Minimality results for the spatial logics

D. Hirschkoff[1], É. Lozes[1], and D. Sangiorgi[2]

[1] ENS Lyon, France
[2] Università di Bologna, Italy

Abstract. A spatial logic consists of four groups of operators: standard propositional connectives; spatial operators; a temporal modality; calculus-specific operators. The calculus-specific operators talk about the capabilities of the processes of the calculus, that is, the process constructors through which a process can interact with its environment. We prove some minimality results for spatial logics. The main results show that in the logics for π-calculus and asynchronous π-calculus the calculus-specific operators can be eliminated. The results are presented under both the strong and the weak interpretations of the temportal modality. Our proof techniques are applicable to other spatial logics, so to eliminate some of – if not all – the calculus-specific operators. As an example of this, we consider the logic for the Ambient calculus, with the strong semantics.

1 Introduction

Over the last 15 years, a lot of research has gone into calculi of mobile processes. Among these, the π-calculus is the best known. A number of other calculi have been put forward to study aspects of mobility not directly covered by the π-calculus. Examples are: the Asynchronous π-calculus (Aπ), which uses asynchronous communications, that are more common in distributed systems than the synchronous communications of the π-calculus; the Ambient calculus and all its variants, which extend the π-calculus with localities and movements of these.

At present, one of the most active research directions in the area of process mobility is that of *spatial logics* [2, 3, 5, 6]. These logics are used to reason about, and express properties of, systems of mobile processes. The logics can describe both the spatial distribution of the processes and their temporal evolutions. A spatial logic consists of four groups of operators: standard propositional connectives; spatial operators; a temporal modality; calculus-specific operators. We briefly comment on them below.

The spatial operators allow us to express properties of the structure of a process. These operators include tensor, $|$, and linear implication, $\triangleright$. The former is used to separate a spatial structure into two parts: thus a process satisfies formula $\mathcal{A}_1 \mid \mathcal{A}_2$ if the process can be decomposed into two subsystems satisfying respectively $\mathcal{A}_1$ and $\mathcal{A}_2$. Operator $\triangleright$ is the adjunct of $|$: a process satisfies formula $\mathcal{A}_1 \triangleright \mathcal{A}_2$ if, whenever put in parallel with a process satisfying $\mathcal{A}_1$, the resulting system satisfies $\mathcal{A}_2$. Other spatial operators are the revelation operator ® and the freshness operator, $\reflectbox{N}$; a combination of these operators give us the logical counterpart of restriction, the construct used in calculi of mobile processes to create fresh names. Note that as an alternative to И, the standard universal quantifier on names, which is more powerful than И, can be included in $\mathcal{L}$. This is not needed for our purposes (except in Section 4 — see below).

The temporal modality, $\Diamond$, can be interpreted strongly or weakly: in the former case the number of reductions a process can perform is visible, in the latter case it is not. Calculus-specific operators talk about the capabilities of the processes of the calculus, that is, the process constructs through which a process can interact with its environment. In the π-calculus, input and output are the only capabilities. The output capability is the only primitive operator in the spatial logic for the π-calculus [2], the input capability formula being derivable.

An important property of a formalism is conciseness: the formalism should have a *small* set of independent operators. Conciseness helps when developing the theory of the formalism and when studying its expressiveness. We call a result that reduces the number of operators – in a logic, or in a calculus – a *minimality result*, in the sense that it helps going in the direction of a minimal language. This terminology should not be misunderstood: we are interested in getting a *smaller* language, without necessarily proving that we obtain the *smallest* possible one. A minimality result can be useful in tools and implementations. For instance, the possibility of encoding the operator of sum in the π-calculus [9] justifies its absence in Picts' abstract machine [10].

In this paper we prove some minimality results for spatial logics. Our main results show that, surprisingly, in the logics for π-calculus and Aπ all calculus-specific operators can be eliminated. These results hold both under the strong and under the weak semantics for $\Diamond$. The resulting *common core spatial logic*, $\mathcal{L}$, has the following grammar:

$$\mathcal{A} \quad ::= \quad \mathcal{A}_1 \wedge \mathcal{A}_2 \;\big|\; \neg\mathcal{A} \;\big|\; 0 \;\big|\; \mathcal{A}_1 \mid \mathcal{A}_2 \;\big|\; \mathcal{A}_1 \triangleright \mathcal{A}_2 \;\big|\; n \circledR \mathcal{A} \;\big|\; \text{И} n.\, \mathcal{A} \;\big|\; \Diamond\mathcal{A}\,.$$

Note that the operators of this logic give no information about the nature of computation (whether it is based on synchronisation, what values are exchanged, etc.). Further, it may be puzzling to see the same logic – same operators, same interpretation – for π-calculus and Aπ, because their behavioural theories are rather different. The point is that spatial logics are rather intensional, and do not agree with the standard behavioural theories. These logics allow us to observe the internal structure of the processes at a very fine-grained detail, much in the same way as structural congruence does [11, 7].

We do not claim that the common core logic $\mathcal{L}$ is universal, i.e., that it can be used on many or all calculi of mobile processes. We think that, usually, some calculus-specific operators are necessary. However we believe that our proof techniques are applicable to other spatial logics, so to eliminate some of – if not all – the calculus-specific operators. As an example of this, we consider also the case of Ambient-like calculi, under the strong semantics. The spatial logics for Ambients [5, 6] have two calculus-specific operators, called ambient and ambient adjunct. We can derive some of, but not all, the capability formulas of Ambients in $\mathcal{L}$. We can derive all of them if we add the ambient adjunct to $\mathcal{L}$: thus the ambient formula can be eliminated from the logic in [5].

Our results suggest that spatial logics are more expressive than standard modal logics. The latter do not have the spatial connectives. However, these logics have more precise temporal connectives, the *modalities*. The only modality of the spatial logics talks about the evolution of a system on its own. In standard modal logics, by contrast, modalities also talk about the potential interactions between a process and its environment. For instance, in the Hennessy-Milner logic the modality $\langle\alpha\rangle.\mathcal{A}$ is satisfied by the processes that can perform the action α and become a process that satisfies $\mathcal{A}$. The action α can be a reduction, but also an input or an output. The formulas for the modalities are similar to those for the capabilities discussed above; in general, in a spatial logic, they are derivable from the capability formulas. (In the paper we focus on the capability formulas, since they are more important in a spatial logic.)

For lack of space we do not present all the details of the proofs, and illustrate them only in the case of the π-calculus (detailed justifications can be found in [8]). When characterising a capability construct, we exploit operator $\triangleright$ to build a scenario that allows the capability to show its effect. This approach works rather smoothly under the strong interpretation of $\Diamond$, the constructions becoming more involved under a weak interpretation. In the latter case, we rely on some non-trivial properties, specific to each calculus we consider, to isolate some kind of composite constituents of interactions, that we call *threads*. In π, threads are sequences of input and output prefixes, while in Ambients they are sequences of nesting of open prefixes and ambients. The use of operators ® and И is crucial to derive formulas characterising threads.

Paper outline. In Section 2, we introduce our core spatial logic. We then present our results on the π-calculus (Section 3), and explain how we derive them. We also mention, in less detail, our results on the asynchronous π-calculus and on Mobile Ambients in Section 4. Section 5 gives some concluding remarks.

2 A Minimal Spatial Logic

A spatial logic is defined on *spatial calculi*, that is, calculi of processes that have the familiar constructs of parallel composition, restriction, and $\mathbf{0}$. These calculi are equipped with the usual relations: *structural congruence*, $\equiv$ (with the usual axioms for parallel composition, restriction, and $\mathbf{0}$), the *one-step reduction* relation $\longrightarrow$, and the *multistep reduction* relation $\longrightarrow^\star$ (the reflexive and transitive closure of $\longrightarrow$). Restriction is a binder, therefore notions of free and bound names of processes are also defined (for each process, the sets of free and bound names are finite). $\mathrm{fn}(P)$ stands for the set of free names of P. For two sets S_1, S_2 of names, $S_1 \setminus S_2$ stands for the set of names that belong to S_1 and not to S_2.

We give some properties, definitions, and notations for spatial calculi. We use $P, Q, \ldots$ to range over the *processes*, and $a, b, n, m, \ldots$ to range over the infinite set $\mathcal{N}$ of names. A process P

- has *normalised restrictions* if, for any occurrence of a subterm $(\boldsymbol{\nu} n)\, P'$ in P, name n occurs free in P' (that is, P has no useless restriction);

– has a *toplevel normalised restriction* if $P \equiv (\boldsymbol{\nu} n)\, P'$ and $n \in \mathrm{fn}(P')$.
– is *non-trivial* if it is not structurally congruent to $\mathbf{0}$;
– is *tight* if, up to structural congruence, it only has one component; in other words, it is non-trivial and is not the composition of two non-trivial processes. For example, in the π-calculus, $a(n).\, \overline{b}\langle n\rangle$ and $(\boldsymbol{\nu} a)\, (\overline{a}\langle b\rangle \mid a(n).\, \overline{c}\langle n\rangle)$ are tight, while $\overline{c}\langle a\rangle \mid \overline{b}\langle d\rangle$ is not.

As a consequence of the axioms of structural congruence for restriction, $n \in \mathrm{fn}(P)$ holds iff $P \not\equiv (\boldsymbol{\nu} n)\, P'$ for all P'. A (possibly empty) sequence of restrictions will be written $(\boldsymbol{\nu} \widetilde{n})\, P$.

Definition 1 (Core spatial logic $\mathcal{L}$). *Formulas of the spatial logic $\mathcal{L}$, ranged over by $\mathcal{A}, \mathcal{B}$, are defined by the following grammar:*

$$\mathcal{A} \quad ::= \; \mathcal{A}_1 \wedge \mathcal{A}_1 \;\big|\; \neg\mathcal{A} \;\big|\; 0 \;\big|\; \mathcal{A}_1 \mid \mathcal{A}_2 \;\big|\; \mathcal{A}_1 \triangleright \mathcal{A}_2 \;\big|\; n \circledR \mathcal{A} \;\big|\; \text{И} n.\, \mathcal{A} \;\big|\; \Diamond\mathcal{A}\,.$$

The set of free names of a formula $\mathcal{A}$ (written $\mathrm{fn}(\mathcal{A})$) is defined by saying that the only binding operator is И. We write $\mathcal{A}(n \leftrightarrow m)$ for the permutation of names n and m in formula $\mathcal{A}$. Given a spatial calculus $\mathcal{C}$, the temporal construct of the logic can be interpreted both *strongly*, that is, using the one-step reduction relation of the calculus, or *weakly*, that is, using multistep reduction. We write $P \models^{\mathtt{s}}_{\mathcal{C}} \mathcal{A}$ for the strong interpretation of $\mathcal{A}$, and $P \models^{\mathtt{w}}_{\mathcal{C}} \mathcal{A}$ for the weak interpretation. We use sw as a variable that ranges over $\{\mathtt{s}, \mathtt{w}\}$.

Definition 2 (Satisfaction in the spatial logic). *On a spatial calculus $\mathcal{C}$, satisfaction is defined by induction over the formulas as follows:*

– $P \models^{sw}_{\mathcal{C}} \mathcal{A}_1 \wedge \mathcal{A}_2$ *if* $P \models^{sw}_{\mathcal{C}} \mathcal{A}_1$ *and* $P \models^{sw}_{\mathcal{C}} \mathcal{A}_2$*;*
– $P \models^{sw}_{\mathcal{C}} \neg\mathcal{A}$ *if it is not the case that* $P \models^{sw}_{\mathcal{C}} \mathcal{A}$*;*
– $P \models^{sw}_{\mathcal{C}} 0$ *if* $P \equiv \mathbf{0}$*;*
– $P \models^{sw}_{\mathcal{C}} \mathcal{A}_1 \mid \mathcal{A}_2$ *if there are* P_1, P_2 *s.t.* $P \equiv P_1 \mid P_2$ *and* $P_i \models^{sw}_{\mathcal{C}} \mathcal{A}_i$ *for* $i = 1, 2$*;*
– $P \models^{sw}_{\mathcal{C}} \mathcal{A}_1 \triangleright \mathcal{A}_2$ *if for all* Q *s.t.* $Q \models^{sw}_{\mathcal{C}} \mathcal{A}_1$, $P \mid Q \models^{sw}_{\mathcal{C}} \mathcal{A}_2$*;*
– $P \models^{sw}_{\mathcal{C}} n \circledR \mathcal{A}$ *if there is* P' *such that* $P \equiv (\boldsymbol{\nu} n)P'$ *and* $P' \models^{sw}_{\mathcal{C}} \mathcal{A}$*;*
– $P \models^{sw}_{\mathcal{C}}$ И$n.\, \mathcal{A}$ *if for any* $n' \in \mathcal{N} \setminus (\mathrm{fn}(P) \cup \mathrm{fn}(\mathcal{A}))$, $P \models^{sw}_{\mathcal{C}} \mathcal{A}(n \leftrightarrow n')$*;*
– $P \models^{\mathtt{s}}_{\mathcal{C}} \Diamond\mathcal{A}$ *if there is* P' *such that* $P \longrightarrow P'$ *and* $P' \models^{\mathtt{s}}_{\mathcal{C}} \mathcal{A}$ *(we write this* $P \longrightarrow P' \models^{\mathtt{s}}_{\mathcal{C}} \mathcal{A}$*);*
– $P \models^{\mathtt{w}}_{\mathcal{C}} \Diamond\mathcal{A}$ *if there is* P' *such that* $P \longrightarrow^* P'$ *and* $P' \models^{\mathtt{w}}_{\mathcal{C}} \mathcal{A}$ *(we write this* $P \longrightarrow^* P' \models^{\mathtt{w}}_{\mathcal{C}} \mathcal{A}$*).*

Figure 1 presents some known formulas. $\vee$ and $\top$ are 'or' and 'true' operators; $\Box$ is the 'always' operator, the dual of $\Diamond$; and $\blacktriangleright$ is the dual of $\triangleright$ (thus $P \models^{sw}_{\mathcal{C}} \mathcal{A}_1 \blacktriangleright \mathcal{A}_2$ if there is Q such that $Q \models^{sw}_{\mathcal{C}} \mathcal{A}_1$ and $P \mid Q \models^{sw}_{\mathcal{C}} \mathcal{A}_2$). The models of formula **1** are tight terms. **2** is satisfied by the parallel composition of two processes satisfying **1**. Formula $\mathbf{Free}(a)$ says that a occurs free. We sometimes write $\mathbf{Free}(\widetilde{a}, \neg\widetilde{b})$ to abbreviate

$$\bigwedge \mathbf{Free}(a) \;\wedge\; \bigwedge \neg\,\mathbf{Free}(b) \qquad \text{for } a \in \widetilde{a} \text{ and } b \in \widetilde{b},$$

$$\mathcal{A}_1 \vee \mathcal{A}_2 \stackrel{\text{def}}{=} \neg(\neg\mathcal{A}_1 \wedge \neg\mathcal{A}_2) \qquad \Box\mathcal{A} \stackrel{\text{def}}{=} \neg\Diamond\neg\mathcal{A} \qquad \mathcal{A}_1 \blacktriangleright \mathcal{A}_2 \stackrel{\text{def}}{=} \neg(\mathcal{A}_1 \triangleright \neg\mathcal{A}_2)$$

$$\top \stackrel{\text{def}}{=} 0 \vee \neg 0 \qquad \mathbf{1} \stackrel{\text{def}}{=} \neg 0 \;\wedge\; \neg\,(\neg 0 \mid \neg 0) \qquad \mathbf{2} \stackrel{\text{def}}{=} (\neg 0 \mid \neg 0) \;\wedge\; \neg(\neg 0 \mid \neg 0 \mid \neg 0)$$

$$\mathbf{Free}(a) \stackrel{\text{def}}{=} \neg\, a\circledR\top \qquad \mathbf{public} \stackrel{\text{def}}{=} \neg И a.\, a\circledR(\neg a\circledR\top) \qquad \mathbf{single} \stackrel{\text{def}}{=} \mathbf{1} \;\wedge\; \mathbf{public}$$

Fig. 1. Some spatial formulas

that is, names $\widetilde{a}$ are free, and $\widetilde{b}$ are not. The models of **public** are the processes that are not structurally congruent to a process of the form $(\boldsymbol{\nu} n)\ P$ with $n \in \mathrm{fn}(P)$, in other words processes having no toplevel normalised restriction. We call such terms *public* processes. Finally, the models of **single** are those processes that are tight and do not exhibit a normalised restriction at toplevel. We call these the *single* processes. In the π-calculus, they are prefixed terms (Lemma 2).

3 The Logic in the π-calculus

3.1 The Process Calculus

Definition 3. *This is the grammar for the processes of the π-calculus:*

$$P \quad ::= \quad \mathbf{0} \;\big|\; P_1 \mid P_2 \;\big|\; (\boldsymbol{\nu} n)\, P \;\big|\; !P \;\big|\; \alpha.\, P, \qquad\qquad \alpha \quad ::= \quad a(b) \;\big|\; \overline{a}\langle b\rangle\,.$$

Here and in the remainder of the paper, we sometimes call this the *synchronous π-calculus*, to distinguish it from the asynchronous π-calculus studied below.

The *subject* of a prefix $\overline{m}\langle a\rangle.\, P$ or $m(a).\, P$ is m. We omit the trailing $\mathbf{0}$ in $\alpha.\,\mathbf{0}$. The set of free names of a process P is defined by saying that restriction and input are binding operators. We write $P_{\{b:=a\}}$ for the capture-avoiding substitution of name b with name a in P. Figure 2 presents the structural congruence and reduction relations for the π-calculus.

Definition 4. *A* thread *is a process given by the following grammar:*

$$Thr \quad ::= \quad \alpha.\, Thr \;\mid\; \alpha.\,\mathbf{0} \qquad (\alpha \textit{ is as by Definition 3}).$$

If $\tilde{\alpha}$ is a (possibly empty) sequence of prefixed actions, such as $\alpha_1.\,\alpha_2.\,\ldots.\,\alpha_n$, and P is a thread $\tilde{\alpha}.\,P'$, then P can perform actions $\tilde{\alpha}$ and then become P'. We indicate this using the notation $P \stackrel{\tilde{\alpha}}{\longrightarrow} P'$. We define a dualisation operation over prefixes by setting $\overline{a(b)} \stackrel{\text{def}}{=} \overline{a}\langle b\rangle$ and $\overline{\overline{a}\langle b\rangle} \stackrel{\text{def}}{=} a(b)$. This induces a similar operation $\overline{P}$ on processes.

Lemma 1 (Properties of π-calculus reductions).

1. *Let P, Q be single and s.t. $P \mid Q \longrightarrow \mathbf{0}$; then there are names a, b s.t. either $P \equiv \overline{a}\langle b\rangle.\,\mathbf{0}$ and $Q \equiv a(b).\,\mathbf{0}$, or vice versa ($Q \equiv \overline{a}\langle b\rangle.\,\mathbf{0}$ and $P \equiv a(b).\,\mathbf{0}$).*
2. *For any thread P we have $P \mid \overline{P} \longrightarrow^* \mathbf{0}$.*

We write $\models_\pi^{sw}$ for the satisfaction relations in the synchronous π-calculus (we recall that sw ranges over $\{\mathtt{s}, \mathtt{w}\}$).

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$
$$!P \equiv\, !P \mid P \qquad !(P \mid Q) \equiv\, !P \mid !Q \qquad !!P \equiv\, !P \qquad !\mathbf{0} \equiv \mathbf{0}$$
$$(\boldsymbol{\nu} n)\mathbf{0} \equiv \mathbf{0} \qquad (\boldsymbol{\nu} n)(\boldsymbol{\nu} m)P \equiv (\boldsymbol{\nu} m)(\boldsymbol{\nu} n)P \qquad (\boldsymbol{\nu} n)(P \mid Q) \equiv P \mid (\boldsymbol{\nu} n)Q \;\text{ if } n \notin \mathrm{fn}(P)$$

$$a(b).\,P \mid \overline{a}\langle c\rangle.\,Q \longrightarrow P_{\{b:=c\}} \mid Q$$
$$\frac{P \equiv P' \qquad P \longrightarrow Q \qquad Q \equiv Q'}{P' \longrightarrow Q'} \qquad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{(\boldsymbol{\nu} n)\,P \longrightarrow (\boldsymbol{\nu} n)\,P'}$$

Fig. 2. π-calculus: structural congruence and reduction

3.2 Main Results

We show the derivability of the logical formulas for the capabilities of the π-calculus (the input and output prefixes), as expressed by the following theorems:

Theorem 1 (Capabilities, strong case). *For any $\mathcal{A}, n, m$, there exist formulas $\mathbf{in}^{\mathtt{s}}(m,n).\,\mathcal{A}$ (with $n \neq m$) and $\mathbf{out}^{\mathtt{s}}(m,n).\,\mathcal{A}$ such that, for any P:*

- *$P \models^{\mathtt{s}}_{\pi} \mathbf{in}^{\mathtt{s}}(m,n).\,\mathcal{A}$ iff there are $P', n',\ n' \notin \mathrm{fn}(\mathcal{A})$. s.t. $P \equiv m(n').\,P'$ and $P' \models^{\mathtt{s}}_{\pi} \mathcal{A}(n \leftrightarrow n')$;*
- *$P \models^{\mathtt{s}}_{\pi} \mathbf{out}^{\mathtt{s}}(m,n).\,\mathcal{A}$ iff there is P' s.t. $P \equiv \overline{m}\langle n\rangle.\,P'$ and $P' \models^{\mathtt{s}}_{\pi} \mathcal{A}$.*

Theorem 2 (Capabilities, weak case). *For any $\mathcal{A}, n, m$, there exist formulas $\mathbf{in}^{\mathtt{w}}(m,n).\,\mathcal{A}$ (for $n \neq m$) and $\mathbf{out}^{\mathtt{w}}(m,n).\,\mathcal{A}$ such that, for any P:*

- *$P \models^{\mathtt{w}}_{\pi} \mathbf{in}^{\mathtt{w}}(m,n).\,\mathcal{A}$ iff there are $P', P'', n', n' \notin \mathrm{fn}(\mathcal{A})$, s.t. $P \equiv m(n').\,P'$ and $P' \longrightarrow^{*} P'' \models^{\mathtt{w}}_{\pi} \mathcal{A}(n \leftrightarrow n')$;*
- *$P \models^{\mathtt{w}}_{\pi} \mathbf{out}^{\mathtt{w}}(m,n).\,\mathcal{A}$ iff there are P', P'' s.t. $P \equiv \overline{m}\langle n\rangle.\,P'$ and $P' \longrightarrow^{*} P'' \models^{\mathtt{w}}_{\pi} \mathcal{A}$.*

These formulas easily allow us to define, in the strong and weak cases:

- the characteristic formulas for finite terms w.r.t. logical equivalence;
- the modality formulas for the input and output actions. For instance, in the weak case, the formula for the output modality $\langle\overline{a}\langle b\rangle\rangle.\,\mathcal{A}$ is satisfied by any process that is liable to perform some reduction steps, emit name b along a, and then perform some other reductions to reach a state where $\mathcal{A}$ is satisfied.

We do not present these constructions in detail, because either they are variations on existing work [7], or they are simple on their own.

3.3 Proofs

We sketch the proofs of Theorems 1 and 2. We consider the strong case first, since it is (much) simpler. The following formula will be useful: it shows that in the π-calculus the single processes are precisely the prefixed terms. The crux of the proof of the theorems, however, especially in the weak case, will be the definition of formulas to distinguish among different prefixes (whether the prefix is an input or an output, which channels it uses, what is its continuation, etc.).

Lemma 2. *For any P, $P \models_\pi^{sw}$ **single** iff $P \equiv m(n).\, P'$ or $P \equiv \overline{m}\langle n\rangle.\, P'$ for some m, n, P'.*

The strong case. In the strong case, the key formula is the following one.

$$\mathbf{test}(m,n) \stackrel{\text{def}}{=} \mathbf{Free}(m,n) \ \wedge \ (\mathbf{single} \blacktriangleright \Diamond 0)$$

Proposition 1. *For any P, n, m such that $n \neq m$, $P \models_\pi^{\mathsf{s}} \mathbf{test}(m,n)$ iff $P \equiv \overline{m}\langle n\rangle$ or $P \equiv \overline{n}\langle m\rangle$.*

Proof. We focus on the direct implication, the reverse direction being trivial. By Lemma 1, P is either of the form $\overline{a}\langle b\rangle.\, \mathbf{0}$ or $a(x).\, \mathbf{0}$. Having two distinct free names, P must be an output particle.

We can now define, using $\mathbf{test}(m,n)$, the formulas of Theorem 1:

$$\mathbf{in}^{\mathsf{s}}(m,n).\, \mathcal{A} \stackrel{\text{def}}{=} \mathbf{single} \ \wedge \ \text{И}n.\, \big(\mathbf{test}(m,n) \blacktriangleright \Diamond\ \mathcal{A}\big)$$
$$\mathbf{out}^{\mathsf{s}}(m,n).\, \mathcal{A} \stackrel{\text{def}}{=} \mathbf{single} \ \wedge \ \text{И}m'.\ \big(\mathbf{in}^{\mathsf{s}}(m,a).\, \mathbf{test}(m',a) \ \triangleright \ \Diamond(\mathbf{test}(m',n) \mid \mathcal{A})\big)$$

The formula for $\mathbf{in}^{\mathsf{s}}(m,n)$ requires a process to be single, and moreover the prefix should disappear in one step when in parallel with a certain tester $\mathbf{test}(m,n)$. The formula for $\mathbf{out}^{\mathsf{s}}(m,n).\, \mathcal{A}$ is similar, exploiting the previous formula $\mathbf{in}^{\mathsf{s}}(m,n).\, \mathcal{A}$; a test $\mathbf{test}(m',a)$ is required to observe the emitted name; this name instantiates a and is different from m' by construction.

The weak case. We first introduce formulas to isolate threads. This is achieved using a 'testing scenario', where the candidate process for being a thread is tested by putting in parallel a tester process. The latter should consume the tested part in a reduction sequence along which no more than two single components are observed. A subtle point is the ability, along the experiment, to distinguish the tested from the tester; we use the following property of π-calculus reductions:

Lemma 3. *Suppose that the following conditions hold: P, Q, R_1, R_2 are single processes such that $P \mid Q \longrightarrow R_1 \mid R_2$, and there exist two distinct names n and m s.t. $\{m,n\} \subseteq \mathrm{fn}(P) \setminus \mathrm{fn}(Q)$, $\{m,n\} \subseteq \mathrm{fn}(R_1) \setminus \mathrm{fn}(R_2)$. Then there exists a prefix α s.t. $P \equiv \alpha.\, R_1$ and $Q \equiv \overline{\alpha}.\, R_2$.*

In our case, the tester process will be identified using two distinct names (n, m below), that do not appear in the tested process, and that act as markers.

$$\begin{aligned}
\mathbf{tested}(m,n) &\stackrel{\text{def}}{=} \mathbf{single} \wedge \mathbf{Free}(\neg\, m, \neg\, n)\\
\mathbf{tester}(m,n) &\stackrel{\text{def}}{=} \mathbf{single} \wedge \mathbf{Free}(m,n)\\
\mathbf{dial}(m,n,\mathcal{A}) &\stackrel{\text{def}}{=} \Diamond(\mathbf{tester}(m,n) \mid \mathcal{A}) \ \wedge\ \Box\big(\mathbf{tester}(m,n) \mid (\mathbf{tested}(m,n) \vee \mathcal{A})\big)
\end{aligned}$$

The formula **dial** (for *dialog*) is supposed to be satisfied by the composition of the tester and the tested processes. Intuitively, **dial** requires that the computation leads to a state where the tested process 'has disappeared', and that at any moment along the computation either the tested process is still present, or formula $\mathcal{A}$ is satisfied. (This actually does not prevent the tester process from 'playing the role of the tested', once the tested has been consumed, but this will be of no harm for our purposes.)

For $m, n, \mathcal{A}$ fixed ($m \neq n$), we say that (P, Q) is *a pair tested/tester for* $\mathcal{A}$, and write this $P \ltimes_{\mathcal{A}} Q$, if $P \models^{\mathtt{w}}_{\pi} \mathbf{tested}(m,n) \vee \mathcal{A}$, $Q \models^{\mathtt{w}}_{\pi} \mathbf{tester}(m,n)$, and $P \mid Q \models^{\mathtt{w}}_{\pi} \mathbf{dial}(m,n,\mathcal{A})$. The following technical lemma, whose proof is based on Lemma 3, describes the execution scenario: as long as $\mathcal{A}$ is not satisfied, the tested process cannot contribute to the 'tester' component; that is, it cannot fork into two components one of which is used for the satisfaction of a 'tester' subformula of **dial**. We write $(P, Q) \longrightarrow (P', Q')$ if either $P \equiv a(x).\, P_1$, $Q \equiv \overline{a}\langle b\rangle.\, Q'$, $P \mid Q \longrightarrow P' \mid Q'$, and $P' \equiv P_1\{b/x\}$, or the symmetric configuration.

Lemma 4. *Assume $P \ltimes_{\mathcal{A}} Q$ for some m, n, A. Then:*

1. *either $(P, Q) \longrightarrow (P_1, Q_1 \mid Q_2)$ for some P_1, Q_1, Q_2, and $P_1 \mid Q_1 \ltimes_{\mathcal{A}} Q_2$,*
2. *or $P \models^{\mathtt{w}}_{\pi} \mathcal{A}$.*

In the above scenario, taking $\mathcal{A} = 0$ amounts to say that the tested process can disappear. We use this fact to define characteristic formulas for the threads.

Lemma 5. *Take $\mathcal{A} = 0$ in Lemma 4. For any single P s.t. $\{m, n\} \cap \mathrm{fn}(P) = \emptyset$, there is Q s.t. $P \ltimes_0 Q$ iff P is a thread.*

Proof. The proof relies on the fact that the scenario prevents the tested process from forking into two subcomponents.

We now define the formula that captures threads. We also need two auxiliary formulas to express existential and universal properties of suffixes of threads.

$$\begin{aligned}
\mathbf{Thread} &\stackrel{\text{def}}{=} \mathbf{single} \ \wedge\ Ⅎ m,n.\ \mathbf{tester}(m,n) \blacktriangleright \mathbf{dial}(m,n,0)\\
\langle\langle_\rangle\rangle.\, \mathcal{A} &\stackrel{\text{def}}{=} \mathbf{Thread} \ \wedge Ⅎ m,n.\ \big(\mathbf{Thread} \ \wedge\ \mathbf{tester}(m,n)\big) \blacktriangleright \mathbf{dial}(m,n,\mathcal{A})\\
[[_]].\, \mathcal{A} &\stackrel{\text{def}}{=} \mathbf{Thread} \ \wedge\ \neg \langle\langle_\rangle\rangle.\, \neg\, \mathcal{A}
\end{aligned}$$

Lemma 6. *The formulas above have the following interpretation:*

– $P \models^{\mathtt{w}}_{\pi}$ **Thread** *iff* P *is a thread.*

– $P \models^{\mathtt{w}}_{\pi} \langle\langle\_\rangle\rangle.\,\mathcal{A}$ *iff* P *is a thread such that there are* P' *and some* $\tilde{\alpha}$ *with* $P \xrightarrow{\tilde{\alpha}} P'$ *and* $P' \models^{\mathtt{w}}_{\pi} \mathcal{A}$.

– $P \models^{\mathtt{w}}_{\pi} [[\_]].\,\mathcal{A}$ *iff* P *is a thread s.t. whenever* $P \xrightarrow{\tilde{\alpha}} P'$ *for some* $P', \tilde{\alpha}$, $P' \models^{\mathtt{w}}_{\pi} \mathcal{A}$.

Proof. The first property is a direct consequence of Lemma 5. The other two are proved by induction on the size of the thread process being considered.

A thread *ends with* α if α is the last prefix in the thread. A thread is *located at* m if the subject of all its prefixes is m. We now refine our analysis of threads, by introducing formulas that isolate located threads that end with a special prefix.

$$\begin{aligned}
\mathbf{Barb}(m) &\stackrel{\text{def}}{=} \mathbf{single} \wedge \big((\mathbf{single} \wedge \neg\mathbf{Free}(m)) \triangleright \Box\, \mathbf{2}\big)\,. \\
\mathbf{EndO}(m,n) &\stackrel{\text{def}}{=} [[_]].\ \big(0 \ \vee\ (\mathbf{Barb}(m) \wedge \mathbf{Free}(n))\big) \\
\mathbf{EndI}(m) &\stackrel{\text{def}}{=} \text{И}n.\ [[_]].\ \big(0 \ \vee\ (\mathbf{EndO}(m,n) \blacktriangleright \Diamond 0)\big) \\
\mathbf{OutOnly}(m,n) &\stackrel{\text{def}}{=} \mathbf{EndO}(m,n) \ \wedge\ \text{И}n'.\ [[_]].\ \big(\mathbf{EndO}(m,n') \triangleright \Box \neg\, \mathbf{EndO}(m,n)\big)
\end{aligned}$$

Formula $\mathbf{Barb}(m)$ captures single terms whose initial prefix has subject m. This is obtained by requiring that such processes cannot interact with single processes that do not know m. Formula $\mathbf{EndO}(m,n)$ is satisfied by located threads that end with the particle $\overline{m}\langle n\rangle$. With a 'dualisation argument' we define $\mathbf{EndI}(n)$ in terms of $\mathbf{EndO}(m,n)$. Finally, formula $\mathbf{OutOnly}(m,n)$ captures the processes that satisfy $\mathbf{EndO}(m,n)$ and that have no input prefix. For this, we require that such processes are not able to 'consume' a thread having at least one output (cf. the $\mathbf{EndO}(m,n')$ subformula).

Lemma 7. *The formulas above have the following interpretation:*

- $P \models^{\mathtt{w}}_{\pi} \mathbf{Barb}(m)$ *iff* $P \equiv m(n).\,P'$ *or* $P \equiv \overline{m}\langle n\rangle.\,P'$ *for some* n *and* P'.
- *For* $n \neq m$, $P \models^{\mathtt{w}}_{\pi} \mathbf{EndO}(m,n)$ *iff* P *is a thread located at* m *ending with* $\overline{m}\langle n\rangle$ *with* n *not bound in* P.
- $P \models^{\mathtt{w}}_{\pi} \mathbf{EndI}(m)$ *iff* P *is a thread located at* m *and ending with* $m(x)$.
- *For* $n \neq m$, $P \models^{\mathtt{w}}_{\pi} \mathbf{OutOnly}(m,n)$ *iff* P *is of the form* $\overline{m}\langle c_1\rangle \ldots \overline{m}\langle c_r\rangle.\,\overline{m}\langle n\rangle$ *for some* $r \geq 0$ *and* $(c_i)_{1\leq i\leq r}$.

The last important step before defining the formula for Theorem 2 is the definition of some formulas that characterise certain 'particle' terms (Lemma 8). We use the same notation for these processes and the corresponding formulas:

$$\overline{m}\langle n\rangle \stackrel{\text{def}}{=} \mathbf{OutOnly}(m,n) \ \wedge\ \big(\mathbf{EndI}(m) \triangleright \Box \neg\, \mathbf{EndO}(m,n)\big)$$

$$
\begin{aligned}
m(n) &\stackrel{\text{def}}{=} \textbf{Thread} \quad\wedge\quad \text{И}n.\,\big(\overline{m}\langle n\rangle \rhd \Diamond 0\big)\\
m(n).\,\overline{p}\langle q\rangle &\stackrel{\text{def}}{=} \textbf{Thread} \quad\wedge\quad \text{И}n.\,\big(\overline{m}\langle n\rangle \rhd \Diamond\overline{p}\langle q\rangle\big)\\
\overline{m}\langle n\rangle.\,\overline{p}\langle q\rangle &\stackrel{\text{def}}{=} \textbf{Thread} \quad\wedge\quad \big(m(n) \rhd \Diamond\overline{p}\langle q\rangle\big)
\end{aligned}
$$

The definition of formula $\overline{m}\langle n\rangle$ imposes that a process satisfying formula $\mathbf{OutOnly}(m,n)$ has only one prefix: if this is not the case, then there exists a '$\mathbf{EndI}(m)$ process' that can be consumed, leading to a $\mathbf{EndO}(m,n)$ term.

Lemma 8. *Let $\mathcal{A}$ be one of the formulas above, where m,n,p,q are distinct names, and $Q_{\mathcal{A}}$ be the corresponding term. Then for any P, $P \models^{\mathtt{w}}_{\pi} \mathcal{A}$ iff $P \equiv Q_{\mathcal{A}}$.*

$$
\begin{aligned}
\mathbf{out}^{\mathtt{w}}(m,n).\,\mathcal{A} &\stackrel{\text{def}}{=} \text{И}p,q.\,\big(\; m(n).\,\overline{p}\langle q\rangle \rhd \Diamond(\overline{p}\langle q\rangle \mid \mathcal{A})\;\big)\\
\mathbf{in}^{\mathtt{w}}(m,n).\,\mathcal{A} &\stackrel{\text{def}}{=} \text{И}n,p,q.\,\big(\; \overline{m}\langle n\rangle.\,\overline{p}\langle q\rangle \rhd \Diamond(\overline{p}\langle q\rangle \mid \mathcal{A})\;\big)
\end{aligned}
$$

In both cases a flag process $\overline{p}\langle q\rangle$ is used to detect when the 'revealing' reduction step has occurred, since in the weak semantics the number of reductions is not observable.

4 The Logic in Other Calculi

The constructions we have shown above can be adapted to obtain similar results for two calculi that are quite different from the (synchronous) π-calculus: the asynchronous π-calculus and Mobile Ambients.

When moving to a different language, we cannot directly apply the formulas and the proofs examined on the π-calculus. The main reason is that the syntax changes, which affects our constructions, for instance when a formula $\mathcal{A}_1 \rhd \mathcal{A}_2$ is used (operator $\rhd$ talks about contexts of the calculus). Indeed, there are for example processes that cannot be distinguished in the logic (i.e., they satisfy the same sets of formulas of $\mathcal{L}$) when they are taken as processes of Aπ, but that can be distinguished when they are taken as processes of the π-calculus. This is the case for instance for processes $a(x).\,(\overline{a}\langle x\rangle \mid a(x).\,P)$ and $a(x).\,P$.

In this section, we present our results on Aπ and on Mobile Ambients. For lack of space, we only briefly hint on some aspects of the derivation of the formulas. Detailed constructions and proofs are available in [8].

4.1 Results in the Asychronous π-calculus

In Aπ there is no continuation underneath the output prefix:

$$
P \quad ::= \quad \mathbf{0} \;\big|\; P_1 \mid P_2 \;\big|\; (\boldsymbol{\nu} n)\,P \;\big|\; !P \;\big|\; n(m).\,P \;\big|\; \overline{n}\langle m\rangle\,.
$$

We omit the resulting modifications to the operational semantics. We write $\models^{sw}_{A\pi}$ for the satisfaction relations in Aπ. Below are the main results for Aπ, showing the derivability of the capability formulas for the strong and the weak semantics.

Theorem 3 (Capabilities, strong case). *For any $\mathcal{A}, n, m$, there exist formulas* $\mathbf{in}^{\mathtt{s}}(m,n).\,\mathcal{A}$ *(with the additional requirement $n \neq m$) and* $\mathbf{out}^{\mathtt{s}}(m,n)$ *such that, for any P:*

– $P \models^{\mathtt{s}}_{A\pi} \mathbf{in}^{\mathtt{s}}(m,n).\,\mathcal{A}$ *iff there are P', n', $n' \notin \mathrm{fn}(\mathcal{A})$. s.t. $P \equiv m(n').\,P'$ and $P' \models^{\mathtt{s}}_{A\pi} \mathcal{A}(n \leftrightarrow n')$;*
– $P \models^{\mathtt{s}}_{A\pi} \mathbf{out}^{\mathtt{s}}(m,n)$ *iff $P \equiv \overline{m}\langle n \rangle$.*

Theorem 4 (Capabilities, weak case). *For any $\mathcal{A}, n, m$, there exist formulas* $\mathbf{in}^{\mathtt{w}}(m,n).\,\mathcal{A}$ *(for $n \neq m$) and* $\mathbf{out}(m,n)$ *such that, for any P:*

– $P \models^{\mathtt{w}}_{A\pi} \mathbf{in}^{\mathtt{w}}(m,n).\,\mathcal{A}$ *iff there are P', P'', n', $n' \notin \mathrm{fn}(\mathcal{A})$ s.t. $P \equiv m(x).\,P'$ and $P \mid \overline{m}\langle n' \rangle \longrightarrow^{*} P'' \models^{\mathtt{w}}_{A\pi} \mathcal{A}(n \leftrightarrow n')$;*
– $P \models^{\mathtt{w}}_{A\pi} \mathbf{out}^{\mathtt{w}}(m,n)$ *iff $P \equiv \overline{m}\langle n \rangle$.*

As a consequence of Theorem 3, the output capability can be removed from the logic in [2]. To derive the capability formulas in Aπ, we proceed in a way that is quite similar to what we did in the π-calculus. It turns out that asynchrony actually simplifies our proofs: certain constructions become simpler, and certain results sharper. For instance, formula **Thread** from Section 3 directly captures output particles (intuitively because output particles cannot play the role of the tester in the scenario explained above).

4.2 Results in Mobile Ambients

The calculus of Mobile Ambients [4] is a model where the basic computational mechanism is movement, rather than communication as in π. The calculus of [4] also includes communications, which for simplicity we have omitted here.

Definition 5. *The grammar of Mobile Ambients is the following:*

$$P \;::=\; \mathbf{0} \;\big|\; P_1 \mid P_2 \;\big|\; (\boldsymbol{\nu} n)\,P \;\big|\; !P \;\big|\; n[P] \;\big|\; \alpha.\,P\,, \qquad \alpha \;::=\; \mathsf{open}\,n \;\big|\; \mathsf{in}\,n \;\big|\; \mathsf{out}\,n\,.$$

The structural congruence rules for mobile ambients are the same as in π, plus the following rule to allow restrictions to cross ambient boundaries:

$$n[(\boldsymbol{\nu} m)\,P] \;\equiv\; (\boldsymbol{\nu} m)\,n[P] \quad \text{if } n \neq m\,. \tag{1}$$

Instead of π's communication rule, we have the following rules for ambients:

$$\begin{aligned} \mathsf{open}\,n.\,P \;\mid\; n[Q] &\longrightarrow P \;\mid\; Q \\ m[\mathsf{in}\,n.\,P \;\mid\; Q] \;\mid\; n[R] &\longrightarrow n[m[P \;\mid\; Q] \;\mid\; R] \\ n[m[\mathsf{out}\,n.\,P \;\mid\; Q] \;\mid\; R] &\longrightarrow n[R] \;\mid\; m[P \;\mid\; Q] \end{aligned}$$

We write $\models^{\mathtt{s}}_{\mathrm{MA}}$ for strong satisfaction in MA. We only have a partial characterisation for the capability formulas for Ambients:

Theorem 5 (Capabilities, strong case). *For any $\mathcal{A}, n$, there exist formulas* $\mathbf{amb}^{\mathtt{s}}(n,\mathcal{A})$ *and* $\mathbf{open}^{\mathtt{s}}(n).\,\mathcal{A}$ *such that, for any P:*

– $P \models^{\mathtt{s}}_{MA} \mathbf{amb}^{\mathtt{s}}(n,\mathcal{A})$ *iff there is P' public s.t. $P \equiv n[P']$ with $P' \models^{\mathtt{s}}_{MA} \mathcal{A}$;*
– $P \models^{\mathtt{s}}_{MA} \mathbf{open}^{\mathtt{s}}(n).\,\mathcal{A}$ *iff there is P' s.t. $P \equiv \mathsf{open}\,n.\,P'$ with $P' \models^{\mathtt{s}}_{MA} \mathcal{A}$.*

The limitation about P' being public when deriving the ambient capability is related to the ability for restrictions to cross ambients in the structural rule (1) above. We believe that, on the other hand, the capability formulas for in and out are not derivable in $\mathcal{L}$. Theorem 5 allows us to obtain characteristic formulas for the finite processes of the dialect of Ambients studied in [1], which does not have the in and out capabilities.

We can derive the missing capability formulas for $\mathsf{in}\, n$ and $\mathsf{out}\, n$, and remove the constraints on the ambient capability formula from Theorem 5, in a variant of $\mathcal{L}$ enriched with the operator of ambient adjunct, @, whose satisfaction rule is:

$$P \models^{sw}_{\mathrm{MA}} \mathcal{A}@n \qquad \text{if} \qquad n[P] \models^{sw}_{\mathrm{MA}} \mathcal{A}.$$

The logics for ambients in the literature [5, 6] include both an ambient concsutruct and an ambient adjunct as primitive operators. As a consequence of this result, the former operator can be removed, at least in the strong case. (In the weak case, Theorem 5 remains valid, but we do not know whether the construction that eliminates the ambient formula can be adapted).

It is worth pointing out that in $\mathcal{L}$ (that is, without the ambient adjunct) we can derive the formulas for the modalities corresponding to the capability formulas of Theorem 5, without constraints on processes being public. For instance, the 'ambient modality' is a formula $\langle \mathbf{amb}^{\mathsf{s}}(n) \rangle . \mathcal{A}$ such that $P \models^{\mathsf{s}}_{\mathrm{MA}} \langle \mathbf{amb}^{\mathsf{s}}(n) \rangle . \mathcal{A}$ if $P \equiv (\boldsymbol{\nu}\widetilde{m})\,(n[P] \mid Q)$ with $n \notin \widetilde{m}$ and $(\boldsymbol{\nu}\widetilde{m})\,(P \mid Q) \models^{\mathsf{s}}_{\mathrm{MA}} \mathcal{A}$ (the weak modality is similar).

For lack of space we do not present the formal statement and the proofs of these results (see [8]).

5 Conclusion

We have showed that with a minimal spatial logic, $\mathcal{L}$, that has no calculus-specific operators, we can derive the formulas for capabilities and modalities in the π-calculus, both for the strong and for the weak semantics. Remarkably, the logic $\mathcal{L}$ does not tell anything about the features of the underlying calculus, other than saying that processes can be put in parallel and names can be restricted. To test the robustness of our techniques we have also considered the calculi Aπ and Ambients. As Ambients show, sometimes not all the capability and modality operators are derivable from $\mathcal{L}$: the addition of some calculus-specific constructs may be needed for this. Still, our constructions may allow us to reduce the number of such operators in the grammar of the logic.

The derivability of capability formulas is also useful for the definition of characteristic formulas w.r.t. logical equivalence, along the lines of [7]. We have not considered communication in Mobile Ambients, but reasoning as in [11] would allow us to handle this extension. Similarly, the π-calculus syntax (in the synchronous case) sometimes includes an operator of choice; we believe that we could adapt our constructions to this extension.

We do not know whether our results can be useful in tools (for model checking, for instance); perhaps our constructions are too complex for this at present. However, since they allow us to reduce the number of operators, our results should be important in the study of metatheoretical properties of the logics.

Our work shows that a logic with spatial constructions and the parallel composition adjunct ($\triangleright$) can express modalities. Conversely, it would be interesting to see whether the adjunct is derivable when the capability or/and the modality formulas are primitive in the logic.

Acknowledgments. We thank support by european project FET- Global Computing 'PROFUNDIS' and by the A.S. CNRS *'Méthodes Formelles pour la Mobilité'*.

References

1. N. Busi and G. Zavattaro. On the expressiveness of Movement in Pure Mobile Ambients. *ENTCS*, 66(3), 2002.
2. L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part I). In *Proc. of TACS'01*, LNCS. Springer Verlag, 2001.
3. L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part II). In *Proc. of CONCUR'02*, volume 2421 of *LNCS*, pages 209–225. Springer Verlag, 2002.
4. L. Cardelli and A. Gordon. Mobile Ambients. In *Proc. of FOSSACS'98*, volume 1378 of *LNCS*, pages 140–155. Springer Verlag, 1998.
5. L. Cardelli and A. Gordon. Anytime, Anywhere, Modal Logics for Mobile Ambients. In *Proc. of POPL'00*, pages 365–377. ACM Press, 2000.
6. L. Cardelli and A. Gordon. Logical Properties of Name Restriction. In *Proc. of TLCA'01*, volume 2044 of *LNCS*. Springer Verlag, 2001.
7. D. Hirschkoff, E. Lozes, and D. Sangiorgi. Separability, Expressiveness and Decidability in the Ambient Logic. In *17th IEEE Symposium on Logic in Computer Science*, pages 423–432. IEEE Computer Society, 2002.
8. D. Hirschkoff, E. Lozes, and D. Sangiorgi. Minimality Results for the Spatial Logics. Technical report, LIP – ENS Lyon, 2003. in preparation – available from `http://www.ens-lyon.fr/~elozes`.
9. U. Nestmann and B. Pierce. Decoding choice encodings. In *Proc. CONCUR '96*, volume 1119 of *LNCS*. Springer Verlag, 1996.
10. B. C. Pierce and D. N. Turner. Pict: A programming language based on the pi-calculus. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 2000.
11. D. Sangiorgi. Extensionality and Intensionality of the Ambient Logic. In *Proc. of 28th POPL*, pages 4–17. ACM Press, 2001.