Elimination of Quantifiers and Undecidability in Spatial Logics for Concurrency

Luís Caires[1] and Étienne Lozes[2]

[1] Departamento de Informática FCT/UNL, Lisboa, Portugal
[2] LIP, École Normal Supérieure 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 quantifiers, 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. Many key properties and concepts related to distributed systems, like locations, resources, independence, distribution, connectivity, and freshness, can be explained in spatial terms. The central idea behind spatial logics is that for specificying 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 of the kind mentioned above. Spatial logics have been proposed for π-calculi [2, 4, 3], and for the ambient calculus [10, 9]. Spatial logics for manipulating and querying semi-structured data have also been developed [8, 7]. Closely related are the separation logics [19, 18], 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 on purely spatial observations, will be referred from now on 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 $\mathcal{A}$ and the other satisfying $\mathcal{B}$. For the adjunct, $\mathcal{A} \triangleright \mathcal{B}$ is satisfied by those processes that, whenever composed with a process satisfying $\mathcal{A}$, are guaranteed to satisfy $\mathcal{B}$. A simple example of a property combining spatial and dynamic operators is the one expressed by the formula $(\neg\, 0 \mid \neg\, 0) \wedge \Diamond 0$; it specifies those processes that have (at least) two separate components and may reduce to the empty system. Adjuncts allow the specification of contextual properties, *e.g.*, consider the formula $1 \wedge (1 \blacktriangleright \Diamond 0)$, that uses the existential version of the adjunct defined $\mathcal{A} \blacktriangleright \mathcal{B} \triangleq \neg\,(\mathcal{A} \triangleright \neg\,\mathcal{B})$. This formula specifies the single-thread processes that can be composed with some other process to yield a system than may evolve to the empty system, after a single reduction step. Adjunct-free spatial logics with behavioral observations (*e.g.,* [1]) are also able to render some kinds of contextual properties. For example, the property just presented can be expressed by the formula $1 \wedge \exists x. \langle x \rangle 0$, using an action modality. Thus, one of the motivations for this work is to get a deeper understanding about the relative expressiveness of these approaches.

For the sake of simplicity and generality, we interpret $\mathcal{L}_{spat}$ in a rather small fragment of choice-free CCS. 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, $\mathcal{L}_{spat}$ seems quite weak, as far as expressiveness is concerned, when compared to other spatial logics. For instance, it provides no constructs referring to names or actions (like *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. 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. Still, due to the presence of the $\Diamond$ operator, the logic is able to make some distinctions between actions, and substitution of actions 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.

However, this question turns out to be a rather difficult one to answer, due to the presence of the composition adjunct operator $\triangleright$. The adjunct is quite powerful, allowing the logic to perform quite strong observations on processes. With adjunct, validity can be internally defined [10] (thus validity-checking is subsumed by model-checking), and use certain forms of specification akin to a comprehension principle (for example, we may specify the set of all processes that have an even number of parallel components). The study of expressiveness for spatial logics usually goes through the definition of an adequate spatial bisimilarity $\approx$ along the lines of [15]. Then, 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 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 approach does not 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. Results are known for some cases, where the logic includes just $\triangleright$ or $\Diamond$ [10, 1], 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 [1]. On the other hand, in the absence of name quantifiers and name revelation it is also known that static fragments are decidable [5], so there could be some hope in obtaining decidability of model-checking the whole of $\mathcal{L}_{spat}$.

We may answer these questions 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, and one may adapt the results of [12] to derive the undecidability of model-checking. But even if $\triangleright$ induces undecidability, we may ask the question of its actual contribution to the expressiveness of the logic. In previous work [17], Lozes has shown that in static spatial logics, that is spatial logics without quantifiers and dynamic operators, 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. An interesting question is then whether something similar happens in $\mathcal{L}_{mod}$: we could possibly think that the expressive power of the adjunct could somehow be recovered by the presence of action modalities, given that both kinds of constructs allow some contextual observations to be made.

So $\mathcal{L}_{mod}$ 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 key 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. 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 expressiveness 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 [17] about whether this connective could also be eliminated in spatial logics for concurrency.

Related Work. Sangiorgi first showed [21] 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 be generalized to other calculi [4, 16]. 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 is being developed by Hirschkoff, characterizing π-calculus behavioral equivalences with a logic with composition adjunct [14].

Adjunct elimination for a static spatial logic was first proved in [17], 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 hidden names and related logical connectives. The counterexample presented here 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 [10]. Validity and model-checking of ambient calculus against spatial logics with existential quantifiers was shown undecidable by Charatonik and Talbot [12]. The same authors also extended the results of [10] to logics with constructs for restricted names, and then with Gordon to the finite-control ambient-calculus [11]. Model-checking the π-calculus against full adjunct-free spatial logic with behavioral modalities, hidden and fresh name quantifiers, and recursion was shown to be decidable in [1]. Decidability of validity in a static spatial logic for trees with adjunct was first shown by Calcagno, Cardelli and Gordon in [5], building on techniques of [6]. More recently, Conforti and Ghelli proved that similar results do not hold in logics with operators for restricted names [13].

To our best knowledge, no results about expressiveness and decidability of dynamic spatial logics so crisp as ours have been presented, 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 [20]). However, we believe 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 spatial logics considered in this work. For the process calculus, we pick a fairly small fragment of CCS.

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

Fig. 1. Semantics of formulas

Definition 1.1. *Assume given an infinite set* A *of* actions*, ranged over by* α, β*.* Processes *are defined by the grammar:* $P, Q, R \ ::= \ \mathbf{0} \ \mid \ P \mid Q \ \mid \ \alpha. P.$

Actions are given in pairs of distinct (co)actions, characterized by the involution **co** : $\mathsf{A} \rightarrow \mathsf{A}$ 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 spatial structure of processes. Dynamics 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*

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

Notice that $\xrightarrow{\alpha}$ is closed under $\equiv$, and that $\xrightarrow{\tau}$ corresponds to the usual relation of *reduction*, noted $\longrightarrow$. We define the *depth* of a process P (maximal nesting of actions in a process P) by letting $\mathsf{ds}(\mathbf{0}) = 0$, $\mathsf{ds}(\alpha. P) = 1 + \mathsf{ds}(P)$, and $\mathsf{ds}(P \mid Q) = \max(\mathsf{ds}(P), \mathsf{ds}(Q))$. Let M_K denote the set of all processes whose depth does not exceed K: $M_K \triangleq \{P \mid \mathsf{ds}(P) \leq K\}$. Then $M_\infty \triangleq \bigcup_{k \in \mathbb{N}} M_k$ coincides with the set of all processes. We also define the projection (by truncation) $\pi_k : M_\infty \rightarrow M_k$, by induction on k by letting $\pi_0(P) = \pi_k(\mathbf{0}) \triangleq \mathbf{0}$, $\pi_k(P \mid Q) \triangleq \pi_k(P) \mid \pi_k(Q)$, and $\pi_{k+1}(\alpha. P) \triangleq \alpha. \pi_k(P)$.

Having defined the intended process model, we turn to logics. The 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 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 quantifiers ranging over actions.

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

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

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 in Fig. 1 by a relation of satisfaction. *Satisfaction* is expressed by $P, v \models_M \mathcal{A}$ 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

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

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

Fig. 2. Definition and semantics of derived operators.

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 clause for $\triangleright$). 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 abreviate $P \models \mathcal{A}$ for $P \models_{M_\infty} \mathcal{A}$.

An *action permutation* is a bijection $\sigma : \mathsf{A} \to \mathsf{A}$ such that $\sigma(\overline{\alpha}) = \overline{\sigma(\alpha)}$. We write $\{\alpha \leftrightarrow \beta\}$ for the action permutation that swaps α and β. Satisfaction verifies the fundamental property of equivariance, which in our present setting is formulated as follows.

Definition 1.4. *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 $P \equiv \sigma(Q)$.*

Proposition 1.5 (Equivariance). *Let $P, v \models_M \mathcal{A}$. For every action permutation σ, if $P \equiv \sigma(Q)$ then $Q, \sigma(v) \models_M \mathcal{A}$.*

We frequently refer to the logical equivalence of processes induced by the logic L (where L is either $\mathcal{L}_{spat}$ or $\mathcal{L}_{mod}$). The relation $=_{\mathrm{L}}$ is defined by setting $P =_{\mathrm{L}} Q$ if 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. By $w(\mathcal{A})$ we denote the maximal level of nesting of composition $|$ in the formula $\mathcal{A}$, and by $\mathsf{ds}(\mathcal{A})$ the maximal nesting of dynamic modalities in the formula $\mathcal{A}$, defined by

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

It is easy to see that a formula $\mathcal{A}$ cannot inspect the part of the process that lies deeper than the depth of $\mathcal{A}$. As a consequence, the restriction to M_k of the denotation of a formula of depth k completely charaterizes its denotation, in the precise sense of:

Proposition 1.6 (Depth finiteness). *For all formulas $\mathcal{A} \in \mathcal{L}_{mod}$, for all $k > \mathsf{ds}(\mathcal{A})$, and for all processes P,*

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

Notations. 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 that, quite surprisingly, the logic $\mathcal{L}_{mod}$, which contains quantifiers and variables, can be embedded into the core logic $\mathcal{L}_{spat}$, which does not seem to contain related constructs, in the sense of the following main result:

Theorem 2.1. *For any closed formula $\mathcal{A} \in \mathcal{L}_{mod}$ and any natural number $K > \mathsf{ds}(\mathcal{A})$, we can efectively construct a formula $[\![\mathcal{A}]\!]_K \in \mathcal{L}_{spat}$ such that for all processes P:*

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

Notice that this result does not state that $\mathcal{L}_{spat}$ and $\mathcal{L}_{mod}$ have the same expressiveness in the usual sense, however, we should note that the denotation of a formula $\mathcal{A}$ is completely characterized by its denotation on some subset of the models M_k, in the sense of Proposition 1.6. 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 separability of $\mathcal{L}_{spat}$, and independence of the composition adjunct.

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 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 we get a better understanding about what can be expressed in $\mathcal{L}_{spat}$, in a sometimes not really obvious way.

We want to reduce 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 certain process $\mathbf{val}(e, \nu, w)_K$, to be composed with the process P being tested for satisfaction. More concretely, we encode the pair P, v by a process of the form $P \mid \mathbf{val}(e, \nu, w)_K$, where $\mathbf{val}(e, \nu, w)_K$ encodes the valuation, and $\nu \circ e = v$ is a decomposition of the valuation v into certain maps $e : \mathsf{X} \to \mathbb{N}$ and $\nu : \mathbb{N} \to \mathsf{A}$, respectively called *environment* and *naming*, and w is a natural number. The role of these data will be explained below.

The encoding of valuations makes use of the notion of *row* process. A row process $\mathbf{row}(n, \alpha)$ is a sequential process 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 logically, and we will use rows to represent bindings between variables (represented by rows of different length) and actions α. Moreover, by imposing a bound K on the depth of the process P one considers, we can easily separate the valuation part from the process that represents the "real" model, in the "soup" $P \mid \mathbf{val}(e, \nu, w)_K$.

We start by introducing formulas whose models are precisely the sequential threads with a given number of actions, in the way we also define the derived modality $?. \mathcal{A}$.

$$\begin{array}{llll} 1 & \triangleq 0 \wedge (0 \,\|\, 0) & \mathsf{Thread}(1) & \triangleq 1 \wedge (1 \blacktriangleright \Diamond 0) \\ ?. \mathcal{A} & \triangleq 1 \wedge (\mathsf{Thread}(1) \blacktriangleright \Diamond \mathcal{A}) & \mathsf{Thread}(n+1) & \triangleq ?. \mathsf{Thread}(n) \end{array}$$

We have

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

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

We now give (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 define a formula $\mathsf{Equals}(k)$ that is satisfied by the of processes which belong to M_k, and are compositions of guarded processes all with the *same* first action. We may then specify rows using appropriate formulas

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

We now prove

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

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

We can now explain our encoding of a valuation v into a certain process. First, we decompose v into two functions ν and e such that $v \ = \ \nu \circ e$. An *environment* e is a partial injective function from variables to an initial segment $[1, \ldots, n]$ of the natural numbers. We note by $e\{x{\leftarrow}n\}$ the extension of e with $x \ \mapsto \ n$, and $\mid \ e \ \mid$ is the maximal value of e, that, is the number of variables already allocated. A *naming* ν is a function from $[1, \ldots, n]$ to A. Notice that the decomposition $v \ = \ \nu \circ e$ is not unique, but will be given by the order in which existential quantified variables are introduced in their scopes. For any naming ν and environment e the process $\mathbf{val}(e, \nu, w)_K$ is

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

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 $\mid$ in the source formulas. Since interpreting $\mid$ also splits the (encoding of the) valuation, we have to provide enough copies (2^w, where w is related to $w(\mathcal{A})$). Note that we can always filter out any undesirable interference of $\mathbf{val}(e, \nu, w)_K$ with the parallel process P, since for any labeled-transition reduct Q of $\mathbf{val}(e, \nu, w)_K$, Q is not an environment since it does not have the right number of rows for each depth. Likewise, for any namings ν, ν', ν'', we have $\mathbf{val}(e, \nu, w+1)_K \ \equiv \ \mathbf{val}(e, \nu', w)_K \ \mid \ \mathbf{val}(e, \nu'', w)_K$ if and only if $\nu = \nu' = \nu''$. Using already defined properties, we set

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

Lemma 2.4. *For any process P, environment e and naturals $K, w \geq 1$*

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

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

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

The formula $\mathsf{ProcVal}(e,w)_K$ specifies a pair process-valuation, where the process belongs to M_K. Now we introduce formulas to match specific entries of the (encoding of the) valuation: selection of the action α associated to the variable x is achieved by filtering the set of row processes of depth $e(x)$. To implement this properties we define the following formulas:

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

$\mathsf{XRow}(x,e)_K$ allows us to select one of the rows that represents the environment entry of the variable x. $\mathsf{UsedXRow}(x,e)_K$ checks that such a row has lost an action prefix (after a reduction step takes place). $\mathsf{EnvX}(x,e,w)_K$ matches all the rows that encode the environment entry for the variable x. 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 bigger than any other (with $\mathsf{Test}(e)$), and then check (using $\Diamond$) that it may react with some row of depth $e(x)$ (with $\mathsf{UsedTest}(e)$). 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 2.5. *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 in Fig. 3.*

Theorem 2.1 follows from Lemmas 2.2, 2.3, 2.4, and the following general result:

Lemma 2.6 (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. (Sketch, see appendix for details) By induction on $\mathcal{A}$. For the connectives of $\mathcal{L}_{spat}$, the encoding is quite natural: in the case of $\mid$, the environment is split in two equal parts, and tested for a sound recombination by $\mathsf{ProcVal}(e,w)_K$. For $\triangleright$, we must check that the composition of the two environments coming from the left and right of $\triangleright$ is actually an environment. This holds if both environments are defined with the same naming ν. For the case of $\Diamond$, any reduction involving the environment is excluded, because otherwise the resulting environment would be ill-formed. For the other connectives, the encoding also involves our abbreviations: the encoding of the quantifier $\exists x.\,\mathcal{A}$ relies on representing the quantification over actions into a quantification (using $\blacktriangleright$) over processes that represent environment entries. For action modalities, one checks for interactions between the process and a row corresponding to the selected variable.

We can thus present the proof of 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 $|$ connectives in $\mathcal{A}$. Then $\pi_K(P) \equiv \pi_K(P) \mid \mathbf{val}(\emptyset, \emptyset, w)_K$, so by Lemma 2.6, $\pi_K(P), \emptyset \models_{M_\infty} [\![\mathcal{A}]\!]_K$ 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 1.6.

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 $\{x_{\alpha_1}, \ldots, x_{\alpha_n}\}$. We have

$$\begin{array}{ll} \chi(\mathbf{0}) \triangleq 0 & \chi(\alpha.\,P) \triangleq 1 \wedge \langle x_\alpha \rangle \chi(P) \\ \chi(\overline{\alpha}.\,P) \triangleq 1 \wedge \langle \overline{x_\alpha} \rangle \chi(P) & \chi(P \mid Q) \triangleq \chi(P) \mid \chi(Q) \end{array}$$

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

where $K = \mathsf{ds}(P)$. Recall that $x = y$ and $x \neq y$ are defined in Fig.2, and notice 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. Induction on P (see appendix).

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

Proof. By Lemma 3.1 and Theorem 2.1 (see appendix).

We then conclude:

Theorem 3.3. *The following statements are equivalent:*

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

Proof. (1)$\Rightarrow$(2) because $\mathcal{L}_{spat} \subset \mathcal{L}_{mod}$, (2)$\Rightarrow$(3) since $C(P) \in \mathcal{L}_{spat}$ and $P \models C(P)$, (3)$\Rightarrow$(4) by Lemma 3.2, and (4)$\Rightarrow$(1) by Proposition 1.5.

4 Expressiveness of Composition Adjunct

It is known that in static spatial logics, that is spatial logics without quantifiers and dynamic operators, the adjunct connective 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 [17]. It is not hard to see that adjunct cannot be dispensed with in $\mathcal{L}_{spat}$, because without adjunct one is not allowed to distinguish threads of different lenght: if we pick $\mathcal{A} \in \mathcal{L}_{spat} - \{\triangleright\}$, we can verify by an easy induction on $\mathcal{A}$ 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 does not hold for the spatial logic $\mathcal{L}_{mod}$. For this, we adapt a scheme suggested by 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, on the other hand, we show that parity cannot be characterized by adjunct-free formulas. We start by defining the following formulas (where $\Box\mathcal{A} \triangleq \neg\,\Diamond\neg\,\mathcal{A}$):

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

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

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

Hence $P \models \mathsf{Even2}$ if and only if $P \equiv \alpha_1.\,(\beta_{1,1} \mid \beta_{1,2}) \mid \cdots \mid \alpha_k.\,(\beta_{k,1} \mid \beta_{k,2})$ 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 (in $P \mid Q$) that the actions that occur in the toplevel of P are exactly the same that appear in the second level of Q using the formula Same:

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

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 $\mathsf{Even} \triangleq \mathsf{Fam} \wedge (\mathsf{Even2} \blacktriangleright \mathsf{Same})$.

A key observation 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 this 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:

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

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

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

Fig. 4. First-Order Logic

Given a family P and a valuation v, we write $P\backslash v$ for the subfamily of P of the actions α that do not appear in the codomain of the valuation v. More precisely, we define $P\backslash v \triangleq \prod\{\alpha \;\mid\; P \equiv \alpha \mid Q \textit{ and } \alpha, \overline{\alpha} \notin \mathsf{codom}(v)\}$. We then have:

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 $(P \mid \alpha)$ 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*

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

Proof. By induction on $\mathcal{A}$ (see appendix).

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

Proof. By contradiction: if $\mathcal{A}$ was a such formula, then we may take a family P and an extended family $P \mid \alpha$ with $w(P) \geq \mathsf{sn}(\mathcal{A})$. Then by previous lemma, $P, \emptyset \models \mathcal{A}$ if and only if $P \mid \alpha, \emptyset \models \mathcal{A}$, which is a contradiction.

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

5 Undecidability

In this section, we show that the validity-, satisfiability- and model-checking problems for the logic $\mathcal{L}_{spat}$ (and hence $\mathcal{L}_{mod}$) are all undecidable. These results are a consequence of our embedding of $\mathcal{L}_{mod}$ into $\mathcal{L}_{spat}$ (Theorem 2.1), and the fact that first-order logic can then be easily encoded into $\mathcal{L}_{mod}$ along the lines of [12]. The language of first-order logic (FOL) and its semantics if defined as usual (see Fig. 4). Formulas are build from a set of $Vars$ of individual variables (x, y), and finite set $Pred$ of predicate symbols (p, q). For simplicity, we consider but binary predicate symbols. 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 mapping assigning to each predicate symbol $p \in Pred$ a binary relation $I(p) \subseteq D \times D$. For our purposes it is enough to focus on finite models. Satisfaction of a FOL formula in a model (D, I) is defined in Fig. 4, using a valuation v that assigns each individual variable an element of D.

We now show how to faithfully 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 a natural number $E > 1$, and assign to each predicate symbol $p \in Preds$ a distinct natural number $Code(p) > E$. We also fix K such that $K > Code(p)$, for all $p \in Preds$. To encode a model (D, I) into a process $\mathcal{M}[\![(D, I)]\!]$, we start by assigning 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 $(d, e) \in I(p)$ by the process

$$\mathcal{T}(p)[\![(d, e)]\!] \triangleq \alpha.\, (\mathbf{row}(Code(p), \beta) \mid A(d).\, \mathbf{0} \mid A(e).\, A(e).\, \mathbf{0})$$

where α, β are some actions. We then let $\mathcal{I}[\![I]\!] \triangleq \prod_{p \in Preds} \prod_{(d,e) \in I(p)} \mathcal{T}(p)[\![(d,e)]\!]$ and then set $\mathcal{M}[\![(D,I)]\!] \triangleq \mathcal{D}[\![D]\!] \mid \mathcal{I}[\![I]\!]$. By construction, we always have $\mathcal{M}[\![(D,I)]\!] \in M_K$. The processes representing FOL models as we have just defined can be logically characterized by a formula Model of $\mathcal{L}_{spat}$ as follows:

$$\begin{array}{llll}
\mathsf{Dom}(x) & \triangleq \left(\mathsf{Row}(E) \wedge \langle x \rangle \top\right)^{\exists} & \mathsf{Diff} & \triangleq \forall x. \forall y. (\langle x \rangle \top \mid \langle y \rangle \top \mid \top) \Rightarrow x \neq y) \\
\mathsf{Domain} & \triangleq \mathsf{Diff} \wedge \left(1 \Rightarrow \exists x. \mathsf{Dom}(x)\right)^{\forall} & \mathsf{Interp} & \triangleq \left(1 \Rightarrow \exists z. \langle z \rangle (\mathsf{Some} \mid \mathsf{Row}(1) \mid \mathsf{Row}(2))\right)^{\forall} \\
\mathsf{Some} & \triangleq \bigvee_{p \in Preds} \mathsf{Row}(Code(p)) & \mathsf{Compat} & \triangleq \forall x. ((\exists z. \langle z \rangle (\langle x \rangle \top \mid \mathsf{Some}))^{\exists} \Rightarrow \mathsf{Dom}(x)^{\exists}) \\
 & & \mathsf{Model} & \triangleq \mathcal{M}_K \wedge [\![(\mathsf{Domain} \mid \mathsf{Interp}) \wedge \mathsf{Compat}]\!]_K
\end{array}$$

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

Proof. Interpreting the formula Model (see appendix).

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

$$\begin{array}{llll}
\mathcal{F}[\![\neg \mathcal{A}]\!] & \triangleq \neg [\![\mathcal{A}]\!] & \mathcal{F}[\![\exists x. \mathcal{A}]\!] & \triangleq \exists x. (\mathsf{Dom}(x) \wedge A) \\
\mathcal{F}[\![\mathcal{A} \wedge \mathcal{B}]\!] & \triangleq \mathcal{F}[\![\mathcal{A}]\!] \wedge \mathcal{F}[\![\mathcal{B}]\!] & \mathcal{F}[\![p(x,y)]\!] & \triangleq \left(\exists z. \langle z \rangle (\mathsf{Row}(Code(p)) \mid \langle x \rangle 0 \mid \langle y \rangle \langle y \rangle 0)\right)^{\forall}
\end{array}$$

Finally, for valuations we set $\mathcal{V}[v](x) = A(v(x))$. We can then show

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. See appendix.

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. By Lemma 5.2, Lemma 5.1 and Theorem 2.1 (see Appendix).

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 [22].

6 Extension to the π-Calculus and 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 [10] 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 ambients: for instance, the formula $1 \wedge \Diamond \top$ may be satisfied by the ambient $n[P]$, but not by the guarded ambient 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.*

Proof: See Appendix.
We should remark that Takhtenbrot also allows us to conclude that there is no complete proof system for validity of $\mathcal{L}_{spat}$ formulas over any of these calculi.

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 operation 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. In this light, we believe that minimality of $\mathcal{L}_{spat}$ could be established in a precise sense. $\mathcal{L}_{spat}$ 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 does not hold for calculi with name restriction, an interesting issue is to get a better understanding of the (coarser) spatial equivalence in the absense of logical operations dealing with restricted names.

Although the composition adjunct is certainly important for general context/system specifications, our work shows that the automated verification of concurrent systems using logics that rely on the composition adjunct seems to be not feasible. 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 behavioral-spatial logics can be identified.

We thank Hongseok Yang for the illuminating discussion that prompted our counterexample in Section 5. We thank 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. 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.
2. L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part I). In N. Kobayashi and B.C. Pierce, editors, *10th Symposium on Theoretical Aspects of Computer Science*, volume 2215 of *Lecture Notes in Computer Science*, pages 1–30. Springer-Verlag, 2001.
3. L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part II). In *CONCUR 2002 (13th International Conference)*, 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. 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.
6. C. Calcagno, H. Yang, and O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In R. Hariharan, M. Mukund, and V. Vinay, editors, *FSTTCS'2001*, volume 2245. Springer-Verlag, 2001.
7. L. Cardelli, P. Gardner, and G. Ghelli. Manipulating Trees with Hidden Labels. In A. D. Gordon, editor, *Proceedings of the First International Conference on Foundations of Software Science and Computation Structures (FoSSaCS '03)*, Lecture Notes in Computer Science. Springer-Verlag, 2003.
8. 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.
9. 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.
10. 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.
11. 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.

12. 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.
13. 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.
14. D. Hirschkoff. An Extensional Spatial Logic for Mobile Processes, 2004.
15. 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.
16. D. Hirschkoff, E. Lozes, and D. Sangiorgi. Minimality results for the spatial logics. In *Proc. of FSTTCS'2003*, LNCS. Springer Verlag, 2003.
17. E. Lozes. Adjunct elimination in the static Ambient Logic. In *Proc. of EXPRESS'2003*, 2003. to appear in ENTCS, Elsevier.
18. 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.
19. J. C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Third Annual Symposium on Logic in Computer Science*, Copenhagen, Denmark, 2002. IEEE Computer Society.
20. M.-F. Roy S. Basu, R. Pollack. On the combinatorial and algebraic complexity ofquantifier elimination. volume IEEE Symposium on Foundations of Computer Science, 1994.
21. D. Sangiorgi. Extensionality and Intensionality of the Ambient Logics. In *28th Annual Symposium on Principles of Programming Languages*, pages 4–13. ACM, 2001.
22. B.A. Trakhtenbrot. The impossibility of an algorithm for the decision problem for finite models. *Doklady Akademii Nauk SSR*, pages 70:569–572, 1950.

Appendix (Proofs)

For Section 2

Proof of Lemma 2.6

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 2.4, 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 do 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 $P \equiv \big(P_{1,a} \mid \mathbf{val}(e,w-1,)_K\big) \mid \big(P_{1,b} \mid \mathbf{val}(e,w-1,)_K\big) \models_{M_\infty} [\![\mathcal{A}]\!]_{(e,w)}$.
- (Case of $\mathcal{A} = \mathcal{A}_1 \triangleright \mathcal{A}_2$) Assume first $P, \emptyset \models_{M_\infty} [\![\mathcal{A}]\!]_{(e,w)}$. By Lemma 2.4, 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, $P \mid Q \mid \mathbf{val}(e,\nu,w)_K \equiv R_1 \mid \mathbf{val}(e,\nu',w)_K$ 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 $P_1 \mid \mathbf{val}(e, \nu, w)_K \models_{M_\infty} [\![\mathcal{A}]\!]$.

Proof of Lemma 3.1

Proof. Induction Hypothesis on 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, we have $Q \xrightarrow{\beta_j} Q'$. 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 $Q, v \models_{M_K} \chi(P)$.

Proof of Lemma 3.2

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 $Q, v \models_{M_K} \chi(A)$. Since the actions β_i are pairwise distinct, by Theorem 2.1 we conclude $Q \models C(P)$.

For Section 4

Proof of Lemma 4.2

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 1.5), 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 $P, v \models \mathcal{A}$.

- $\mathcal{A} = \Diamond \mathcal{A}_1$. Then both P and $P \mid \alpha$ do not satisfy $\mathcal{A}$, because they are deadlocked.
- $\mathcal{A} = \exists x. \mathcal{A}_1$. 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 1.5. 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 symetry, 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 $P, v \models \mathcal{A}$.
- $\mathcal{A} = \langle x \rangle \mathcal{A}_1$. 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 $P, v \models \mathcal{A}$.
- $\mathcal{A} = \langle \overline{x} \rangle. \mathcal{A}_1$ proceeds in the same way.

For Section 5

Proof Lemma of 5.1

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(only if), 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 $I(p) = \{(d,e) : \exists R, \alpha, \beta. P_I \equiv \alpha. (A(d). \mathbf{0} \mid A(e). A(e). \mathbf{0} \mid \mathbf{row}(Code(p), \beta)) \mid R\}$, for all $p \in Preds$. Since $P \models \mathsf{Compat}$ we indeed have $I(p) \subseteq D \times D$.

Proof of Lemma 5.2

Proof. By an easy 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

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

for some α, β and R. Hence we have

$$\mathcal{M}[\![(D,I)]\!], \mathcal{V}[\![v]\!] \models_{M_K} \big(\exists z.\, \langle z\rangle(\mathsf{Row}(Code(p)) \mid \langle x\rangle 0 \mid \langle y\rangle\langle y\rangle 0)\big)^{\forall}$$

Conversely, if $\mathcal{M}[\![(D,I)]\!], \mathcal{V}[\![v]\!] \models_{M_K} \mathcal{F}[\![p(x,y)]\!]$, we conclude that

$$\mathcal{M}[\![(D,I)]\!] \equiv \alpha.\, (\mathbf{row}(Code(p),\delta) \mid \beta.\,\mathbf{0} \mid \gamma.\,\gamma.\,\mathbf{0}) \mid R$$

for some R and $\alpha, \beta, \gamma, \delta$ such that $v(x) = \beta$ and $v(y) = \gamma$. We conclude that

$$\mathcal{I}[\![I]\!] \equiv \alpha.\, (\mathbf{row}(Code(p),\delta) \mid \beta.\,\mathbf{0} \mid \gamma.\,\gamma.\,\mathbf{0}) \mid R'$$

, and so there are $d, e \in D$ such that $A(d) = \beta$ and $A(e) = \gamma$ and $(d, e) \in I(p)$, by construction of $\mathcal{I}[\![I]\!]$. Hence $(D,I) \models_v p(x,y)$.

Proof of Theorem 5.3

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 $\mathcal{M}[\![(D,I)]\!] \models_{M_S} \mathcal{F}[\![\mathcal{A}]\!]$. 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 $(D,I) \models \mathcal{A}$.

For Section 6

Proof of Theorem 6.1

We only sketch our proof, since it is obtained by adapting to the ambient calculus and to the π-calculus the constructions and reasoning 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 then 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 submodels M_k.

Thus we just detail how to provide counterparts to these formulas, by taking in consideration each of the models 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 passing, it is slightly more involved.

Mobile Ambients

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

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

where a, n, m, p ranges over the set of names Λ. We assume given the reduction relation $P \to P'$ as defined in [10]. 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 $\mathcal{L}^{amb}_{mod}$ of $\mathcal{L}_{spat}$:

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

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, small differences only appear in encoding of modalities. We set

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

Fig. 5.

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

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

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

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 2.6 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 $Q \xrightarrow{\mathsf{open}\ a} Q'$. 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 $P \xrightarrow{\mathsf{open}\ a} P'$ 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

We consider the choice-free finite synchronous π-calculus, given by:

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

where n, m, p ranges over the set of names Λ, and we assume defined in the standard way the relation $P \to P'$ of reduction, and the relation of $P \xrightarrow{\alpha} P'$ of labeled transition, over the set of labels τ, $n(m)$, and $\overline{n}(m)$, where we assume that the case $P \xrightarrow{\overline{n}(m)} P'$ where $m \notin \mathrm{fn}(P)$ corresponds to a bound output. We then 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 $\mathcal{L}_{mod}$ specific connectives over the π-calculus is defined as expected, relative to a valuation $v : \mathsf{X} \to \mathsf{C}$. We have

$$\begin{array}{l} P, v \models_M \exists x.\,\mathcal{A} \ \ \textit{if}\ \exists \alpha \in \mathsf{C}.\ P, (v\{x \leftarrow \alpha\}) \models_M \mathcal{A} \\ P, v \models_M \langle x\rangle.\,\mathcal{A} \ \ \textit{if}\ \exists P', n \in \Lambda.\ P \xrightarrow{v(x)(n)} P' \textit{ and } P', v \models_M \mathcal{A} \\ P, v \models_M \langle \overline{x}\rangle.\,\mathcal{A} \ \ \textit{if}\ \exists P', n \in \Lambda.\ P \xrightarrow{\overline{v(x)}\langle n\rangle} P' \text{ and } P', v \models_M \mathcal{A} \end{array}$$

To embed $\mathcal{L}_{mod}$ into $\mathcal{L}_{spat}$ in the case of π, we use a slightly different (when compared with the one developed in Section 2) notion of row, but completely equivalent for our purposes, and that makes use of the particulars of the π-calculus model. So, instead of letting a row of size k be a sequential thread on the same action α, in this case we consider a row of size k holding the action α to be a π-process P such that the following hold:

- For all n, β, we have $P \xrightarrow{\beta} P_0 \to P_1 \to \cdots \to \mathcal{P}_n \to Q$ if and only if
  - $n = k$, $\beta = \alpha$, $Q \equiv \mathbf{0}$, and
  - For all $i \in \{0, \ldots, k\}$, there are no $\ell \neq \tau$ and R such that $P_i \xrightarrow{\ell} R$, and

- $P \models 1$, and for all $i \in \{0, \ldots, k\}$ we have $P_i \models 1$.

Processes P satisfying this specification are said to be rows of size k on α. Such processes exist in the π-calculus because of the possibility of synchronizations on restricted channels, *e.g.*,

$$\mathbf{row}(p, 2) \triangleq p(z).\,(\nu n)(\overline{n}.\,\mathbf{0} \mid n(x).\,(\nu m)(\overline{m}.\,\mathbf{0} \mid m(y).\,\mathbf{0}))$$

Of course, many other implementations exist, for example we could also have

$$\mathbf{row}(p, 2) \triangleq p(z).\,(\nu n)(\overline{n}(n).\,\mathbf{0} \mid n(x).\,(\overline{x}x.\,\mathbf{0} \mid x(y).\,\mathbf{0}))$$

It seems quite difficult to characterize all these processes "intensionally", but we can do it logically as follows (let $\Box\mathcal{A} \triangleq \neg\,\Diamond\neg\,\mathcal{A}$):

$$\begin{array}{ll}
\mathsf{piAct} & \triangleq 1 \blacktriangleright \Diamond 0 \\
\mathsf{piRow}(k) & \triangleq 1 \wedge \Box\bot \wedge (\mathsf{piAct} \blacktriangleright \Diamond\mathsf{piThread}(k)) \\
\mathsf{Grow} & \triangleq \neg\,0 \mid \neg\,0 \mid \neg\,0 \\
\mathsf{piNoExternal} & \triangleq ((1 \wedge \Box\bot) \,\triangleright\, \neg\,\Diamond\mathsf{Grow}) \\
\mathsf{piThread}(0) & \triangleq 0 \\
\mathsf{piThread}(n+1) & \triangleq 1 \wedge (\Diamond\top) \wedge \mathsf{piNoExternal} \wedge \Box\mathsf{piThread}(n)
\end{array}$$

We can then show that $P \models \mathsf{piRow}(k)$ if and only if P is a row of size k on some action α. We also have that $P_0 \models \mathsf{piThread}(k)$ whenever

- For all n, we have $P_0 \to P_1 \to \cdots \to \mathcal{P}_n \to Q$ if and only if
  - $n = k$, $Q \equiv \mathbf{0}$, and
  - For all $i \in \{0, \ldots, k\}$, $P_i \models 1$ and there are no $\ell \neq \tau$ and R such that $P_i \xrightarrow{\ell} R$.

The constructions above gives us suitable notions of row and thread of depth k. Notice that unlike for the rows defined in Section 2, the name associated to a row is just the first action (the other transitions of a row are always reductions). This is not a problem, because only the first action of a row is actually used in testing for the value of the variable it represents in the encoding of valuations.

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 identical to those presented in Section 4 and 5.