process $\mathbf{val}(e,\nu,w)_K$, using the $\Diamond$ operator. To encode the modality $\langle x \rangle \mathcal{A}$ we need to check for the presence of the complementary of the action $v(x)$. To this end, we specify a row longer than any other (with $\mathsf{Test}(e)$), and then check (using $\Diamond$) that it may react with some row of depth $e(x)$ (with $\mathsf{UsedTest}(e)$): this means that the action assigned to $e(x)$ is complementary of the action on the test row specified by $\mathsf{Test}(e)$. Let then:

$$
\begin{array}{rcl}
\mathsf{Test}(e)_K & \triangleq & \mathsf{Row}(\mid e \mid +K+2) \\
\mathsf{UsedTest}(e)_K & \triangleq & \mathsf{Row}(\mid e \mid +K+1) \\
\mathsf{TestMatchesX}(x,e,w)_K & \triangleq & (\mathsf{Test}(e)_K \mid \mathsf{EnvX}(x,e,w)_K) \ \wedge \ \Diamond\top
\end{array}
$$

We are now ready to present our encoding of formulas of $\mathcal{L}_{mod}$ into formulas of $\mathcal{L}_{spat}$.

Definition 2.6 *Let $\mathcal{A} \in \mathcal{L}_{mod}$ be a formula, e an environment mapping the free variables of $\mathcal{A}$, and w, K be integers such that $w > w(\mathcal{A})$, and $K > 0$. Then, the formula $[\![\mathcal{A}]\!]_{(e,w)} \in \mathcal{L}_{spat}$ is inductively defined as shown in Fig. 3.*

We may note on the encoding of Fig.3 that the encoding is quite straightforward for the connectives of $\mathcal{L}_{spat}$, since they are basically expressed by themselves. Some attention is required only in taking into account the valuation process, and the way its width is changed in the encoding of $|$ and $\triangleright$. For the encoding of connectives specific to $\mathcal{L}_{mod}$, we require some more elaborated machinery. Modalities are encoded by stimulating an interaction between the process P and the valuation process, and an existential quantifier is expressed by an extension of the valuation process, using the $\triangleright$ connective.

The remaining of this section is devoted to the proof of the main Theorem 2.1, that follows from Lemmas 2.2, 2.3, 2.5, and the following general result:

Lemma 2.7 (Correctness of the encoding) *For all processes P, all formulas $\mathcal{A} \in \mathcal{L}_{mod}$, all environments e declaring the free variables of $\mathcal{A}$, all integers $w > w(A)$, and all $K > 0$ we have:*

$$P, \emptyset \models_{M_\infty} [\![\mathcal{A}]\!]_{(e,w)} \quad \textit{if and only if} \quad \exists Q \in M_K, \exists \nu. \begin{cases} P \equiv Q \mid \mathbf{val}(e, \nu, w)_K \\ Q, \nu \circ e \models_{M_K} \mathcal{A} \end{cases}$$

Proof: By induction on $\mathcal{A}$.

- (Cases of) $\mathcal{A} = \mathcal{A}_1 \wedge \mathcal{A}_2, \neg\, \mathcal{A}_1, 0$ straightforward.
- (Case of $\mathcal{A} = \mathcal{A}_a \mid \mathcal{A}_b$) Assume first $P, \emptyset \models_{M_\infty} [\![A]\!]_{(e,w)}$. By Lemma 2.5, there is $P_1 \in M_K$ and ν such that $P \equiv P_1 \mid \mathbf{val}(e, \nu, w)_K$. Moreover, there is a splitting $P_1 \mid \mathbf{val}(e, \nu, w)_K \equiv P_a \mid P_b$ with $P_\epsilon, \emptyset \models_{M_\infty} \mathcal{A}_\epsilon$. By induction hypothesis, each P_ϵ contains a $\mathbf{val}(e, \nu, w-1)_K$. Due to the depth of the rows, P_1 does not contribute to that, so $P_\epsilon \equiv P_{1,\epsilon} \mid \mathbf{val}(e, \nu, w-1)_K$ with $P_1 \equiv P_{1,a} \mid P_{1,b}$. By induction hypothesis, $P_{1,\epsilon}, \nu \circ e \models_{M_K} \mathcal{A}_\epsilon$, hence the result. Conversely, if $P \equiv P_1 \mid \mathbf{val}(e, \nu, w)_K$ with $P_1, \nu \circ e \models_{M_K} \mathcal{A}$, there is $P_{1,a}, P_{1,b}$ such that $P \equiv P_{1,a} \mid P_{1,b}$ and $P_{1,\epsilon}, \nu \circ e \models_{M_K} \mathcal{A}_\epsilon$, hence by induction hypothesis $P_{1,\epsilon} \models_{M_\infty} [\![\mathcal{A}_\epsilon]\!](e, w)$ and $P \equiv \left(P_{1,a} \mid \mathbf{val}(e, w-1, \right)_K) \mid \left(P_{1,b} \mid \mathbf{val}(e, w-1, \right)_K) \models_{M_\infty} [\![\mathcal{A}]\!]_{(e,w)}$.
- (Case of $\mathcal{A} = \mathcal{A}_1 \triangleright \mathcal{A}_2$) Assume first $P, \emptyset \models_{M_\infty} [\![\mathcal{A}]\!]_{(e,w)}$. By Lemma 2.5, there is $P_1 \in M_K$ and ν such that $P \equiv P_1 \mid \mathbf{val}(e, \nu, w)_K$. To prove that $P_1, \nu \circ e \models_{M_K} \mathcal{A}_1 \triangleright \mathcal{A}_2$, we pick some $Q \in M_K$ such that $Q, \nu \circ e \models_{M_K} \mathcal{A}_1$. Then by induction hypothesis $Q \mid \mathbf{val}(e, \nu, w)_K \models_{M_\infty} [\![\mathcal{A}_1]\!]_{(e,w)}$, and $P \mid Q \mid \mathbf{val}(e, \nu, w)_K \models_{M_\infty} \mathsf{ProcVal}(e, w+1)_K$, so $P \mid Q \mid \mathbf{val}(e, \nu, w)_K \models_{M_\infty} [\![\mathcal{A}_2]\!]_{(e,w+1)}$. By induction hypothesis, we have $P \mid Q \mid \mathbf{val}(e, \nu, w)_K \equiv R_1 \mid \mathbf{val}(e, \nu', w)_K$ with $R_1, \nu' \circ e \models_{M_K} \mathcal{A}_2$ Due to the depth of the rows in

$\mathbf{val}(e, \nu', w)_K$, one has necessarily $\nu = \nu'$ and $R_1 \equiv P_1 \mid Q$, hence the result. Assume now that $P \equiv P_1, \nu \circ e \models_{M_K} \mathcal{A}_1 \triangleright \mathcal{A}_2$. To prove that $P, \emptyset \models_{M_\infty} [\![\mathcal{A}_1 \triangleright \mathcal{A}_2]\!]_{(e,w)}$, we take $Q \in M_\infty$ such that $Q \models_{M_\infty} [\![\mathcal{A}_1]\!]_{(e,w)}$. By induction hypothesis, there is ν' such that $Q \equiv Q_1 \mid \mathbf{val}(e, \nu', w)_K$ and $Q_1, \nu' \circ e \models_{M_K} \mathcal{A}_1$. If $\nu \neq \nu'$, then $\mathbf{val}(e, \nu, w)_K \mid \mathbf{val}(e, \nu', w)_K \not\models_{M_\infty} \mathsf{ProcVal}(e, w+1)_K$, so $P \mid Q \models_{M_\infty} \mathsf{ProcVal}(e, w+1)_K \rightarrow [\![\mathcal{A}_2]\!]_{(e,w+1)}$. Otherwise, $\nu = \nu'$ and by hypothesis $P_1 \mid Q_1, \nu \circ e \models_{M_K} \mathcal{A}_2$, so by induction hypothesis $P \mid Q \models_{M_\infty} [\![\mathcal{A}_2]\!]_{(e,w+1)}$.

- (Case of $\mathcal{A} = \Diamond \mathcal{A}'$) Assume first that $P \models_{M_\infty} [\![\Diamond \mathcal{A}']\!]$. Then $P \equiv P_1 \mid \mathbf{val}(e, \nu, w)_K$, and there is R such that $P \longrightarrow R \models_{M_\infty} [\![\mathcal{A}']\!]_{(e,w)}$. By induction hypothesis, $R \equiv R_1 \mid \mathbf{val}(e, \nu', w)_K$ for some ν' and $R_1, \nu' \circ e \models_{M_K} \mathcal{A}'$. If $\mathbf{val}(e, \nu, w)_K$ takes part to this reduction, it decreases the size of one row or two rows of different depth. So the number of copies of the deeper one is not 2^k any more, and this process is not congruent to $\mathbf{val}(e, \nu', w)_K$. So $P_1 \longrightarrow R_1$ and the result. Assume now $P_1, \nu \circ e \models_{M_K} \Diamond \mathcal{A}'$ and let R_1 be such that $R_1, , \nu \circ e \models_{M_K} \Diamond \mathcal{A}'$ and $P_1 \longrightarrow R_1$. Then $P_1 \mid \mathbf{val}(e, \nu, w)_K \longrightarrow R_1 \mid \mathbf{val}(e, \nu, w)_K$, so $P \models_{M_\infty} [\![\Diamond \mathcal{A}']\!]_{(e,w)}$.
- (Case of $\mathcal{A} = \exists x.\, \mathcal{A}'$) Assume first that $P \models_{M_\infty} [\![\mathcal{A}]\!]$. Then there is an action α such that $\mathbf{row}(K+ \mid e \mid +1, \alpha)^{2^w} \mid P \models [\![\mathcal{A}]\!]$, so by induction hypothesis $P \mid \mathbf{row}(K+ \mid e \mid +1, \alpha)^{2^w} \equiv \mathbf{val}(e', \nu', w)_K \mid P_1$ with $P_1, \nu' \circ e' \models_{M_K} \mathcal{A}'$. Due the difference of row depths, we have $\nu' = \nu, \mid e' \mid \mapsto \alpha$. So $P, \nu \circ e \models_{M_K} \mathcal{A}$, and the result. Conversely, assume $P, \nu \circ e \models_{M_K} \mathcal{A}$. Then there is an action α such that $P, \nu\{x \leftarrow \alpha\} \models_{M_K} \mathcal{A}$. Let consider the process $R = \mathbf{row}(K+ \mid e \mid +1, \alpha)^{2^w}$. Then $\mathbf{val}(e, \nu, w)_K \mid R \equiv \mathbf{val}(e', \nu', w)_K$ with $e' = e, x \mapsto \mid e \mid +1$ and $\nu' = \nu, \mid e \mid +1 \mapsto \alpha$. By induction hypothesis, $P \mid \mathbf{val}(e', \nu', w)_K \models_{M_\infty} [\![\mathcal{A}']\!]$, so $P \mid \mathbf{val}(e, \nu, w)_K \models_{M_\infty} [\![\mathcal{A}]\!]$ and the result.
- (Case of $\mathcal{A} = \langle x \rangle \mathcal{A}'$) Assume first that $P \models_{M_\infty} [\![\mathcal{A}]\!]$. Then there is an action α such that $\mathbf{row}(K+ \mid e \mid +2, \alpha) \mid \mathbf{row}(K + e(x), \nu \circ e(x)) \longrightarrow$. So $\alpha = \overline{\nu \circ e(x)}$. Moreover, there is P' such that $P_1 \mid \mathbf{val}(e, \nu, w)_K \mid \mathbf{row}(K+ \mid e \mid +2, \alpha) \longrightarrow P'$ and $P', \nu \circ e \models_{M_\infty} \mathsf{UsedTest}(e) \mid [\![\mathcal{A}']\!]_{(e,w)}$. So P' has a row of depth $K+ \mid e \mid +1$, which is only possible if the reduction involved $\mathbf{row}(K+ \mid e \mid +2, \alpha)$. It cannot involve $\mathbf{val}(e, \nu, w)_K$ since P' contains it unchanged, so it necessarily involve P_1, that is $P' = \mathbf{row}(K+ \mid e \mid +1, \alpha) \mid \mathbf{val}(e, \nu, w)_K \mid P_1'$ with $P_1 \xrightarrow{\overline{\alpha}} P_1'$, hence the result. Assume now that there is P_1' such that $P_1 \xrightarrow{\nu \circ e(x)} P_1'$; then adding the process $\mathbf{row}(K+ \mid e \mid +2, \overline{\nu \circ e(x)})$ and performing the reduction we just described, we have that $P_1 \mid \mathbf{val}(\nu, e, w)_K, \emptyset \models_{M_\infty} \mathcal{A}$.
- (Case of $\mathcal{A} = \langle \overline{x} \rangle \mathcal{A}'$) Assume first that $P \models_{M_\infty} [\![\mathcal{A}]\!]$. Then there is P', α, β such that $P \longrightarrow P' \mid \mathbf{row}(\beta, n-1)$ and $P' \mid \mathbf{row}(\alpha, n) \models_{M_\infty} [\![\mathcal{A}']\!]_{(e,w)}$, with $n = e(x)$. By induction hypothesis, $P' \mid \mathbf{row}(\alpha, n)$ contains an environment, so in order to have the right number of rows of each depth it must be that a row of size n was absent in P', and one had an extra row of size $n-1$. Then it must be that a row of size n in P contributed to the reduction $P \longrightarrow P' \mid \mathbf{row}(\beta, n-1)$. Hence in the reduction the number of rows

of size n decrease by one, the number of rows of size $n-1$ increased by one, and other rows remained 2^w copies. Moreover, since rows of the same depth always have the same action; we have at least two copies at each size since $w \geq 1$, so necessarily $\alpha = \nu(n)$ and $\mathbf{row}(\beta, n-1)$ is the row that was generated by the reduction, that is $\beta = \alpha = \nu(n)$. Since the interaction did not involve any other row from the environment, it actually must have involved P_1. So there is P_1' such that $P_1 \xrightarrow{\overline{\nu(n)}} P_1'$ and $P' \equiv P_1' \mid \mathsf{env}'$, where env' is the environment $\mathbf{val}(e,\nu,w)_K$ from which a row of size n has been picked up. Then $P' \mid \mathbf{row}(\alpha, n) \equiv P_1' \mid \mathbf{val}(e,\nu,w)_K$ so by induction hypothesis $P_1', \nu \circ e \models_{M_K} \mathcal{A}'$, that is $P_1, \nu \circ e \models_{M_K} \langle \overline{x} \rangle \mathcal{A}'$. Assume now that $P_1, \nu \circ e \models_{M_K} \langle \overline{x} \rangle \mathcal{A}'$. Then there is P_1' such that $P_1 \xrightarrow{\overline{\nu(e(x))}} P_1'$ and $P_1', \nu \circ e \models_{M_K} \mathcal{A}'$. Then by induction hypothesis $P_1' \mid \mathbf{val}(e,\nu,w)_K \models_{M_\infty} [\![\mathcal{A}]\!]'$, that is $_1 \mid \mathbf{val}(e,\nu,w)_K \longrightarrow P_1' \mid \mathsf{env}' \mid \mathbf{row}(\alpha, n-1)$ where $e(x) = n$, $\alpha = \nu(n)$, and env' is $\mathbf{val}(e,\nu,w)_K$ from which one row of size n has been removed. So $P_1' \mid \mathsf{env}' \mid \mathbf{row}(\alpha, n) \models_{M_\infty} [\![\mathcal{A}]\!]'$ by induction hypothesis, that is $P_1 \mid \mathbf{val}(e,\nu,w)_K \models_{M_\infty} [\![\mathcal{A}]\!]$.

□

We can now present the proof of our main Theorem 2.1.

Proof: Let $\mathcal{A}$ be a formula of $\mathcal{L}_{mod}$. Set $[\![\mathcal{A}]\!]_K = [\![\mathcal{A}]\!]_{(\emptyset,w)}$ for some w greater than the maximal nesting of $\mid$ connectives in $\mathcal{A}$. Then $\pi_K(P) \equiv \pi_K(P) \mid \mathbf{val}(\emptyset,\emptyset,w)_K$, so by Lemma 2.7, $\pi_K(P), \emptyset \models_{M_\infty} [\![\mathcal{A}]\!]_K$ if and only if $\pi_K(P), \emptyset \models_{M_K} \mathcal{A}$, which is equivalent to $P, \emptyset \models_{M_\infty} \mathcal{A}$ by Proposition 1.9.

□

In the next section, as a first application of Theorem 2.1 we determine the separation power of $\mathcal{L}_{spat}$.

3 Separability of $\mathcal{L}_{spat}$

As a first application of the main Theorem 2.1, we define characteristic formulas and characterize the separation power of the logic $\mathcal{L}_{spat}$ (and thus of $\mathcal{L}_{mod}$). We conclude that $\mathcal{L}_{spat}$ is able to describe processes quite precisely, just abstracting away from the identity of the particular names used by processes. We start by introducing a characteristic formula $C(P)$ for any process P. For any complementary pair of actions $\{\alpha, \overline{\alpha}\}$ occurring in P, we reserve a specific variable x_a, collected in the set $\{x_{\alpha_1}, \ldots, x_{\alpha_n}\}$.

$$\chi(\mathbf{0}) \triangleq 0 \qquad \chi(\alpha.\, P) \triangleq 1 \wedge \langle x_\alpha \rangle \chi(P)$$
$$\chi(\overline{\alpha}.\, P) \triangleq 1 \wedge \langle \overline{x_\alpha} \rangle \chi(P) \qquad \chi(P \mid Q) \triangleq \chi(P) \mid \chi(Q)$$

$$C(P) \triangleq [\![\exists x_{\alpha_1} \ldots \exists x_{\alpha_n}. (\bigwedge_{i \neq j} x_{\alpha_i} \neq x_{\alpha_j} \ \wedge \ x_{\alpha_i} \neq \overline{x_{\alpha_j}}) \wedge \chi(P)]\!]_K$$

where we fix $K = \mathsf{ds}(P)$. Recall that the abbreviations $(x = y)$ and $(x = \overline{y})$ are defined in Fig.2, and notice indeed that $C(P) \in \mathcal{L}_{spat}$, while $\chi(P) \in \mathcal{L}_{mod}$.

Lemma 3.1 *Let $P \in M_K$, let v be the valuation such that $v(x_{\alpha_i}) = \beta_i$, for pairwise distinct actions $\beta_1, \ldots, \beta_n$, and let σ be the action permutation that sends α_i into β_i. Then we have that $Q, v \models_{M_K} \chi(P)$ if and only $Q \equiv \sigma(P)$.*

Proof: By induction on the structure of P. We detail the case of $P = \alpha_j.\, P'$. If $Q, v \models_{M_K} \chi(P)$ then $Q, v \models_{M_K} 1$ and $Q, v \models_{M_K} \langle x_{\alpha_j} \rangle \chi(P')$. This means that $Q \equiv \beta_j.\, Q'$ and $Q', v \models_{M_K} \chi(P')$, where $b_j = v(x_{\alpha_j})$. By inductive hypothesis, $Q' \equiv \sigma(P')$. Since $\sigma(\alpha_j) = \beta_j$ we conclude $Q \equiv \sigma(P)$. Conversely, assume $Q \equiv \sigma(P)$. This means that $\beta_j = \sigma(\alpha_j)$ and $Q = \beta_j.\, Q'$ where $Q' \equiv \sigma(P')$. By inductive hypothesis, $Q', v \models_{M_K} \chi(P')$. Then, $Q \xrightarrow{\beta_j} Q'$. Since $Q, v \models 1$, and $v(x_{\alpha_j}) = \beta_j$ we conclude $Q, v \models_{M_K} \langle x_\alpha \rangle \chi(P')$ and then $Q, v \models_{M_K} \chi(P)$. $\square$

Lemma 3.2 *For all processes Q and P, $Q \models C(P)$ if and only if $Q \equiv_s P$.*

Proof: Let $K = \mathsf{ds}(P)$ and assume $Q, \emptyset \models C(P)$. Then $Q \in M_K$ and thus $\pi_K(Q) = Q$. By Theorem 2.1 we have $Q, \emptyset \models_{M_K} \exists x_{\alpha_1}. \ldots \exists x_{\alpha_n}. \chi(P)$, so there are pairwise distinct actions β_i such that $v(x_{\alpha_i}) = \beta_i$ and $Q, v \models_{M_K} \chi(P)$. By Lemma 3.1, we conclude that $Q \equiv \sigma(P)$, where $\sigma(\alpha_i) = \beta_i$. Conversely, let $Q \equiv \sigma(P)$ for some action permutation σ; thus if $P \in M_K$ then also $Q \in M_K$. Let $v(\alpha_i) = \beta_i$ whenever $\sigma(\alpha_i) = \beta_i$. By Lemma 3.1, we conclude $Q, v \models_{M_K} \chi(P)$. Since the actions β_i are pairwise distinct, by Theorem 2.1 we conclude $Q \models C(P)$. $\square$

We then conclude:

Theorem 3.3 *The following statements are equivalent:*

$$(1)\ P =_{\mathcal{L}_{mod}} Q \qquad (2)\ P =_{\mathcal{L}_{spat}} Q \qquad (3)\ Q, \emptyset \models C(P) \quad (4)\ P \ \equiv_s \ Q$$

Proof: (1)⇒(2) because $\mathcal{L}_{spat} \subset \mathcal{L}_{mod}$, (2)⇒(3) since $C(P) \in \mathcal{L}_{spat}$ and $P \models C(P)$, (3)⇒(4) by Lemma 3.2, and (4)⇒(1) by Proposition 1.6. $\square$

We thus conclude that the separation power of $\mathcal{L}_{spat}$ and $\mathcal{L}_{mod}$ on the tiny CCS model is exactly the same, and logical equivalence on processes coincides with structural congruence modulo action permutation.

4 Expressiveness of Composition Adjunct

It is known [19] that in static spatial logics, that is spatial logics without quantifiers and dynamic operators, the composition adjunct is not independent of the remaining connectives, and can in fact be eliminated, in the sense that for any formula of such a logic we can find a logically equivalent adjunct-free formula. It is not hard to see that the composition adjunct cannot be dispensed with in the core logic $\mathcal{L}_{spat}$, because without it one is not allowed to distinguish between threads of different length: if we pick $\mathcal{A} \in \mathcal{L}_{spat} - \{\triangleright\}$, we can verify by an easy induction on the structure of the formula $\mathcal{A}$ that $\alpha.\,0 \models \mathcal{A}$ if and only if $\alpha.\,\beta.\,0 \models \mathcal{A}$, for all $\alpha, \beta \in \mathsf{A}$.

In this section, we prove that the adjunct elimination property also does not hold for the spatial logic $\mathcal{L}_{mod}$. For this, we adapt an argument suggested by Hongseok Yang: on the one hand, we define in $\mathcal{L}_{mod}$ a formula that says of a process that its number of toplevel parallel components is even numbered, on the other hand, we show that parity cannot be characterized by any adjunct-free $\mathcal{L}_{mod}$ formula. We start by defining a few formulas:

$$
\begin{aligned}
\Box\mathcal{A} \quad &\triangleq \neg\Diamond\neg\,\mathcal{A} \\
\mathsf{Top}(x) &\triangleq \langle x\rangle 0 \\
\mathsf{Fam} \quad &\triangleq \Box\bot \wedge \Big(1 \Rightarrow \exists x.\,\mathsf{Top}(x)\Big)^{\forall} \wedge \forall x.\,\forall y.\,(\mathsf{Top}(x) \mid \mathsf{Top}(y) \mid \top) \Rightarrow x \neq y)
\end{aligned}
$$

We can verify that $P \models \mathsf{Fam}$ if and only if $P \equiv \alpha_1.\,\mathbf{0} \mid \ldots \mid \alpha_k.\,\mathbf{0}$ for some pairwise distinct k actions $\alpha_1, \ldots, \alpha_k$ such that $P \not\rightarrow$. We call a process of such a form a *family*. The width of such a family P is defined to be the number $w(P) = k$ of parallel threads in P. Notice that the requirement that a family is deadlocked is not essential: it is just a means to simplify our proofs.

Now, we define a formula $\mathsf{Even2}$ that is satisfied by processes that contain exactly an even number of distinct actions at the *second level* (that is, behind the first prefix).

$$
\begin{aligned}
\mathsf{Pair} \quad &\triangleq 1 \;\wedge\; \exists xyz.\,\langle x\rangle(\mathsf{Top}(y) \mid \mathsf{Top}(z)) \wedge (y \neq z) \\
\mathsf{Below}(x) &\triangleq 1 \wedge \exists z.\,\langle z\rangle\langle x\rangle\top \\
\mathsf{Even2} \quad &\triangleq (1 \Rightarrow \mathsf{Pair})^{\forall} \wedge \forall x.\,\forall y.\,(\mathsf{Below}(x) \mid \mathsf{Below}(y) \mid \top) \Rightarrow x \neq y)
\end{aligned}
$$

We can now verify that

$$P \models \mathsf{Even2} \textit{ if and only if } P \equiv \alpha_1.\,(\beta_{1,1}.\,\mathbf{0} \mid \beta_{1,2}.\,\mathbf{0}) \mid \cdots \mid \alpha_k.\,(\beta_{k,1}.\,\mathbf{0} \mid \beta_{k,2}.\,\mathbf{0})$$

for some k actions $\alpha_1, \ldots, \alpha_k$, and some pairwise distinct $2k$ actions $\beta_{1,i}, \ldots, \beta_{k,i}$ for $i = 1, 2$. Now, if we compose a process P satisfying Fam in parallel with a process Q satisfying $\mathsf{Even2}$, we can check, using the formula Same in the composition $P \mid Q$, that the actions that occur in the toplevel of process P are exactly the same that appear in the second level of process Q :

$$\mathsf{Same} \triangleq \forall x.\, (\mathsf{Top}(x)^{\exists} \Leftrightarrow\ \mathsf{Below}(x)^{\exists})$$

Hence we have the following result

Lemma 4.1 *There is a closed formula* $\mathsf{Even} \in \mathcal{L}_{mod}$ *such that for any process* P*, we have that* $P \models \mathsf{Even}$ *if and only if* P *is a family and* $w(P)$ *is even.*

Proof: Let $\mathsf{Even} \triangleq \mathsf{Fam} \wedge (\mathsf{Even2} \blacktriangleright \mathsf{Same})$. □

The key observation here is that the formula Even contains an essential use of the composition adjunct operator. In fact, although the properties denoted by the formulas $\mathsf{Even2}$ and Fam can be expressed by appropriate adjunct-free formulas of $\mathcal{L}_{spat}$, the same situation does not hold for the parity property expressed by Even. In the remainder of this section, we prove that there is no formula of $\mathcal{L}_{mod} - \{\triangleright\}$ able to express the same property. The argument consists in showing that any family P considered in $\mathcal{L}_{mod} - \{\triangleright\}$ admits a saturation level from which it is always possible to add an extra parallel component to it while preserving satisfaction. We first define $\mathsf{sn}(\mathcal{A})$ (the *sticks number* of the formula $\mathcal{A}$) to be the natural number defined by induction on $\mathcal{A}$ as follows:

$$\begin{array}{llll}
\mathsf{sn}(\neg\,\mathcal{A}) & \triangleq \mathsf{sn}(\mathcal{A}) & \mathsf{sn}(\mathcal{A}_1 \wedge \mathcal{A}_2) & \triangleq \max(\mathsf{sn}(\mathcal{A}_1), \mathsf{sn}(\mathcal{A}_2)) \\
\mathsf{sn}(0) & \triangleq 1 & \mathsf{sn}(\mathcal{A}_1 \mid \mathcal{A}_2) & \triangleq \mathsf{sn}(\mathcal{A}_1) + \mathsf{sn}(\mathcal{A}_2) \\
\mathsf{sn}(\Diamond\mathcal{A}) & \triangleq 0 & \mathsf{sn}(\langle x\rangle.\,\mathcal{A}) & \triangleq \mathsf{sn}(\mathcal{A}) \\
\mathsf{sn}(\exists x.\,\mathcal{A}) & \triangleq \mathsf{sn}(\mathcal{A}) + 1 & \mathsf{sn}(\langle \overline{x}\rangle.\,\mathcal{A}) & \triangleq \mathsf{sn}(\mathcal{A})
\end{array}$$

Given a family P and a valuation v, we write $P\backslash v$ for the subfamily of P grouping the actions α that do not appear in the codomain of the valuation v. More precisely, we define

$$P\backslash v \triangleq \prod\{\alpha.\,\mathbf{0}\ :\ P \equiv \alpha.\,\mathbf{0} \mid Q \text{ and } \alpha, \overline{\alpha} \notin \mathsf{codom}(v)\}$$

We then state and prove

Lemma 4.2 *Let* P *be a family, let* v *be a valuation* $v : \mathsf{X} \rightharpoonup \mathsf{A}$*, and let* $\alpha \in \mathsf{A}$ *be an action such that* $\alpha, \overline{\alpha} \notin \mathsf{codom}(v)$ *and* $(P \mid \alpha.\,\mathbf{0})$ *is a family. Then, for any* $\triangleright$*-free formula* $\mathcal{A} \in \mathcal{L}_{mod}$ *such that* $w(P\backslash v) \geq \mathsf{sn}(\mathcal{A})$ *we have*

$$P,\ v\ \models\ \mathcal{A} \qquad \textit{if and only if} \qquad P\ \mid\ \alpha.\,\mathbf{0},\ v\ \models\ \mathcal{A}.$$

Proof: By induction on $\mathcal{A}$.

- (Cases of $\mathcal{A} = \mathcal{A}_1 \wedge \mathcal{A}_2$, $\mathcal{A} = \neg\,\mathcal{A}_1$) Straightforward.
- (Case of $\mathcal{A} = 0$) We have $w(P) \geq 1$ so neither P nor $P \mid \alpha$ satisfy $\mathcal{A}$.
- (Case of $\mathcal{A} = \mathcal{A}_1 \mid \mathcal{A}_2$). We assume first that $P, v \models \mathcal{A}$. Then there is P_1, P_2 such that $P \equiv P_1 \mid P_2$ and $P_i \models \mathcal{A}_i$, for $i \in \{1, 2\}$. Then

  $$w(P\backslash v) = w(P_1\backslash v) + w(P_2\backslash v) \geq \mathsf{sn}(\mathcal{A}) = \mathsf{sn}(\mathcal{A}_1) + \mathsf{sn}(\mathcal{A}_2)$$

  so there is some $e \in \{1, 2\}$ such that $w(P_e\backslash v) \geq \mathsf{sn}(\mathcal{A}_e)$. By induction hypothesis then $P_e \mid \alpha \models \mathcal{A}_e$, so that $P \mid \alpha \models \mathcal{A}$. We assume now that $P \mid \alpha \models \mathcal{A}$. Then there are Q_1, Q_2 such that $P \mid \alpha = Q_1 \mid Q_2$ and $Q_i \models \mathcal{A}_i$. Since $\alpha \notin \mathsf{codom}(v)$, $w(P \mid \alpha\backslash v) = w(P\backslash v) + 1$, so

  $$w(P \mid \alpha\backslash v) = w(Q_1\backslash v) + w(Q_2\backslash v) > \mathsf{sn}(\mathcal{A}) = \mathsf{sn}(\mathcal{A}_1) + \mathsf{sn}(\mathcal{A}_2)$$

  and there is some $e \in \{1, 2\}$ such that $w(Q_e\backslash v) > \mathsf{sn}(\mathcal{A}_e)$. We pick some $\alpha' \in Q_e$ with $\alpha' \notin \mathsf{codom}(v)$, which is possible since $w(Q_e\backslash v) \geq 1$. We note P_e the family such that $Q_e\{\alpha \leftrightarrow \alpha'\} \equiv P_e \mid \alpha$. Then $w(P_e\backslash v) = w(P_e \mid \alpha\backslash v) - 1 = w(Q_e\backslash(v\{\alpha \leftrightarrow \alpha\}')) - 1 = w(Q_e\backslash v) - 1$, hence $w(P_e\backslash v) \geq \mathsf{sn}(\mathcal{A}_e)$. By equivariance (Proposition 1.6), we get from $Q_e, v \models \mathcal{A}_e$ that $P_e \mid \alpha, v \models \mathcal{A}_e$, and by induction hypothesis $P_e, v \models \mathcal{A}_e$. Then we write $P \mid \alpha = Q_1\{\alpha \leftrightarrow \alpha'\} \mid Q_2\{\alpha \leftrightarrow \alpha'\} = P_1 \mid \alpha \mid P_2$, which gives that $P, v \models \mathcal{A}$.
- (Case of $\mathcal{A} = \Diamond\mathcal{A}_1$) Then both P and $P \mid \alpha$ do not satisfy $\mathcal{A}$, because they are deadlocked.
- (Case of $\mathcal{A} = \exists x.\,\mathcal{A}_1$) We assume first that $P, v \models \mathcal{A}$. Then there is β such that for $v' = v, x \mapsto \beta$, $P, v' \models \mathcal{A}_1$. We may assume that $\beta \notin \{\alpha, \overline{\alpha}\}$, otherwise we would pick some fresh action β', and consider instead $v' = v, x \mapsto \beta'$: then $P = P\{\beta \leftrightarrow \beta\}'$ and $v' = v\{\beta \leftrightarrow \beta\}'$ (since $\beta \notin \mathsf{codom}(v)$), so $P, v' \models \mathcal{A}_1$ by Proposition 1.6. So we assume $\beta \notin \{\alpha, \overline{\alpha}\}$. We have $w(P \mid \alpha\backslash v') = w(P\backslash v') + 1 \geq w(P\backslash v)$, so $w(P \mid \alpha\backslash v') \geq \mathsf{sn}(\mathcal{A}_1)$, and by induction hypothesis $P \mid \alpha, v' \models \mathcal{A}_1$, that is $P \mid \alpha, v \models \mathcal{A}$. We assume now that $P \mid \alpha, v \models \mathcal{A}$. Then there is β such that for $v' = v, x \mapsto \beta$, $P \mid \alpha, v' \models \mathcal{A}_1$. If $\beta \in \{\alpha, \overline{\alpha}\}$, we may pick some other β' occurring in $P\backslash v$: then $P\{\beta \leftrightarrow \beta'\} = P$ by internal symmetry, and $v'' = v'\{\beta \leftrightarrow \beta'\}$, so $P \mid \alpha, v'' = P\{\beta \leftrightarrow \beta'\}, v'\{\beta \leftrightarrow \beta'\} \models \mathcal{A}_1$ for $v'' = v, x \mapsto \beta'$. So we may assume $\beta \notin \{\alpha, \overline{\alpha}\}$. Then $w(P\backslash v') \geq w(P\backslash v) - 1 \geq \mathsf{sn}(\mathcal{A}_1)$, so by induction hypothesis $P, v' \models \mathcal{A}_1$, that is $P, v \models \mathcal{A}$.
- (Case $\mathcal{A} = \langle x\rangle\mathcal{A}_1$) Assume first that $P, v \models \mathcal{A}$. Then there is P' such that $P \xrightarrow{v(x)} P'$ and $P', v \models \mathcal{A}_1$. $w(P'\backslash v) = w(P\backslash v) \geq \mathsf{sn}(\mathcal{A}_1)$, so by induction hypothesis $P' \mid \alpha, v \models \mathcal{A}_1$, that is $P \mid \alpha, v \models \mathcal{A}$. Assume now that $P \mid \alpha, v \models \mathcal{A}$. Then there is P_1 such that $P \mid \alpha \xrightarrow{v(x)} P_1$ with $P_1, v \models \mathcal{A}_1$. Since $\alpha \notin \mathsf{codom}(v)$ by assumption, $P_1 \equiv P' \mid \alpha$ with $P \xrightarrow{v(x)} P'$. We have $w(P'\backslash v) = w(P\backslash v) \geq \mathsf{sn}(\mathcal{A}_1)$, so by induction hypothesis $P', v \models \mathcal{A}_1$, that is $P, v \models \mathcal{A}$.

$$
\begin{array}{ll}
(D,I) \models_v \mathcal{A} \wedge \mathcal{B} & \textit{if } (D,I) \models_v \mathcal{A} \textit{ and } (D,I) \models_v \mathcal{B} \\
(D,I) \models_v \neg \mathcal{A} & \textit{if not } (D,I) \models_v \mathcal{A} \\
(D,I) \models_v \exists x.\, \mathcal{A} & \textit{if there is } d \in D.\, (D,I) \models_{v\{x \leftarrow d\}} \mathcal{A} \\
(D,I) \models_v p(x,y) & \textit{if } (v(x), v(y)) \in I
\end{array}
$$

Fig. 4. Satisfaction of FOL

- (Case of $\mathcal{A} = \langle \overline{x} \rangle.\, \mathcal{A}_1$) the proof proceeds as in the previous case.

□

Theorem 4.3 *There is no closed formula $\mathcal{A} \in \mathcal{L}_{mod} - \{\triangleright\}$ that exactly characterizes the set of all families P such that $w(P)$ is even.*

Proof: By contradiction. If there exists such formula $\mathcal{A}$, then we may take a family P, and an extended family $P \mid \alpha$ with $w(P) \geq \mathsf{sn}(\mathcal{A})$. Then, by the Lemma 4.2, we have $P, \emptyset \models \mathcal{A}$ if and only if $P \mid \alpha, \emptyset \models \mathcal{A}$, which is a contradiction. □

We conclude that, unlike in static spatial logics, in the logic $\mathcal{L}_{mod}$ the composition adjunct operator is independent of the remaining operators, since there are properties expressible with the composition adjunct that cannot be expressed with action modalities and quantifiers. Hence,

Theorem 4.4 *$\mathcal{L}_{mod}$ is strictly more expressive than $\mathcal{L}_{mod} - \{\triangleright\}$.*

5 Undecidability

In this section, we show that the validity-checking, satisfiability-checking and model-checking problems for the logic $\mathcal{L}_{spat}$ (and hence for $\mathcal{L}_{mod}$) are all undecidable. These results are a consequence of our embedding of $\mathcal{L}_{mod}$ into $\mathcal{L}_{spat}$ (Theorem 2.1), and of the fact that first-order logic can then be easily encoded into $\mathcal{L}_{mod}$ along the lines of Charatonik and Talbot [13].

The language of first-order logic (FOL) is defined in the standard way from a set $Vars$ of individual variables (x, y). Without loss of generality we consider a single binary predicate symbol p.

$$\mathcal{A}, \mathcal{B} ::= \mathcal{A} \wedge \mathcal{B} \mid \neg \mathcal{A} \mid \exists x.\, \mathcal{A} \mid p(x,y)$$

A model for FOL is a pair (D, I) where D is a set of individuals (the domain of the model), and I is a binary relation $I \subseteq D \times D$. For our purposes it is

enough to focus on finite models. Satisfaction of a FOL formula by a model is defined in Fig. 4, using a valuation v that assigns each individual variable an element of D.

We now show how to encode any FOL satisfaction judgment $(D, I) \models_v \mathcal{A}$ into a $\mathcal{L}_{mod}$ satisfaction judgment

$$\mathcal{M}[\![(D, I)]\!], \mathcal{V}[\![v]\!] \models \mathcal{F}[\![A]\!]$$

by means of appropriate translations $\mathcal{M}[\![-]\!]$, $\mathcal{V}[\![-]\!]$ and $\mathcal{F}[\![-]\!]$. We pick natural numbers K, E such that $K > E > 2$. To encode a model (D, I) into a process $\mathcal{M}[\![(D, I)]\!]$, we start by assigning to each element $d \in D$ a distinct action $A(d) \in \mathsf{A}$, and define $\mathcal{E}[\![d]\!] \triangleq \mathbf{row}(E, A(d))$. The domain $D = \{d_1, \ldots, d_n\}$ is then represented by the process $\mathcal{D}[\![D]\!] \triangleq \mathcal{E}[\![d_1]\!] \mid \ldots \mid \mathcal{E}[\![d_k]\!]$. For the interpretation I, we represent each pair $(d, e) \in I$ by the process $\mathcal{T}[\![(d, e)]\!] \triangleq A(d).A(e).\mathbf{0}$. We then let $\mathcal{I}[\![I]\!] \triangleq \prod_{(d,e)\in I} \mathcal{T}[\![(d, e)]\!]$ and finally set $\mathcal{M}[\![(D, I)]\!] \triangleq \mathcal{D}[\![D]\!] \mid \mathcal{I}[\![I]\!]$. Notice that, by construction, we always have $\mathcal{M}[\![(D, I)]\!] \in M_K$.

Processes encoding FOL models can be characterized by a formula Model of $\mathcal{L}_{spat}$, which is defined by making use of our encoding of $\mathcal{L}_{mod}$ into $\mathcal{L}_{spat}$ as follows

$$\begin{array}{ll}
\mathsf{D}(x) & \triangleq \mathsf{Row}(E) \wedge \langle x \rangle \top \\
\mathsf{Diff} & \triangleq \forall x. \forall y. (\langle x \rangle \top \mid \langle y \rangle \top) \Rightarrow x \neq y) \\
\mathsf{Domain} & \triangleq \mathsf{Diff} \wedge \Big(1 \Rightarrow \exists x. \mathsf{D}(x)\Big)^{\forall} \\
\mathsf{Compat} & \triangleq \forall x. \forall y((\langle x \rangle \langle y \rangle 0)^{\exists} \Rightarrow (\mathsf{D}(x)^{\exists} \wedge \mathsf{D}(y)^{\exists})) \\
\mathsf{Interp} & \triangleq \Big(1 \Rightarrow \mathsf{Thread}(2)\Big)^{\forall} \\
\mathsf{Model} & \triangleq \mathcal{M}_K \wedge [\![(\mathsf{Domain} \mid \mathsf{Interp}) \wedge \mathsf{Compat}]\!]_K
\end{array}$$

The $\mathsf{Row}(n)$ and $\mathsf{Thread}(n)$ formulas were defined in Section 2. We can then verify

Lemma 5.1 *We have $P \models \mathsf{Model}$ if and only if there is a finite FOL model (D, I) and $\mathcal{M}[\![(D, I)]\!] \equiv P$.*

Proof: (if) As already remarked, we have $P \equiv \mathcal{M}[\![(D, I)]\!] \in M_K$, hence $P \models \mathcal{M}_K$. We can also check that $P \models_{M_K} (\mathsf{Domain} \mid \mathsf{Interp}) \wedge \mathsf{Compat})$, because $\mathcal{D}[\![D]\!] \models \mathsf{Domain}$ and $\mathcal{I}[\![I]\!] \models \mathsf{Interp}$. Since $P \in M_K$, by Theorem 2.1(if) we conclude that $P \models \mathsf{Model}$.

(only if) If $P \models \mathsf{Model}$ we conclude that $P \in M_K$ and $P \models [\![(\mathsf{Domain} \mid \mathsf{Interp}) \wedge$

$\mathsf{Compat}]\!]_K$. By Theorem 2.1, we have $P \models_{M_K} (\mathsf{Domain} \mid \mathsf{Interp}) \wedge \mathsf{Compat}$. This means that $P \equiv P_D \mid P_I$ where $P_D \models \mathsf{Domain}$ and $P_I \models \mathsf{Interp}$. In turn, we conclude that $P_D \equiv \mathbf{row}(E, \alpha_1) \mid \cdots \mid \mathbf{row}(E, \alpha_k)$, where the actions α_i are pairwise distinct. We can then construct a finite FOL model (D, I) from P_D and P_I, by letting $D = \{\alpha_1, \ldots, \alpha_k\}$, and $I = \{(d, e) : \exists R.\, P_I \equiv A(d).\, A(e).\, \mathbf{0} \mid R\}$. Since $P \models \mathsf{Compat}$ we indeed have $I \subseteq D \times D$. □

Then, formulas of FOL are encoded into formulas of $\mathcal{L}_{mod}$ as follows

$$\begin{aligned}
\mathcal{F}[\![\neg \mathcal{A}]\!] &\triangleq \neg \mathcal{F}[\![\mathcal{A}]\!] \\
\mathcal{F}[\![\exists x.\, \mathcal{A}]\!] &\triangleq \exists x.\, (\mathsf{D}(x)^{\exists} \wedge A) \\
\mathcal{F}[\![\mathcal{A} \wedge \mathcal{B}]\!] &\triangleq \mathcal{F}[\![\mathcal{A}]\!] \wedge \mathcal{F}[\![\mathcal{B}]\!] \\
\mathcal{F}[\![p(x, y)]\!] &\triangleq \left(1 \wedge \langle x \rangle \langle y \rangle 0\right)^{\exists}
\end{aligned}$$

while, for valuations, we simply set $\mathcal{V}[v]$ to be the valuation such that $\mathcal{V}[v](x) = A(v(x))$. We can then prove

Lemma 5.2 *Let $v = \{x_1 \mapsto d_1, \ldots, x_k \mapsto d_k\}$ be a valuation for $\mathcal{A}$. Then we have $(D, I) \models_v \mathcal{A}$ if and only if $\mathcal{M}[\![(D, I)]\!], \mathcal{V}[\![v]\!] \models_{M_K} \mathcal{F}[\![\mathcal{A}]\!]$.*

Proof: By induction on the structure of formulas. We detail the case of $A = p(x, y)$. If $(D, I) \models p(x, y)$ then $(v(x), v(y)) \in I(p)$, and thus

$$\mathcal{I}[\![I]\!] \equiv A(v(x)).\, A(v(y)).\, \mathbf{0} \mid R$$

for some R. Hence we have $\mathcal{M}[\![(D, I)]\!], \mathcal{V}[\![v]\!] \models_{M_K} (1 \wedge \langle x \rangle \langle y \rangle 0)^{\exists}$. Conversely, if $\mathcal{M}[\![(D, I)]\!], \mathcal{V}[\![v]\!] \models_{M_K} \mathcal{F}[\![p(x, y)]\!]$, we conclude that $\mathcal{M}[\![(D, I)]\!] \equiv \alpha.\, \beta.\, \mathbf{0} \mid R$ for some R and α, β such that $v(x) = \alpha$ and $v(y) = \beta$. We conclude that there are $d, e \in D$ such that $A(d) = \alpha$ and $A(e) = \beta$ and $(d, e) \in I(p)$, by construction of $\mathcal{I}[\![I]\!]$. Hence $(D, I) \models_v p(x, y)$. □

Proposition 5.3 *Let $\mathcal{A}$ be a closed formula of FOL. Then the formula $\mathcal{A}$ is satisfiable if and only if the $\mathcal{L}_{spat}$ formula $\mathsf{Model} \wedge [\![\mathcal{F}[\![\mathcal{A}]\!]]\!]_K$ is satisfiable.*

Proof: Assume that the formula $\mathcal{A}$ is satisfiable. Then there is a FOL model (D, I) such that $(D, I) \models \mathcal{A}$. By Lemma 5.2, we have that $\mathcal{M}[\![(D, I)]\!] \models_{M_K} \mathcal{F}[\![\mathcal{A}]\!]$. By Theorem 2.1, we conclude $\mathcal{M}[\![(D, I)]\!] \models [\![\mathcal{F}[\![A]\!]]\!]_K$, for $\mathcal{M}[\![(D, I)]\!] \in M_K$. Since $\mathcal{M}[\![(D, I)]\!] \models \mathsf{Model}$ by Lemma 5.1, we conclude that $\mathsf{Model} \wedge [\![\mathcal{F}[\![\mathcal{A}]\!]]\!]_K$ is satisfiable. Conversely, if $\mathsf{Model} \wedge [\![\mathcal{F}[\![\mathcal{A}]\!]]\!]_K$ is satisfiable, then there is a process P such that $P \models \mathsf{Model}$ (and thus $P \in M_K$) and $P \models [\![\mathcal{F}[\![\mathcal{A}]\!]]\!]_K$. By Theorem 2.1, we have that $P, \emptyset \models_{M_K} \mathcal{F}[\![A]\!]$, and by Lemma 5.1 we conclude that there is a finite model (D, I) such that $P \equiv \mathcal{M}[\![(D, I)]\!]$. Hence $\mathcal{M}[\![(D, I)]\!] \models_{M_K} \mathcal{F}[\![\mathcal{A}]\!]$, so by Lemma 5.2 we conclude $(D, I) \models \mathcal{A}$. □

As a corollary of Proposition 5.3, we conclude

Theorem 5.4 *The problems of validity-checking, satisfiability-checking, and model-checking of $\mathcal{L}_{spat}$ formulas are all undecidable.*

Proof: Follows from Proposition 5.3 and Trakhtenbrot's Theorem [24]. □

6 Extension to the π-Calculus and Mobile Ambients

In this section, we briefly discuss how our results extend to richer models, namely the π-calculus and the ambient calculus. We may pick any of these calculi as models for the core logic $\mathcal{L}_{spat}$, which is a fragment of both the ambient logic of [11] and the π-calculus logic of [4]. We discuss first the case of the ambient calculus without name restriction, and just with the open capability. In this case, we can show that $\mathcal{L}_{spat}$ can also encode, for processes of bounded depth, its extension with the quantifier $\exists x.\,\mathcal{A}$, and modalities of the form $\langle \mathsf{open}\ x\rangle.\,\mathcal{A}$ and $x[\mathcal{A}]$. However, as we might expect, the symmetry between input and output (Theorem 3.3(4)) does not carry over to the calculus of mobile ambients: for instance, the formula $1 \wedge \Diamond\top$ may be satisfied by the ambient $n[P]$, but not by the guarded ambient $\mathsf{open}\ n.\,P$. For the π-calculus, we may consider the extension of $\mathcal{L}_{spat}$ with the quantifier $\exists x.\,\mathcal{A}$ and the modalities $\langle x\rangle\mathcal{A}$ and $\langle \overline{x}\rangle\mathcal{A}$, able to observe just the subjects of π-calculus actions. In this case, we may also prove that this extension can be encoded in $\mathcal{L}_{spat}$ for bounded depth processes, as we did for the other cases. From these results, we conclude

Theorem 6.1 *The model-checking and validity problems for the π-calculus and the ambient calculus against $\mathcal{L}_{spat}$ are both undecidable.*

We only sketch our proofs, since they are obtained by adapting to the ambient calculus and to the π-calculus the constructions and arguments already shown in detail for the fragment of CCS considered in the paper.

It should be clear that the basic ingredients of our encoding of $\mathcal{L}_{mod}$ in $\mathcal{L}_{spat}$ (Theorem 2.1), in turn used to prove independence of adjunct (Theorem 4.3), and undecidability (Theorem 5.4), are the definitions for the formulas $\mathsf{Thread}(k)$, $\mathsf{Row}(k)$ and $\mathcal{M}_k$, characterizing respectively threads, rows, and the bounded interpretations M_k.

Thus, we just detail here how to provide counterparts to these formulas, by taking in consideration each of the calculi now under consideration. It turns out that for ambients this is quite easy, while for the case of the π-calculus, because of name restriction and name passing, the development is slightly more involved.

For simplicity, we consider the fragment of the Ambient calculus defined by the following grammar:

$$P, Q \quad ::= \quad \mathsf{open}\ n.\, P \;\Big|\; n[P] \;\Big|\; P \mid Q \;\Big|\; \mathbf{0}$$

where a, n, m, p ranges over the set of names Λ. We assume given the reduction relation $P \to P'$ as defined in [15]. We also define the depth $\mathsf{ds}(P)$ of a process P, and the set of models M_k as expected. We also consider the extension $\mathcal{L}_{mod}^{AMB}$ of $\mathcal{L}_{spat}$:

$$\begin{array}{rll} \mathcal{A}, \mathcal{B} ::= & \mathcal{A} \wedge \mathcal{B} \quad \mid \mathcal{A} \mid \mathcal{B} \mid \quad \neg\, \mathcal{A} \quad \mid \mathcal{A} \triangleright \mathcal{B} \mid 0 \mid \Diamond \mathcal{A} & (\mathcal{L}_{spat}) \\ \mid & \langle \mathsf{open}\ x \rangle \mathcal{A} \quad \mid x[\mathcal{A}] \quad \mid \quad \exists x.\, \mathcal{A} & (\mathcal{L}_{mod}^{AMB}) \end{array}$$

Semantics for $\mathcal{L}_{mod}^{AMB}$ specific connectives is defined as expected, relative to a valuation $v : \mathsf{X} \to \Lambda$. We then have

$$\begin{array}{lll} P, v \models_M \exists x.\, \mathcal{A} & \textit{iff} & \exists\ n \in \Lambda.\ P, v\{x \leftarrow n\} \models_M \mathcal{A} \\ P, v \models_M x[\mathcal{A}] & \textit{iff} & \exists Q.\ P \equiv v(x)[Q] \text{ and } Q, v \models_M \mathcal{A} \\ P, v \models_M \langle \mathsf{open}\ x \rangle \mathcal{A} & \textit{iff} & \exists Q.\ P \xrightarrow{\mathsf{open}\ v(x)} Q \text{ and } Q, v \models_M \mathcal{A} \end{array}$$

where $P \xrightarrow{\mathsf{open}\ n} Q \triangleq \exists P.\, P \equiv \mathsf{open}\ n.\, R \mid R'$ and $Q \equiv R \mid R'$. We now show how to mimick the encoding of Section 2 for the case of ambients: we encode the valuation together with process P to be model-checked by defining "rows processes" that exceed P in depth. Then using such rows we make the model-checked process interact, so that we may encode the modalities $x[\mathcal{A}]$ and $\langle \mathsf{open}\ x \rangle \mathcal{A}$. We then define rows and threads as processes of the form:

$$\begin{array}{ll} \mathsf{threadopen}(\widetilde{a}, n) & \triangleq \mathsf{open}\ a_1.\, \mathsf{open}\ a_2 \ldots \mathsf{open}\ a_n.\, \mathbf{0} \\ \mathsf{rowopen}(a, n) & \triangleq \underbrace{\mathsf{open}\ a.\, \mathsf{open}\ a \ldots \mathsf{open}\ a.}_{n \text{ times}}\, \mathbf{0} \end{array}$$

The formulas shown in Fig. 5 show how we may characterise these processes logically. The embedding of $\mathcal{L}_{mod}^{AMB}$ into $\mathcal{L}_{spat}$ then follows the one in Section 2,

Formula	Encoding	Interpretation
atom	$1 \blacktriangleright 1\Diamond 0$	$\exists a.\ \ P \equiv \mathsf{open}\ a$ or $P \equiv a[0]$
$\mathsf{testamb}$	$1\ \wedge \Diamond\mathsf{atom}$	$\exists a, b\ P \equiv a\Big[\mathsf{open}\ b \mid b[\mathbf{0}]\Big]$
$\mathsf{1open}$	$\mathsf{atom}\ \wedge \mathsf{testamb} \blacktriangleright \Diamond\Diamond 0$	$\exists a.\ P \equiv \mathsf{open}\ a$
$\mathsf{1amb}$	$\mathsf{atom}\ \wedge \mathsf{1open} \blacktriangleright \Diamond 0$	$\exists a.\ P \equiv a[0]$
$\mathsf{ThrOp}(1)$	$\mathsf{1amb} \blacktriangleright \Diamond 0$	$\exists \widetilde{a}.\ P \equiv \mathsf{threadopen}(\widetilde{a}, 1)$
$\mathsf{ThrOp}(k+1)$	$1\ \wedge\ \mathsf{1amb} \blacktriangleright \Diamond\mathsf{ThrOp}(k)$	$\exists \widetilde{a}.\ P \equiv \mathsf{threadopen}(\widetilde{a}, k+1)$
$\mathcal{M}_0$	0	$P \in M_0$
$\mathcal{M}_{k+1}$	$\Big(1 \Rightarrow \mathsf{atom} \blacktriangleright \Diamond\mathcal{M}_k\Big)^{\forall}$	$P \in M_{k+1}$
EqOp	$\Big(\mathsf{1open} \triangleright \Box\bot\Big)\ \wedge\ \Big(\mathsf{1amb} \blacktriangleright (\mathsf{1amb} \mid 1 \Rightarrow \Diamond\top)^{\forall}\Big)$	$\exists\ a, \widetilde{P}.\ P \equiv \mathsf{open}\ a.\ P_1 \mid \ldots \mid \mathsf{open}\ a.\ P_n$
$\mathsf{RowOp}(1)$	$\mathsf{1open}$	$\exists a.\ P \equiv \mathsf{rowopen}(a, 1)$
$\mathsf{RowOp}(k+1)$	$(\mathsf{1amb} \mid \mathsf{1open}) \blacktriangleright \Big((\mathsf{1amb} \mid \mathsf{EqOp}) \wedge \Diamond\ \Big((\mathsf{RowOp}(k) \mid \mathsf{1open}) \wedge \mathsf{EqOp}\Big)\Big)$	$\exists a.\ P \equiv \mathsf{rowopen}(a, k+1)$

Fig. 5.

small differences only appear in encoding of modalities. We set

$$
\begin{array}{lcl}
[\![\exists x.\, \mathcal{A}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K\ \wedge \Big(\Big(\mathsf{RowOp}(\mid e \mid +1)\Big)^{2^w} \blacktriangleright [\![\mathcal{A}]\!]_{(e',w)}\Big) \\
 & & \quad \text{where } e' = e\{x{\leftarrow} \mid e \mid +1\} \\
[\![\langle \mathsf{open}\ x\rangle \mathcal{A}]\!]_{(e,w)} & \triangleq & \mathsf{ProcVal}(e,w)_K\ \wedge \\
 & & \mathsf{RowOp}(K + e(x)) \ \Big|\ \Big(\mathsf{TestRow}(K + e(x)) \blacktriangleright \Diamond [\![\mathcal{A}]\!]_{(e,w)}\Big) \\
[\![x[\mathcal{A}]]\!]_{(e,w)} & \triangleq & (\mathsf{Val}(e,w)_K \mid \mathcal{M}_k) \\
 & & \wedge \Diamond\Big(\mathsf{RowOp}(K + e(x) - 1) \ \Big| \\
 & & \quad (\mathsf{RowOp}(K + e(x)) \blacktriangleright [\![\mathcal{A}]\!]_{(e,w)})\Big)
\end{array}
$$

where $\mathsf{TestRow}(n)$ is the formula

$$\mathsf{TestRow}(n) \stackrel{\text{def}}{=} 1 \ \wedge \ (\mathsf{1amb} \mid \mathsf{1open} \wedge \Diamond\top) \blacktriangleright \Diamond\Big(\mathsf{1amb} \mid \mathsf{RowOp}(n) \wedge \Diamond\top\Big)$$

which characterises processes of the form $\mathsf{testproc}(a,n) \stackrel{\text{def}}{=} a[\mathsf{rowopen}(a,n)]$.

We again prove the correctness of the embedding by induction on $\mathcal{A}$. Differences with the proof of Lemma 2.7 happen only for the modality cases:

- $\mathcal{A} = \langle \mathsf{open}\ x\rangle\mathcal{A}'$: Assume first that $P \models [\![\mathcal{A}]\!]_{e,w}$. Then $P \equiv Q \mid \mathbf{val}(e,\nu,w)_K$, and if Val' is such that $\mathbf{val}(e,\nu,w)_K \equiv \mathsf{Val}' \mid \mathsf{rowopen}(\nu(K+e(x)),)$, then there is some name a such that $Q \mid \mathsf{Val}' \mid \mathsf{testproc}(a, K+e(x)) \longrightarrow P'$ and $P' \models [\![\mathcal{A}']\!]_{e,w}$. By induction, $P' \models \mathsf{ProcVal}(e,w)_K$, so the $\mathsf{testproc}$ must have been alterated by the reduction. Moreover, the other partner for the reduction cannot be in Val' since it would consume a row copy which would be visible at end. So this must be the process, that is the reduction is $Q \mid \mathsf{Val}' \mid \mathsf{testproc}(b, K+e(x)) \longrightarrow Q' \mid \mathsf{Val}' \mid \mathsf{rowopen}(a, K+e(x))$ with $Q \stackrel{\mathsf{open}\ a}{\longrightarrow} Q'$. Since $\mathsf{Val}' \mid \mathsf{rowopen}(a, \equiv)\mathbf{val}(e,\nu',w)_K$, this must be that $\nu = \nu'$ and $a = \nu(e(x))$, and the result by induction. Reciproquely, if $P, \nu \circ e \models_{M_K} \mathcal{A}$, then there is P' such that $P \stackrel{\mathsf{open}\ a}{\longrightarrow} P'$ with $a = \nu \circ e(x)$ and $P', \nu \circ e \models' \mathcal{A}$. Then $P \mid \mathsf{Val}' \mid \mathsf{Testproc}(a, K+e(x)) \longrightarrow P' \mid \mathbf{val}(e,\nu,w)_K$, and $P' \mid \mathbf{val}(e,\nu,w)_K \models [\![\mathcal{A}']\!]_{e,w}$ by induction, so that finaly $P \models [\![\mathcal{A}]\!]_{e,w}$.
- $\mathcal{A} = x[\mathcal{A}']$: Assume first that $P \models [\![\mathcal{A}]\!]_{e,w}$. Then $P \equiv Q \mid \mathbf{val}(e,\nu,w)_K$ with Q single, there is a, b, P' such that $P \longrightarrow P' \mid \mathsf{rowopen}(a, K+e(x)-1)$, and $P' \mid \mathsf{rowopen}(b, K+e(x)) \models [\![\mathcal{A}']\!]_{e,w}$. By induction, $P' \mid \mathsf{rowopen}(b, K+e(x))$ contains a process $\mathbf{val}(e,\nu',w)_K$, and again the only way to have this is to have $\nu = \nu'$ and $b = \nu(K+e(x))$. This says that a row of depth $K+e(x)$ was consumed by the reduction but not any other, so that $Q \equiv a[Q']$, $P' \equiv Q' \mid \mathsf{Val}'$ with $\mathbf{val}(e,\nu,w)_K \equiv \mathsf{Val}' \mid \mathsf{rowopen}(a, K+e(x)+1)$, $a = b = \nu(x)$, and the result. Reciproquely, if $P, \nu \circ e \models_{M_K} \mathcal{A}$, then there is a, P' such that $P \equiv a[P']$, $P', \nu \circ e \models_{M_K} \mathcal{A}'$, and $\nu \circ e(x) = a$. So $P \mid \mathbf{val}(e,\nu,w)_K \longrightarrow P' \mid \mathsf{Val}' \mid \mathsf{rowopen}(a, K+e(x)-1)$, and $P' \mid \mathsf{Val}' \mid \mathsf{rowopen}(a, K+e(x)) \models [\![\mathcal{A}']\!]_{e,w}$ by induction, that is $P \mid \mathbf{val}(e,\nu,w)_K \models [\![\mathcal{A}]\!]_{e,w}$.

The π-calculus

Without loss of generality, we consider a fragment of the π-calculus, the choice-free finite synchronous π-calculus [23], with abstract syntax given by:

$$P \quad ::= \quad n(m).\,P \ \Big|\ \overline{n}m.\,P \ \Big|\ (\nu n)P \ \Big|\ P \mid P \ \Big|\ \mathbf{0}$$

where n, m, p ranges over the set of pure names Λ. We assume defined in the standard way the relation $P \to P'$ of reduction, and the relation of $P \stackrel{\alpha}{\longrightarrow} P'$

of labeled transition, over the set of labels τ (internal reduction), $\underline{n}m$ (input), and $\overline{n}m$ (output). We will adopt the convention that the case $P \xrightarrow{\overline{n}m} P'$ where $m \notin \mathrm{fn}(P)$ corresponds to a bound output (usually written $\overline{n}(m)$), and the case $P \xrightarrow{\underline{n}m} P'$ where $m \notin \mathrm{fn}(P)$ corresponds to a bound input (usually written $n(m)$).

We consider the logics $\mathcal{L}_{spat}$ and $\mathcal{L}_{mod}$ exactly as defined in Section 1, but where we now consider quantifiers and modalities to range over *commitments* $n, \overline{n} \in \mathsf{C}$, where $n \in \Lambda$. Semantics for the $\mathcal{L}_{mod}$ specific connectives interpreted over the π-calculus is defined as expected, with relation to a valuation $v : \mathsf{X} \to \mathsf{C}$. We thus define

$$P, v \models_M \exists x.\, \mathcal{A} \quad \textit{if } \exists \alpha \in \mathsf{C}.\ P, (v\{x \leftarrow \alpha\}) \models_M \mathcal{A}$$

$$P, v \models_M \langle x \rangle.\, \mathcal{A} \quad \textit{if } \exists P', n \in \Lambda.\ P \xrightarrow{v(x)n} P' \textit{ and } P', v \models_M \mathcal{A}$$

Hence, our action modalities only observe the subject of π-calculus actions. To embed $\mathcal{L}_{mod}$ into $\mathcal{L}_{spat}$ in the case of the π-calculus, we use some slightly different (when compared with the one developed in Section 2) notions of thread and row process. These notion, defined below, are completely equivalent for our purposes as the ones previously given for CCS and Mobile Ambients, but build on some particularities of the π-calculus model.

Instead of defining a thread of size k "syntactically", as in Section 2, in the current setting we consider a thread of size k to be a π-calculus process P satisfying the following property:

Every labeled transition sequence of maximal length for P is of the form

$$P = P_0 \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} P_2 \to \cdots \xrightarrow{\alpha_k} P_k$$

where $\alpha_i \neq \tau$ and $P_i \models 1$, for all $i = 0, \ldots, k-1$, and $P_k \models 0$.

By a labeled transition sequence of maximal length we mean a transition sequence that cannot be further extended by a labeled transition step. Notice that this notion of thread allows for some nondeterminism, in the sense that a thread of size k can offer two different sequences of k actions. This does not interfere with our intended usage for threads, namely the definition of row processes. We consider a *row* of size k on the commitment α to be a π-calculus process P satisfying the following property:

Every labeled transition sequence of maximal length for P is of the form

$$P = P_0 \xrightarrow{\alpha} P_1 \xrightarrow{\alpha} P_2 \to \cdots \xrightarrow{\alpha} P_k$$

where $P_i \models 1$, for all $i = 0, \ldots, k-1$, and $P_k \models 0$.

Rows can be defined by filtering threads on the same action, but the π-calculus can admit quite complex implementations for threads, due to the presence of name restriction. Therefore, it seems quite difficult to characterize thread and row processes in the π-calculus "syntactically", but we can do it logically. We start by defining the following abbreviations.

$$
\begin{array}{ll}
\Box\mathcal{A} & \triangleq \neg \Diamond \neg \mathcal{A} \\
\mathsf{piAct} & \triangleq 1 \wedge \Box\bot \wedge (1 \blacktriangleright \Diamond 0) \\
\mathsf{piThr}(0) & \triangleq 0 \\
\mathsf{piThr}(k+1) & \triangleq 1 \wedge \Box\bot \wedge (\mathsf{piAct} \blacktriangleright \Diamond \mathsf{piThr}(k)) \wedge (\mathsf{piAct} \triangleright \Box \mathsf{piThr}(k))
\end{array}
$$

$$
\begin{array}{ll}
\mathsf{AllAct} & \triangleq (1 \Rightarrow \mathsf{piAct})^{\forall} \\
\mathsf{inact} & \triangleq (\Box\bot) \triangleright \Box\bot \\
\mathsf{Shh} & \triangleq 1 \wedge \Box\bot \wedge (\mathsf{piAct} \blacktriangleright \Diamond(1 \wedge \mathsf{inact})) \\
\mathsf{Equals} & \triangleq \mathsf{AllAct} \ \wedge \ \Big(\mathsf{Shh} \blacktriangleright \big((\mathsf{Shh} \mid 1) \Rightarrow \Diamond\top\big)^{\forall}\Big) \\
\mathsf{Reds}(0) & \triangleq 0 \\
\mathsf{Reds}(n+1) & \triangleq \Diamond \mathsf{Reds}(n) \\
\mathsf{piRow}(n) & \triangleq \mathsf{piThr}(n) \wedge ((\mathsf{AllAct} \wedge \Box\bot \wedge \neg \mathsf{Equals}) \ \triangleright \ \neg \mathsf{Reds}(n))
\end{array}
$$

By analyzing the previous definitions, we can show that $P \models \mathsf{piRow}(k)$ if and only if P is a row of size k on some action α, and $P \models \mathsf{piThr}(k)$ if and only if P is a thread of size k. Hence, the constructions above gives us suitable notions of row process and thread process of depth k in the π-calculus.

From these ingredients, we can then develop a counterpart of Theorem 2.1; given the previous definitions specific for the π-calculus model, the encoding of $\mathcal{L}_{mod}$ into $\mathcal{L}_{spat}$ is the same as the one in Fig 3. We can then obtain results analogous to those presented in Section 4 and 5.

7 Concluding Remarks

We have studied a core spatial logic for concurrency, aiming at a better understanding of the relative role of the very basic logical operations present in most logics of this family. In particular, we have shown that quantifiers and action modalities can be embedded, and that the composition adjunct plays a key role in the expressiveness of this logic; these results allowed us to also prove its undecidability. Ours results are expected to hold for most process

calculi, even in the presence of recursion or replication. In this light, we believe that minimality of $\mathcal{L}_{spat}$ could be established in a precise sense.

The logics $\mathcal{L}_{spat}$ and $\mathcal{L}_{mod}$ have not been shown to have the same expressiveness in the strict technical sense. However, we believe this is the case for their extension with freshness quantifiers and a free name occurrence predicate. Since Theorem 3.3(4) does not hold for calculi with name restriction, an interesting issue is to get a better understanding of the (coarser) spatial equivalence in the absence of logical operations dealing with restricted names.

Although the composition adjunct operation is certainly important for general context/system specifications, our work shows that the automated verification of concurrent systems using spatial logics that make essential use of the composition adjunct seems to be unfeasible. An important issue is then whether other expressive and tractable forms of contextual reasoning inspired by the composition adjunct, and extending those already provided by decidable behavioral-spatial logics, can be identified.

We thank Hongseok Yang for the illuminating discussion that prompted our counterexample in Section 4. We acknowledge Luís Monteiro, Daniel Hirschkoff and Davide Sangiorgi for all the rich exchanges and encouragement; and Luca Cardelli for many related discussions. E. Jeandel provided some references about quantifier elimination. This collaboration was supported by FET IST 2001-33310 Profundis. E. Lozes was also funded by an "Eurodoc" grant from *Région Rhône Alpes*.

References

[1] S. Basu, R. Pollack, and M.-F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. In *IEEE Symposium on Foundations of Computer Science*, 1994.

[2] L. Caires. Behavioral and Spatial Properties in a Logic for the Pi-Calculus. In Igor Walukiwicz, editor, *Proc. of Foundations of Software Science and Computation Structures'2004*, number 2987 in Lecture Notes in Computer Science. Springer Verlag, 2004.

[3] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part II). In *CONCUR 2002 (13th International Conference)*, number 2421 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

[4] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part I). *Information and Computation*, 186(2):194–235, 2003.

[5] L. Caires and E. Lozes. Elimination of Quantifiers and Undecidability in Spatial Logics for Concurrency. Technical report, ENS-Lyon LIP Report, 2004.

[6] C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding Validity in a Spatial Logic of Trees. In *ACM Workshop on Types in Language Design and Implementation*, pages 62–73, New Orleans, USA, 2003. ACM Press.

[7] C. Calcagno, H. Yang, and O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In Hariharan, Mukund, and Vinay, editors, *Proc. of FST TCS'2001*, volume 2245 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[8] L. Cardelli, P. Gardner, and G. Ghelli. Manipulating Trees with Hidden Labels. In A. D. Gordon, editor, *Proceedings of the Sixth International Conference on Foundations of Software Science and Computation Structures (FoSSaCS '03)*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[9] L. Cardelli and G. Ghelli. A Query Language Based on the Ambient Logic. In D. Sands, editor, *10th European Symposium on Programming (ESOP 2001)*, volume 2028 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, 2001.

[10] L. Cardelli and A. Gordon. Logical Properties of Name Restriction. In S. Abramsky, editor, *Typed Lambda Calculi and Applications*, number 2044 in Lecture Notes in Computer Science. Springer-Verlag, 2001.

[11] L. Cardelli and A. D. Gordon. Anytime, Anywhere. Modal Logics for Mobile Ambients. In *27th ACM Symp. on Principles of Programming Languages*, pages 365–377. ACM, 2000.

[12] W. Charatonik, A. D. Gordon, and J.-M. Talbot. Finite-control mobile ambients. In D. Metayer, editor, *11th European Symposium on Programming (ESOP 2002)*, number 2305 in Lecture Notes in Computer Science. Springer-Verlag, 2002.

[13] W. Charatonik and J.-M. Talbot. The decidability of model checking mobile ambients. In *Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[14] G. Conforti and G. Ghelli. Decidability of Freshness, Undecidability of Revelation. In Igor Walukiwicz, editor, *Proc. of Foundations of Software Science and Computation Structures'2004*, number 2987 in Lecture Notes in Computer Science. Springer Verlag, 2004.

[15] L. Cardelli and A. Gordon Mobile Ambients. In *Proc. of FOSSACS*, 1998

[16] D. Hirschkoff. An Extensional Spatial Logic for Mobile Processes. In *CONCUR 2004 (15th International Conference)*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[17] D. Hirschkoff, E. Lozes, and D. Sangiorgi. Separability, Expressiveness and Decidability in the Ambient Logic. In *Third Annual Symposium on Logic in Computer Science*, Copenhagen, Denmark, 2002. IEEE Computer Society.

[18] D. Hirschkoff, E. Lozes, and D. Sangiorgi. Minimality results for the spatial logics. In *Proc. FSTTCS'2003*, number 2914 in Lecture Notes in Computer Science. Springer Verlag, 2003.

[19] E. Lozes. Adjunct elimination in the static Ambient Logic. In *Proc. of EXPRESS'2003*, 2003. to appear in ENTCS, Elsevier.

[20] P. O'Hearn. Resources, Concurrency, and Local Reasoning (Abstract). In D. Schmidt, editor, *Proc. of ESOP'2004*, Lecture Notes in Computer Science, pages 1–2. Springer, 2004.

[21] J. C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Seventieth Annual Symposium on Logic in Computer Science*, Copenhagen, Denmark, 2002. IEEE Computer Society.

[22] D. Sangiorgi. Extensionality and Intensionality of the Ambient Logics. In *28th Annual Symposium on Principles of Programming Languages*, pages 4–13. ACM, 2001.

[23] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes* Cambridge University Press, 2001.

[24] B.A. Trakhtenbrot. The impossibility of an algorithm for the decision problem for finite models. *Dokłady Akademii Nauk SSR*, pages 70:569–572, 1950.