Anytime, Anywhere
Modal Logics for Mobile Ambients

Luca Cardelli, Andrew D. Gordon
Microsoft Research

Abstract

The Ambient Calculus is a process calculus where processes may reside within a hierarchy of locations and modify it. The purpose of the calculus is to study mobility, which is seen as the change of spatial configurations over time. In order to describe properties of mobile computations we devise a modal logic that can talk about space as well as time, and that has the Ambient Calculus as a model.

1 Introduction

In the course of our ongoing work on mobility [3,4,5,12], we have often struggled to express precisely certain properties of mobile computations. Informally, these are properties such as "the agent has gone away", "eventually the agent crosses the firewall", "every agent always carries a suitcase", "somewhere there is a virus", or "there is always at most one agent called n here". There are several conceivable ways of formalizing these assertions. It is possible to express some of them in terms of equations [12], but this is sometimes difficult or unnatural. It is easier to express some of them as properties of computational traces, but this is very low-level.

Modal logics (particularly, temporal logics) have emerged in many domains as a good compromise between expressiveness and abstraction. In addition, many modal logics support useful computational applications, such as model checking. In our context, it makes sense to talk about properties that hold at particular locations, and it becomes natural to consider *spatial modalities* for properties that hold at a certain location, at some location, or at every location.

Space

Interesting spatial structures can be represented conveniently as unordered edge-labeled trees, where edge labels correspond to names of sublocations, and subtrees correspond to sublocations. Such a representation of locations is shared by the Ambient Calculus [3], the Distributed Join Calculus [10], the Seal Calculus [20], and trivially by the many distributed process calculi with a flat location structure (e.g.: [2]).

The following edge-labeled tree represents two contiguous locations, a and b, such that b has no sublocations, and a has a sublocation called p. The diagram on the right gives a more intuitive but equivalent description of location contiguity and containment:

In the Ambient Calculus, contiguous locations (or processes) are represented by standard parallel composition ($P \mid Q$), and named locations are represented by ambients ($n[P]$) which name a location n with contents P. This fragment of the Ambient Calculus, together with a void process ($\mathbf{0}$) and simple syntactic equivalences, amounts to a textual representation of edge-labeled trees. The example above could be written as $a[p[\mathbf{0}]] \mid b[\mathbf{0}]$, assuming there are no active processes within the locations.

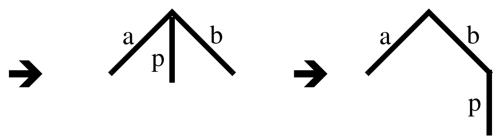
Even before considering process execution, we can talk about spatial properties and spatial specifications. For example, we have the following correspondence between spatial constructs in the Ambient Calculus and certain formulas of the logic we develop later:

Processes	
$\mathbf{0}$	(void)
$n[P]$	(location)
$P \mid Q$	(composition)
Formulas	
$\mathbf{0}$	(there is nothing here)
$n[\mathcal{A}]$	(there is one thing here)
$\mathcal{A} \mid \mathcal{B}$	(there are two things here)

We have a logical constant $\mathbf{0}$ that is satisfied by the process $\mathbf{0}$ representing void. We have logical propositions of the form $n[\mathcal{A}]$ (meaning that $\mathcal{A}$ holds at location n) that are satisfied by processes of the form $n[P]$ (meaning that process P is located at n) provided that P satisfies $\mathcal{A}$. We have logical propositions of the form $\mathcal{A} \mid \mathcal{B}$ (meaning that $\mathcal{A}$ and $\mathcal{B}$ hold contiguously) which are satisfied by contiguous processes of the form $P \mid Q$ if P satisfies $\mathcal{A}$ and Q satisfies $\mathcal{B}$, or vice versa.

Time

Spatial configurations evolve over time as a consequence of the activities of processes. For example, our initial tree may go through the following two steps of evolution, as the result of a process moving the location p from a to b through the ether in between.



POPL 2000, Boston, USA.

We can think of processes as sitting at the nodes of edge-labeled trees, and directing the movement of those nodes through the trees. So, the steps above could be caused by a process executing movement instructions at the node under p.

Mobility

We regard *mobility* as the evolution of spatial configurations over time. A specification logic for mobility should be able to talk about the structure of spatial configurations and about their evolution through time; that is, it should be a modal logic of space and time.

A typical specification would say that the configuration looks initially like a certain tree, and eventually like some other tree. In some cases we may want to be very precise about describing the structure of locations, even though this runs against the traditional attitude in logics for process calculi that prevents "counting" the number of processes (or locations) involved. Our logic can be very specific, in this sense.

Of course, since we are dealing with specifications, we may also want to be able to be imprecise, and describe things that happen "somewhere" or "sometime". Rarely, though, we want to be very precise about particular execution steps, so that the same flavor of logic of mobility seems applicable to a variety of calculi. In fact, the notion of mobility as evolution of location trees is shared by several calculi, including Ambients, Join, and Seal, although the mechanism and properties of mobility steps differ greatly between them.

In this paper, we concentrate on the Ambient Calculus for concreteness, but our main thrust is applicable to any distributed process calculus that includes a hierarchical and dynamic structure of locations.

Paper Outline

Spatial modalities have an intensional flavor that distinguishes our logic from other modal logics for concurrency. Previous work in the area concentrates on properties that are invariant up to strong equivalences such as bisimulation [15,6], while our properties are invariant only up to simple spatial rearrangements. Some of our techniques can be usefully applied to other process calculi, even ones that do not have locations, such as CCS.

We start from a computational basis: a process calculus, summarized in Section 2, that acts as a model for the logic. In Section 3 we introduce logical formulas and a notion of satisfaction. In Section 4, we derive logical inference rules, including rules for time, space, and satisfiability modalities, and novel rules for locations and process composition (the rules are summarized in the Appendix). At the end of this section we give a detailed example of logical inference. In Section 5 we investigate model checking of mobile programs, on the basis of the satisfaction relation between processes and formulas. Finally, in Section 6, we compare our logic with relevant and linear logics.

2 The Ambient Calculus with Public Names

In this paper we consider only ambients having public names; that is we do not deal with name restriction and scope extrusion. Handling of private names in a logic is a very interesting topic, but we leave it for future work.

2.1 Ambients

We summarize a modified version of the basic Ambient Calculus of [3]. The changes consist in removing name restriction, and in strengthening the definition of structural congruence so that it characterizes the intended equivalence on spatial configurations.

The following table summarizes the syntax of processes. We have separated the process constructs into *spatial* and *temporal*; this is similar to the distinction between static and dynamic constructs in CCS [17]. This paper focuses on the spatial constructs; the temporal constructs and the dynamic behavior are necessary but secondary for our current purposes.

Processes

$P,Q,R ::=$	processes	
$\mathbf{0}$	void	spatial
$P \mid Q$	composition	spatial
$!P$	replication	spatial
$M[P]$	ambient	spatial
$M.P$	capability action	temporal
$(n).P$	input action	temporal
$\langle M \rangle$	output action	temporal
$M ::=$	messages	
n	name	names
$in\ M$	can enter into M	capabilities
$out\ M$	can exit out of M	capabilities
$open\ M$	can open M	capabilities
ε	null	paths
$M.M'$	composite	paths

The set of free names of a process P, written $fn(P)$, is defined as usual; the only binder is in the input action. We write $P\{n \leftarrow M\}$ for the substitution of the message M for each free occurrence of the name n in the process P. Similarly for $M\{n \leftarrow M'\}$. The $\mathbf{0}$ process is often omitted in the contexts $n[\mathbf{0}]$ and $M.\mathbf{0}$, yielding $n[]$ and M.

2.2 Structural Congruence and Reduction

Structural congruence is a relation between processes; it is used heavily in the logic, as well as in the reduction semantics. Intuitively, structural congruence equates processes up to simple "rearrangement" of parts, without any computational significance. We can identify five groups of rules in the following table: for equivalence, for congruence of spatial operators, for composition, for replication, and for temporal operators and paths.

Structural Congruence

$P \equiv P$	(Struct Refl)
$P \equiv Q \;\Rightarrow\; Q \equiv P$	(Struct Symm)
$P \equiv Q,\ Q \equiv R \;\Rightarrow\; P \equiv R$	(Struct Trans)
$P \equiv Q \;\Rightarrow\; P \mid R \equiv Q \mid R$	(Struct Par)
$P \equiv Q \;\Rightarrow\; !P \equiv !Q$	(Struct Repl)
$P \equiv Q \;\Rightarrow\; M[P] \equiv M[Q]$	(Struct Amb)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$P \mid \mathbf{0} \equiv P$	(Struct Par Zero)
$!(P \mid Q) \equiv !P \mid !Q$	(Struct Repl Par)
$!\mathbf{0} \equiv \mathbf{0}$	(Struct Repl Zero)
$!P \equiv P \mid !P$	(Struct Repl Copy)
$!P \equiv !!P$	(Struct Repl Repl)

$P \equiv Q \;\Rightarrow\; M.P \equiv M.Q$	(Struct Action)
$P \equiv Q \;\Rightarrow\; (x).P \equiv (x).Q$	(Struct Input)
$\varepsilon.P \equiv P$	(Struct ε)
$(M.M').P \equiv M.M'.P$	(Struct .)

Spatial configurations are ambient configurations consisting only of spatial operators. For example, $a[b[\mathbf{0}] \mid !c[\mathbf{0} \mid \mathbf{0}] \mid !\mathbf{0}]$ is a spatial configuration. These configurations have a natural interpretation as edge-labeled finite-depth trees, where replication introduces infinite branching. The rules for structural congruence are sound and complete for equivalence of these trees. We do not elaborate this further, but it suffices to say that this completeness result motivates the choice of axioms for structural congruence, and particularly the axioms for replication (which are the same as in Engelfriet's work on the π-calculus [9]).

Reduction

$n[in\ m.\ P \mid Q] \mid m[R] \longrightarrow m[n[P \mid Q] \mid R]$	(Red In)
$m[n[out\ m.\ P \mid Q] \mid R] \longrightarrow n[P \mid Q] \mid m[R]$	(Red Out)
$open\ n.\ P \mid n[Q] \longrightarrow P \mid Q$	(Red Open)
$(n).P \mid \langle M \rangle \longrightarrow P\{n \leftarrow M\}$	(Red Comm)
$P \longrightarrow Q \;\Rightarrow\; n[P] \longrightarrow n[Q]$	(Red Amb)
$P \longrightarrow Q \;\Rightarrow\; P \mid R \longrightarrow Q \mid R$	(Red Par)
$P' \equiv P,\ P \longrightarrow Q,\ Q \equiv Q' \;\Rightarrow\; P' \longrightarrow Q'$	(Red ≡)
$\longrightarrow^*$ is the reflexive and transitive closure of $\longrightarrow$	

The reduction relation describes the dynamic behavior of ambients. In particular, the rules (Red In), (Red Out) and (Red Open) represent mobility, while (Red Comm) represents local communication (see [3] for an extended discussion). For example, the process:

$$a[p[out\ a.\ in\ b.\ \langle m \rangle]] \mid b[open\ p.\ (x).\ x[]]$$

represents a packet p that travels out of host a and into host b, where it is opened, and its contents m are read and used to create a new ambient. The process reduces in four steps (illustrating each of the four reduction rules) to the residual process $a[] \mid b[m[]]$. The first three states correspond to the tree diagrams in the Introduction.

$a[p[out\ a.\ in\ b.\ \langle m \rangle]] \mid b[open\ p.\ (x).\ x[]]$	
$\longrightarrow a[] \mid p[in\ b.\ \langle m \rangle] \mid b[open\ p.\ (x).\ x[]]$	(Red Out)
$\longrightarrow a[] \mid b[p[\langle m \rangle] \mid open\ p.\ (x).\ x[]]$	(Red In)
$\longrightarrow a[] \mid b[\langle m \rangle \mid (x).\ x[]]$	(Red Open)
$\longrightarrow a[] \mid b[m[]]$	(Red Comm)

2-1 Facts about Structural Congruence

(1) $P \mid Q \equiv \mathbf{0}$ iff $P \equiv \mathbf{0}$ and $Q \equiv \mathbf{0}$.

(2) $n[P] \not\equiv \mathbf{0}$.

(3) $n[P] \equiv Q \mid R$ iff either $Q \equiv n[P]$ and $R \equiv \mathbf{0}$, or $Q \equiv \mathbf{0}$ and $R \equiv n[P]$.

(4) $m[P] \equiv n[Q]$ iff $m = n$ and $P \equiv Q$.

(5) $m[P] \mid n[Q] \equiv m'[P'] \mid n'[Q']$ iff either $m = m'$, $n = n'$, $P \equiv P'$, $Q \equiv Q'$, or $m = n'$, $n = m'$, $P \equiv Q'$, $Q \equiv P'$.

□

3 The Logic

In a modal logic, the truth of a formula is relative to a state (or world). In our case, the truth of a *space-time* modal formula is relative to the *here and now*. Each formula talks about the current time, that is, the current state of execution, and the current place, that is, the current location. For example, the formula $n[\mathbf{0}]$ is read: *there is here and now an empty location called n*. The operator $n[\mathcal{A}]$ represents a single step in space, allowing us to talk about the place one step down into n. Another operator, $\diamond\!\!\!\!\cdot\,\mathcal{A}$, allows us to talk about an arbitrary number of steps in space; this is akin to the temporal eventuality operator, $\Diamond\mathcal{A}$.

3.1 Logical Formulas

The syntax of logical formulas is summarized below. This is a modal predicate logic with classical negation. As usual, many standard connectives are interdefinable. The meaning of the formulas will be given shortly in terms of a satisfaction relation. Informally, the first three formulas (true, negation, disjunction) give propositional logic. The next three (void, location, composition) capture spatial configurations, as we discussed. Then we have quantification over names, the two temporal and spatial modalities, and two further operators that we explain later. Quantified variables range only over names: these variables may appear in the location and location adjunct constructs.

Logical Formulas

η is a name n or a variable x

$\mathcal{A}, \mathcal{B}, \mathcal{C} ::=$

$\mathbf{T}$	true
$\neg\mathcal{A}$	negation
$\mathcal{A} \vee \mathcal{B}$	disjunction
$\mathbf{0}$	void
$\eta[\mathcal{A}]$	location
$\mathcal{A} \mid \mathcal{B}$	composition
$\forall x.\mathcal{A}$	universal quantification over names
$\Diamond\mathcal{A}$	sometime modality
$\diamond\!\!\!\!\cdot\,\mathcal{A}$	somewhere modality
$\mathcal{A}@\eta$	location adjunct
$\mathcal{A} \triangleright \mathcal{B}$	composition adjunct

The free names of a formula, $fn(\mathcal{A})$, are easily defined since there are no name binders. The free variables of a formula, $fv(\mathcal{A})$, are defined along standard lines: only quantifiers bind variables. A formula $\mathcal{A}$ is closed if $fv(\mathcal{A}) = \varnothing$.

3.2 Satisfaction

The satisfaction relation $P \vDash \mathcal{A}$ means that the process P satisfies the closed formula $\mathcal{A}$. This relation is defined inductively in the following table, where Π is the sort of processes, Φ is the sort of formulas, ϑ is the sort of variables, and Λ is the sort of names. We are very explicit about quantification and sorting of meta-variables because of subtle scoping issues, particularly in the definition of $P \vDash \forall x.\mathcal{A}$. We use the same syntax for logical connectives at the meta-level and object-level, but this is unambiguous.

The meaning of the temporal modality is given by reductions in the operational semantics of the Ambient Calculus. For the spatial modality, we need the following definition: the relation $P \downarrow P'$ indicates that P contains P' within exactly one level of nesting; that is, P' is one step away from P in space, in some downward direction.

$$P \downarrow P' \quad \text{iff} \quad \exists n, P''.\ P \equiv n[P'] \mid P''$$

Then, $P \downarrow^* P'$ is the reflexive and transitive closure of the previous relation, indicating that P contains P' at some nesting level. Note that P' consists of either the top level P, or the entire contents of an enclosed ambient.

Satisfaction

$\forall P{:}\Pi.$	$P \vDash \mathbf{T}$		
$\forall P{:}\Pi, \mathcal{A}{:}\Phi.$	$P \vDash \neg\mathcal{A}$	$\triangleq$	$\neg\, P \vDash \mathcal{A}$
$\forall P{:}\Pi, \mathcal{A},\mathcal{B}{:}\Phi.$	$P \vDash \mathcal{A}\vee\mathcal{B}$	$\triangleq$	$P \vDash \mathcal{A} \vee P \vDash \mathcal{B}$
$\forall P{:}\Pi.$	$P \vDash \mathbf{0}$	$\triangleq$	$P \equiv \mathbf{0}$
$\forall P{:}\Pi, n{:}\Lambda, \mathcal{A}{:}\Phi.$	$P \vDash n[\mathcal{A}]$	$\triangleq$	$\exists P'{:}\Pi.\ P \equiv n[P'] \wedge P' \vDash \mathcal{A}$
$\forall P{:}\Pi, \mathcal{A},\mathcal{B}{:}\Phi.$	$P \vDash \mathcal{A} \mid \mathcal{B}$	$\triangleq$	$\exists P',P''{:}\Pi.\ P \equiv P' \mid P'' \wedge P' \vDash \mathcal{A} \wedge P'' \vDash \mathcal{B}$
$\forall P{:}\Pi, x{:}\vartheta, \mathcal{A}{:}\Phi.$	$P \vDash \forall x.\mathcal{A}$	$\triangleq$	$\forall m{:}\Lambda.\ P \vDash \mathcal{A}\{x \leftarrow m\}$
$\forall P{:}\Pi, \mathcal{A}{:}\Phi.$	$P \vDash \Diamond\mathcal{A}$	$\triangleq$	$\exists P'{:}\Pi.\ P \rightarrow^* P' \wedge P' \vDash \mathcal{A}$
$\forall P{:}\Pi, \mathcal{A}{:}\Phi.$	$P \vDash \text{✧}\mathcal{A}$	$\triangleq$	$\exists P'{:}\Pi.\ P \downarrow^* P' \wedge P' \vDash \mathcal{A}$
$\forall P{:}\Pi, \mathcal{A}{:}\Phi.$	$P \vDash \mathcal{A}@n$	$\triangleq$	$n[P] \vDash \mathcal{A}$
$\forall P{:}\Pi, \mathcal{A},\mathcal{B}{:}\Phi.$	$P \vDash \mathcal{A}\triangleright\mathcal{B}$	$\triangleq$	$\forall P'{:}\Pi.\ P' \vDash \mathcal{A} \Rightarrow P \mid P' \vDash \mathcal{B}$

We spell out some of these definitions. A process P satisfies the formula $n[\mathcal{A}]$ if there exists a process P' such that P has the shape $n[P']$ with P' satisfying $\mathcal{A}$. A process P satisfies the formula $\mathcal{A}' \mid \mathcal{A}''$ if there exist processes P' and P'' such that P has the shape $P' \mid P''$ with P' satisfying $\mathcal{A}'$ and P'' satisfying $\mathcal{A}''$. A process P satisfies the formula $\Diamond\mathcal{A}$ if $\mathcal{A}$ holds in the future for some residual P' of P, where "residual" is defined by $P \rightarrow^* P'$. A process P satisfies the formula $\text{✧}\mathcal{A}$ if $\mathcal{A}$ holds at some sublocation P' within P, where "sublocation" is defined by $P \downarrow^* P'$.

The last two connectives, @ and $\triangleright$, can be used to express assumption/guarantee specifications [1]; they were inspired by the wish to express security properties. A reading of $P \vDash \mathcal{A}@n$ is that P (together with its context) manages to satisfy $\mathcal{A}$ even when placed into a location called n. A reading of $P \vDash \mathcal{A}\triangleright\mathcal{B}$ is that P (together with its context) manages to satisfy $\mathcal{B}$ under any possible attack by an opponent that is bound to satisfy $\mathcal{A}$. Moreover, $P \vDash (\Box\mathcal{A})\triangleright(\Box\mathcal{A})$ can be interpreted as saying that P preserves the invariant $\mathcal{A}$. We will see that these two connectives arise as natural adjuncts to the location and composition connectives, respectively.

The definition of satisfaction is based heavily on the structural congruence relation. This use of structural congruence may appear arbitrary: other equivalence relations could be used in its place. We have tried to motivate the choice of structural congruence by discussing in Section 2.2 how structural congruence precisely captures the intuition of ambients as spatial configurations. Moreover, structural congruence is easily decidable, which is useful in model-checking applications (see Section 5).

The following table lists some derived connectives, illustrating some properties that can be expressed in the logic. The informal meanings can be understood better by expanding out the definitions from the table above. Some discussion follows.

Derived Connectives

$\mathbf{F}$	$\triangleq \neg\mathbf{T}$	false
$\mathcal{A} \wedge \mathcal{B}$	$\triangleq \neg(\neg\mathcal{A} \vee \neg\mathcal{B})$	conjunction
$\mathcal{A} \Rightarrow \mathcal{B}$	$\triangleq \neg\mathcal{A} \vee \mathcal{B}$	implication
$\mathcal{A} \Leftrightarrow \mathcal{B}$	$\triangleq (\mathcal{A} \Rightarrow \mathcal{B}) \wedge (\mathcal{B} \Rightarrow \mathcal{A})$	logical equivalence
$\mathcal{A} \parallel \mathcal{B}$	$\triangleq \neg(\neg\mathcal{A} \mid \neg\mathcal{B})$	decomposition
$\mathcal{A}^\forall$	$\triangleq \mathcal{A} \parallel \mathbf{F}$	every component satisfies $\mathcal{A}$
$\mathcal{A}^\exists$	$\triangleq \mathcal{A} \mid \mathbf{T}$	some component satisfies $\mathcal{A}$
$\exists x.\mathcal{A}$	$\triangleq \neg\forall x.\neg\mathcal{A}$	existential quantification
$\Box\mathcal{A}$	$\triangleq \neg\Diamond\neg\mathcal{A}$	everytime modality
¤$\mathcal{A}$	$\triangleq \neg\text{✧}\neg\mathcal{A}$	everywhere modality
$\mathcal{A} \propto \mathcal{B}$	$\triangleq \neg(\mathcal{B} \triangleright \neg\mathcal{A})$	fusion
$\mathcal{A} \mid\!\Rightarrow \mathcal{B}$	$\triangleq \neg(\mathcal{A} \mid \neg\mathcal{B})$	fusion adjunct

Syntactic conventions: '$\triangleright$', '$\Diamond$', '$\Box$', '✧', and '¤' bind more strongly than '$\mid$'; and they all bind more strongly than the standard logical connectives, which have standard precedences. Quantifiers extend to the right as far as possible.

Decomposition is the DeMorgan dual of composition. A decomposition formula $\mathcal{A} \parallel \mathcal{B}$ is satisfied if for every parallel decomposition of the process in question, either one component satisfies $\mathcal{A}$ or the other satisfies $\mathcal{B}$. Then, $\mathcal{A}^\forall$ means that in every decomposition either one component satisfies $\mathcal{A}$ or the other satisfies $\mathbf{F}$; since the latter is impossible, in every possible decomposition one component must satisfy $\mathcal{A}$. For example: $(n[\mathbf{T}]\Rightarrow n[m[\mathbf{T}]])^\forall$ means that every ambient n that can be found here contains a single subambient m. The DeMorgan dual of $\mathcal{A}^\forall$ is $\mathcal{A}^\exists$, which means that it is possible to find a decomposition where one component satisfies $\mathcal{A}$. For example, $n[m[\mathbf{T}]^\exists]^\exists$ means that there is at least one ambient n here that contains at least one subambient m.

Other operators are derived as DeMorgan duals: existential quantification, and everytime and everywhere modalities. Examples for these modalities are: $\Box n[\mathbf{T}]$ (there is always a location called n here), and ¤$\neg(n[\mathbf{T}]^\exists)$ (there is now no location called n anywhere).

Fusion, $\mathcal{A} \propto \mathcal{B}$, is an operator that arises in relevant logic (when $\triangleright$ is seen as relevant implication). In our context, $\mathcal{A} \propto \mathcal{B}$ means that there is a context satisfying $\mathcal{B}$ that helps ensuring $\mathcal{A}$. The adjunct of fusion, $\mathcal{A} \mid\!\Rightarrow \mathcal{B}$, turns out to be very natural in specifications: it means that in every decomposition, if one part satisfies $\mathcal{A}$, then the other part must satisfy $\mathcal{B}$.

The following is a fundamental property of the satisfaction relation; it states that satisfaction is invariant under structural congruence of processes. In other words, logical formulas can only express properties that are invariant up to structural congruence. The proof is a simple induction on the structure of $\mathcal{A}$.

3-1 Proposition (Satisfaction is up to $\equiv$)

$(P \vDash \mathcal{A} \wedge P \equiv P') \Rightarrow P' \vDash \mathcal{A}$

$\Box$

We end this section with an example of a proof that a certain process satisfies a certain formula. A proof of even a very simple negative formula requires techniques for analyzing the derivation of structural congruences. For example, consider proving the following assertion, where $m \neq n$:

$$m[] \mid n[] \vDash \neg\, \exists x.\ x[\mathbf{T}] \mid x[\mathbf{T}]$$

For a contradiction, suppose that $m[] \mid n[] \vDash \exists x.\ x[\mathbf{T}] \mid x[\mathbf{T}]$. By definition, this means there is a P such that $m[] \mid n[] \equiv P$ and there is a q with $P \vDash q[\mathbf{T}] \mid q[\mathbf{T}]$. This implies that there are processes P' and P'' such that $m[] \mid n[] \equiv P' \mid P''$ with $P' \vDash q[\mathbf{T}]$ and $P'' \vDash q[\mathbf{T}]$. In turn, $P' \vDash q[\mathbf{T}]$ implies there is Q' such that $P' \equiv q[Q']$. Similarly, $P'' \vDash q[\mathbf{T}]$ implies there is Q'' such that $P'' \equiv q[Q'']$. In summary:

$$m[] \mid n[] \equiv q[Q'] \mid q[Q'']$$

According to the Fact 2-1(5), there are two ways in which this equation can have been derived. In either case, it follows that $m = q$ and $n = q$, and therefore $m = n$. This yields the desired contradiction, as we are assuming that $m \neq n$.

4 Validity

In this section, we study valid formulas, valid sequents, and valid logical inference rules. All these are based on the satisfaction relation given in the previous section. Once the definition of satisfaction is fixed, we are basically committed to whatever logic comes out of it. Therefore, it is important to stress that the satisfaction relation appears very natural to us. In particular, the definitions of $\mathbf{0}$, $n[\mathcal{A}]$, and $\mathcal{A} \mid \mathcal{B}$ seem inevitable, once we accept that formulas should be able to talk about the tree structure of locations, and that they should not distinguish processes that are surely indistinguishable (up to $\equiv$). The connectives $\mathcal{A}@n$ and $\mathcal{A}{\triangleright}\mathcal{B}$ have natural security motivations. The modalities $\Diamond\mathcal{A}$ and ✧$\mathcal{A}$ talk about process evolution and structure in an undetermined way, which is good for mobility specifications. The rest is classical predicate logic, with the ability to quantify over location names.

Through the satisfaction relation, our logic is based on solid computational intuitions. We should now approach the task of discovering the rules of the logic without preconceptions. As we shall see, what we get has familiar as well as novel aspects.

4.1 The Meaning of Rules

A closed formula is valid if it is satisfied by every process. (For the moment, we consider only validity for closed formulas, i.e., propositional validity.) We use validity for interpreting logical inference rules, as described in the next definition. We use a linearized notation for inference rules, where the usual horizontal bar separating antecedents from consequents is written '$\}$' in-line, and ';' is used to separate antecedents.

Validity, Sequents, and Rules

$\boldsymbol{vld}(\mathcal{A}) \triangleq \forall P{:}\Pi.\ P \vDash \mathcal{A}$	Validity for (closed) $\mathcal{A}$
$\mathcal{A} \vdash \mathcal{B} \triangleq \boldsymbol{vld}(\mathcal{A} \Rightarrow \mathcal{B})$	Sequent
$\mathcal{A} \dashv\vdash \mathcal{B} \triangleq \mathcal{A} \vdash \mathcal{B} \wedge \mathcal{B} \vdash \mathcal{A}$	Double Sequent
$\mathcal{A}_1 \vdash \mathcal{B}_1; \ldots; \mathcal{A}_n \vdash \mathcal{B}_n \} \mathcal{A}_0 \vdash \mathcal{B}_0 \triangleq$ $\mathcal{A}_1 \vdash \mathcal{B}_1 \wedge \ldots \wedge \mathcal{A}_n \vdash \mathcal{B}_n \Rightarrow \mathcal{A}_0 \vdash \mathcal{B}_0$	Inference Rule ($n \geq 0$)
$\mathcal{A}_1 \vdash \mathcal{B}_1; \ldots; \mathcal{A}_n \vdash \mathcal{B}_n \} \mathcal{A} \dashv\vdash \mathcal{B} \triangleq$ $\mathcal{A}_1 \vdash \mathcal{B}_1 \wedge \ldots \wedge \mathcal{A}_n \vdash \mathcal{B}_n \Rightarrow \mathcal{A}_0 \dashv\vdash \mathcal{B}_0$	Double Conclusion
$\mathcal{A}_1 \vdash \mathcal{B}_1 \{\} \mathcal{A}_2 \vdash \mathcal{B}_2 \triangleq$ $\mathcal{A}_1 \vdash \mathcal{B}_1 \} \mathcal{A}_2 \vdash \mathcal{B}_2 \wedge \mathcal{A}_2 \vdash \mathcal{B}_2 \} \mathcal{A}_1 \vdash \mathcal{B}_1$	Double Rule

We adopt a non-standard formulation of sequents, where each sequent has exactly one assumption and one conclusion: $\mathcal{A} \vdash \mathcal{B}$. Our intention in doing so is to avoid pre-judging the interpretation of the structural operator "," in standard sequents. In our logic, by taking $\wedge$ on the left and $\vee$ on the right of $\vdash$ as structural operators (i.e., as ","), all the standard rules of sequent and natural deduction systems with multiple premises/conclusions can be derived. Instead, by taking $\mid$ on the left of $\vdash$ as a structural operator, all the rules of intuitionistic linear logic can be derived. Finally, by taking nestings of $\wedge$ and $\mid$ on the left of $\vdash$ as structural "bunches", we obtain a bunched logic [18]. We discuss this further in Section 6.

Noticeably, we abandon Gentzen's distinction between structural rules and other logical rules, which has been a staple of formal logic since [11]. We do not see this as a fundamental or irrevocable step. Not all logics fit easily into Gentzen's initial approach, and many alternative sequent structures have been studied [7]. Therefore, there may be formulations of our logic which identify a set of structural rules, perhaps along the lines of [18]. At the current stage in the development of our logic, however, it is unclear how to proceed in that direction.

4.2 Rules of the Logic

In the sequel, we organize our results into tables of Rules, which are validated in the model, and into tables of Corollaries, which are derived purely logically from the inference rules.

4.2.1 Propositions

The following is a non-standard presentation of the propositional sequent calculus [14], based on our single-assumption single-conclusion sequents. In this presentation, the rules of propositional logic become very symmetrical, and many proofs become more regular, having to consider only single formulas instead of sequences of formulas.

Propositional Rules

(A-L)	$\mathcal{A}\wedge(\mathcal{C}\wedge\mathcal{D}) \vdash \mathcal{B} \{\} (\mathcal{A}\wedge\mathcal{C})\wedge\mathcal{D} \vdash \mathcal{B}$
(A-R)	$\mathcal{A} \vdash (\mathcal{C}\vee\mathcal{D})\vee\mathcal{B} \{\} \mathcal{A} \vdash \mathcal{C}\vee(\mathcal{D}\vee\mathcal{B})$
(X-L)	$\mathcal{A}\wedge\mathcal{C} \vdash \mathcal{B} \} \mathcal{C}\wedge\mathcal{A} \vdash \mathcal{B}$
(X-R)	$\mathcal{A} \vdash \mathcal{C}\vee\mathcal{B} \} \mathcal{A} \vdash \mathcal{B}\vee\mathcal{C}$
(C-L)	$\mathcal{A}\wedge\mathcal{A} \vdash \mathcal{B} \} \mathcal{A} \vdash \mathcal{B}$
(C-R)	$\mathcal{A} \vdash \mathcal{B}\vee\mathcal{B} \} \mathcal{A} \vdash \mathcal{B}$
(W-L)	$\mathcal{A} \vdash \mathcal{B} \} \mathcal{A}\wedge\mathcal{C} \vdash \mathcal{B}$
(W-R)	$\mathcal{A} \vdash \mathcal{B} \} \mathcal{A} \vdash \mathcal{C}\vee\mathcal{B}$
(Id)	$\} \mathcal{A} \vdash \mathcal{A}$
(Cut)	$\mathcal{A} \vdash \mathcal{C}\vee\mathcal{B};\ \mathcal{A}'\wedge\mathcal{C} \vdash \mathcal{B}' \} \mathcal{A}\wedge\mathcal{A}' \vdash \mathcal{B}\vee\mathcal{B}'$
(**T**)	$\mathcal{A}\wedge\mathbf{T} \vdash \mathcal{B} \} \mathcal{A} \vdash \mathcal{B}$
(**F**)	$\mathcal{A} \vdash \mathbf{F}\vee\mathcal{B} \} \mathcal{A} \vdash \mathcal{B}$
($\neg$-L)	$\mathcal{A} \vdash \mathcal{C}\vee\mathcal{B} \} \mathcal{A}\wedge\neg\mathcal{C} \vdash \mathcal{B}$
($\neg$-R)	$\mathcal{A}\wedge\mathcal{C} \vdash \mathcal{B} \} \mathcal{A} \vdash \neg\mathcal{C}\vee\mathcal{B}$

The standard deduction rules of propositional logic, both for the sequent calculus and for natural deduction (interpreting "," as $\wedge$ on the left and $\vee$ on the right), are derivable from the rules in the table.

4.2.2 Composition

The logical rules of composition apply not only to our calculus but also to any calculus that includes a standard process composition operator, for example, CCS.

Composition Rules

($\mid \mathbf{0}$)	$\} \mathcal{A} \mid \mathbf{0} \dashv\vdash \mathcal{A}$
($\mid \neg\mathbf{0}$)	$\} \mathcal{A} \mid \neg\mathbf{0} \vdash \neg\mathbf{0}$
(A $\mid$)	$\} \mathcal{A} \mid (\mathcal{B} \mid \mathcal{C}) \dashv\vdash (\mathcal{A} \mid \mathcal{B}) \mid \mathcal{C}$
(X $\mid$)	$\} \mathcal{A} \mid \mathcal{B} \vdash \mathcal{B} \mid \mathcal{A}$
($\mid \vdash$)	$\mathcal{A}' \vdash \mathcal{B}';\ \mathcal{A}'' \vdash \mathcal{B}'' \} \mathcal{A}' \mid \mathcal{A}'' \vdash \mathcal{B}' \mid \mathcal{B}''$
($\mid \vee$)	$\} (\mathcal{A}\vee\mathcal{B}) \mid \mathcal{C} \vdash \mathcal{A} \mid \mathcal{C} \vee \mathcal{B} \mid \mathcal{C}$
($\mid \Vert$)	$\} \mathcal{A}' \mid \mathcal{A}'' \vdash (\mathcal{A}' \mid \mathcal{B}'') \vee (\mathcal{B}' \mid \mathcal{A}'') \vee (\neg\mathcal{B}' \mid \neg\mathcal{B}'')$
($\mid \triangleright$)	$\mathcal{A} \mid \mathcal{C} \vdash \mathcal{B} \{\} \mathcal{A} \vdash \mathcal{C}{\triangleright}\mathcal{B}$

The first two rules assert that $\mathbf{0}$ is part of any process, and that if a part is non-$\mathbf{0}$ so is the whole. The next three rules give associativity, commutativity, and congruence of composition.

The converse of the $\mid$-$\vee$ distribution rule ($\mid \vee$), namely $\mathcal{A} \mid \mathcal{C} \vee \mathcal{B} \mid \mathcal{C} \vdash (\mathcal{A}\vee\mathcal{B}) \mid \mathcal{C}$, is derivable. So is a $\mid$-$\wedge$ distribution rule, $(\mathcal{A}\wedge\mathcal{B})$

$\mid \mathcal{C} \vdash \mathcal{A} \mid \mathcal{C} \wedge \mathcal{B} \mid \mathcal{C}$. However, the converse of that, namely $\mathcal{A} \mid \mathcal{C} \wedge \mathcal{B} \mid \mathcal{C} \vdash (\mathcal{A}\wedge\mathcal{B}) \mid \mathcal{C}$, is not sound. (Take $\mathcal{A} = n[m[\mathbf{T}]]$, $\mathcal{B} = n[p[\mathbf{T}]]$, $\mathcal{C} = n[\mathbf{T}]$, and $P = n[m[]] \mid n[p[]]$; then $P \vDash \mathcal{A} \mid \mathcal{C}$ and $P \vDash \mathcal{B} \mid \mathcal{C}$, but $\neg\, P \vDash (\mathcal{A}\wedge\mathcal{B}) \mid \mathcal{C}$.) As a consequence, one cannot always "push $\mid$ inside $\wedge$" on the left-hand side of a sequent. In particular, after an application of $(\mid \vdash)$ one cannot in general renormalize a sequent to bring $\wedge$ (or ",") to the top level.

The decomposition axiom, $(\mid \|)$, can be used to analyze a composition $\mathcal{A}' \mid \mathcal{A}''$ with respect to arbitrarily chosen $\mathcal{B}'$ and $\mathcal{B}''$. An easy consequence of it is $\neg(\mathcal{A} \mid \mathcal{B}) \vdash (\mathcal{A} \mid \mathbf{T}) \Rightarrow (\mathbf{T} \mid \neg\mathcal{B})$, which means that if a process cannot be decomposed into parts that satisfy $\mathcal{A}$ and $\mathcal{B}$, but can be decomposed in such a way that a part satisfies $\mathcal{A}$, then it can also be decomposed in such a way that a part does not satisfy $\mathcal{B}$. An even simpler consequence is that $\neg(\mathbf{T} \mid \mathcal{B}) \vdash \mathbf{T} \mid \neg\mathcal{B}$, which is one of the few cases in which one can push $\neg$ across $\mid$.

The rule $(\mid \triangleright)$ states that $\mathcal{A} \mid \mathcal{B}$ and $\mathcal{A}\triangleright\mathcal{B}$ are logical adjuncts[1]. This has a large number of interesting consequences, most of them deriving from the adjunction along standard lines.

Some Composition Corollaries

$(\triangleright \vdash)$	$\mathcal{A}' \vdash \mathcal{A};\ \mathcal{B} \vdash \mathcal{B}' \ \}\ \mathcal{A}\triangleright\mathcal{B} \vdash \mathcal{A}'\triangleright\mathcal{B}'$	
$(\triangleright \mid)$	$\}\ (\mathcal{A}\triangleright\mathcal{B}) \mid \mathcal{A} \vdash \mathcal{B}$	
$(\triangleright\triangleright)$	$\}\ (\mathcal{A}\triangleright\mathcal{B}) \mid (\mathcal{B}\triangleright\mathcal{C}) \vdash \mathcal{A}\triangleright\mathcal{C}$	
$(\triangleright$-L$)$	$\mathcal{D} \vdash \mathcal{A};\ \mathcal{B} \vdash \mathcal{C}\ \}\ \mathcal{D} \mid (\mathcal{A}\triangleright\mathcal{B}) \vdash \mathcal{C}$	
$(\mid \mathbf{T})$	$\}\ \mathcal{A} \vdash \mathcal{A} \mid \mathbf{T}$	
$(\mid \mathbf{F})$	$\}\ \mathcal{A} \mid \mathbf{F} \vdash \mathbf{F}$	
$(\mid \wedge)$	$\}\ (\mathcal{A}\wedge\mathcal{B}) \mid \mathcal{C} \vdash \mathcal{A} \mid \mathcal{C} \wedge \mathcal{B} \mid \mathcal{C}$	
$(\mid \vee)$	$\}\ \mathcal{A} \mid \mathcal{C} \vee \mathcal{B} \mid \mathcal{C} \vdash (\mathcal{A}\vee\mathcal{B}) \mid \mathcal{C}$	
$(\mathbf{T} \triangleright)$	$\}\ \mathbf{T}\triangleright\mathcal{A} \vdash \mathcal{A}$	
$(\mathbf{F} \triangleright)$	$\}\ \mathbf{T} \vdash \mathbf{F}\triangleright\mathcal{A}$	
$(\triangleright\wedge\vee)$	$\}\ \mathcal{A}\triangleright\mathcal{B} \vdash (\mathcal{A}\wedge\mathcal{C})\triangleright\mathcal{B}.$	$\}\ \mathcal{A}\triangleright(\mathcal{B}\wedge\mathcal{C}) \dashv\vdash \mathcal{A}\triangleright\mathcal{B} \wedge \mathcal{A}\triangleright\mathcal{C}$
	$\}\ \mathcal{A}\triangleright(\mathcal{C}\wedge\mathcal{B}) \vdash \mathcal{A}\triangleright\mathcal{B}.$	$\}\ (\mathcal{A}\vee\mathcal{C})\triangleright\mathcal{B} \dashv\vdash \mathcal{A}\triangleright\mathcal{B} \wedge \mathcal{C}\triangleright\mathcal{B}$
	$\}\ \mathcal{A}\triangleright\mathcal{B} \vdash \mathcal{A}\triangleright(\mathcal{C}\vee\mathcal{B}).$	$\}\ \mathcal{A}\triangleright\mathcal{B} \vee \mathcal{A}\triangleright\mathcal{C} \vdash \mathcal{A}\triangleright(\mathcal{B}\vee\mathcal{C})$
	$\}\ (\mathcal{A}\vee\mathcal{C})\triangleright\mathcal{B} \vdash \mathcal{A}\triangleright\mathcal{B}.$	$\}\ \mathcal{A}\triangleright\mathcal{B} \vee \mathcal{C}\triangleright\mathcal{B} \vdash (\mathcal{A}\wedge\mathcal{C})\triangleright\mathcal{B}$

It is worth pointing out that some composition rules produce interesting interactions between the $\wedge$ and $\mid$ fragments of the logic. For example, $(\mathcal{A} \mid \mathcal{B}) \wedge \mathbf{0} \vdash \mathcal{A}$ is derivable using $(\mid \|)$ and $(\mid \neg\mathbf{0})$.

4.2.3 Locations

The location rules are specific to calculi with tree-structured locations, such as the Ambient Calculus.

Location Rules

$(n[]\ \neg\mathbf{0})$	$\}\ n[\mathcal{A}] \vdash \neg\mathbf{0}$
$(n[]\ \neg\mid)$	$\}\ n[\mathcal{A}] \vdash \neg(\neg\mathbf{0} \mid \neg\mathbf{0})$
$(n[] \vdash)$	$\mathcal{A} \vdash \mathcal{B}\ \{\}\ n[\mathcal{A}] \vdash n[\mathcal{B}]$
$(n[]\ \wedge)$	$\}\ n[\mathcal{A}]\wedge n[\mathcal{B}] \vdash n[\mathcal{A}\wedge\mathcal{B}]$
$(n[]\ \vee)$	$\}\ n[\mathcal{A}\vee\mathcal{B}] \vdash n[\mathcal{A}]\vee n[\mathcal{B}]$
$(n[]\ @)$	$n[\mathcal{A}] \vdash \mathcal{B}\ \{\}\ \mathcal{A} \vdash \mathcal{B}@n$
$(\neg\ @)$	$\}\ \mathcal{A}@n \dashv\vdash \neg((\neg\mathcal{A})@n)$

The first two rules assert that locations are non-void and are not decomposable. The next three rules give congruence and distributivity of locations with respect to $\wedge$ and $\vee$. The rule $(n[]\ @)$ states that $\mathcal{A}@n$ and $n[\mathcal{A}]$ are adjuncts, and the rule $(\neg\ @)$ states that the location adjunct $@$ is self-dual.

Note that $(n[] \vdash)$ holds in both directions, and that the inverse directions of $(n[]\ \wedge)$ and $(n[]\ \vee)$ are derivable; hence, the location fragment of the logic is particularly simple to handle.

Some Location Corollaries

$(n[]\ \mathbf{F})$	$\}\ n[\mathbf{F}] \vdash \mathbf{F}$
$(n[]\ \wedge)$	$\}\ n[\mathcal{A}\wedge\mathcal{B}] \vdash n[\mathcal{A}]\wedge n[\mathcal{B}]$
$(n[]\ \vee)$	$\}\ n[\mathcal{A}]\vee n[\mathcal{B}] \vdash n[\mathcal{A}\vee\mathcal{B}]$
$(@ \vdash)$	$\mathcal{A} \vdash \mathcal{B}\ \}\ \mathcal{A}@n \vdash \mathcal{B}@n$
$(n[\mathcal{A}@n])$	$\}\ n[\mathcal{A}@n] \vdash \mathcal{A}$
$(n[\mathcal{A}]@n)$	$\}\ \mathcal{A} \dashv\vdash n[\mathcal{A}]@n$
$(n[\neg\mathcal{A}])$	$\}\ n[\neg\mathcal{A}] \vdash \neg n[\mathcal{A}]$
$(\neg n[\mathcal{A}])$	$\}\ \neg n[\mathcal{A}] \dashv\vdash n[\mathbf{T}] \Rightarrow n[\neg\mathcal{A}]$

4.2.4 Time and Space Modalities

The "somewhere" modality was our starting point in developing our logic. We can now investigate its properties.

Time and Space Modality Rules

$(\Diamond)$	$\}\ \Diamond\mathcal{A} \dashv\vdash \neg\Box\neg\mathcal{A}$	(✧)	$\}\ \text{✧}\mathcal{A} \dashv\vdash \neg\text{¤}\neg\mathcal{A}$
$(\Box\ \mathrm{K})$	$\}\ \Box(\mathcal{A}\Rightarrow\mathcal{B}) \vdash \Box\mathcal{A}\Rightarrow\Box\mathcal{B}$	$(\text{¤}\ \mathrm{K})$	$\}\ \text{¤}(\mathcal{A}\Rightarrow\mathcal{B}) \vdash \text{¤}\mathcal{A}\Rightarrow\text{¤}\mathcal{B}$
$(\Box\ \mathrm{T})$	$\}\ \Box\mathcal{A} \vdash \mathcal{A}$	$(\text{¤}\ \mathrm{T})$	$\}\ \text{¤}\mathcal{A} \vdash \mathcal{A}$
$(\Box\ 4)$	$\}\ \Box\mathcal{A} \vdash \Box\Box\mathcal{A}$	$(\text{¤}\ 4)$	$\}\ \text{¤}\mathcal{A} \vdash \text{¤}\text{¤}\mathcal{A}$
$(\Box\ \mathbf{T})$	$\}\ \mathbf{T} \vdash \Box\mathbf{T}$	$(\text{¤}\ \mathbf{T})$	$\}\ \mathbf{T} \vdash \text{¤}\mathbf{T}$
$(\Box \vdash)$	$\mathcal{A} \vdash \mathcal{B}\ \}\ \Box\mathcal{A} \vdash \Box\mathcal{B}$	$(\text{¤} \vdash)$	$\mathcal{A} \vdash \mathcal{B}\ \}\ \text{¤}\mathcal{A} \vdash \text{¤}\mathcal{B}$
$(\Diamond n[])$	$\}\ n[\Diamond\mathcal{A}] \vdash \Diamond n[\mathcal{A}]$	$(\text{✧} n[])$	$\}\ n[\text{✧}\mathcal{A}] \vdash \text{✧}\mathcal{A}$
$(\Diamond \mid)$	$\}\ \Diamond\mathcal{A} \mid \Diamond\mathcal{B} \vdash \Diamond(\mathcal{A} \mid \mathcal{B})$	$(\text{✧} \mid)$	$\}\ \text{✧}\mathcal{A} \mid \mathcal{B} \vdash \text{✧}(\mathcal{A} \mid \mathbf{T})$
$(\text{✧}\Diamond)$	$\}\ \text{✧}\Diamond\mathcal{A} \vdash \Diamond\text{✧}\mathcal{A}$		

The operators $\Diamond$ and ✧ obey the rules of S4 modalities (the first 6 rules in each column); these follow simply from reflexivity and transitivity of $\longrightarrow^*$ and $\downarrow^*$. These operators, however, are not S5 modalities, that is, $\Diamond\mathcal{A} \vdash \Box\Diamond\mathcal{A}$ is not valid (if $\mathcal{A}$ may happen along some reduction branch, it will not necessarily happen starting from every reduction point), and neither is $\text{✧}\mathcal{A} \vdash \text{¤}\text{✧}\mathcal{A}$ (if $\mathcal{A}$ holds in some sublocation, it does not necessarily hold in some sublocation of every sublocation).

The modalities differ prominently in the way they distribute over compositions and locations, as seen in the subsequent 4 rules.

The last rule shows that the two modalities permute in one direction: somewhere sometime implies sometime somewhere. But the other direction is not sound. (Consider $P = (open\ n.\ m[p[]]) \mid n[]$. Then $P \vDash \Diamond\text{✧}p[\mathbf{0}]$, but $P \nvDash \text{✧}\Diamond p[\mathbf{0}]$).

Some Modality Corollaries

$(\Diamond \vdash)$	$\mathcal{A} \vdash \mathcal{B}\ \}\ \Diamond\mathcal{A} \vdash \Diamond\mathcal{B}$	$(\text{✧} \vdash)$	$\mathcal{A} \vdash \mathcal{B}\ \}\ \text{✧}\mathcal{A} \vdash \text{✧}\mathcal{B}$
$(\Box\ \wedge)$	$\}\ \Box(\mathcal{A}\wedge\mathcal{B}) \dashv\vdash \Box\mathcal{A}\wedge\Box\mathcal{B}$	$(\text{¤}\ \wedge)$	$\}\ \text{¤}(\mathcal{A}\wedge\mathcal{B}) \dashv\vdash \text{¤}\mathcal{A}\wedge\text{¤}\mathcal{B}$
$(\Diamond\ \mathrm{T})$	$\}\ \mathcal{A} \vdash \Diamond\mathcal{A}$	$(\text{✧}\ \mathrm{T})$	$\}\ \mathcal{A} \vdash \text{✧}\mathcal{A}$
$(\Box\ \Diamond)$	$\}\ \Box\mathcal{A} \vdash \Diamond\mathcal{A}$	$(\text{¤}\ \text{✧})$	$\}\ \text{¤}\mathcal{A} \vdash \text{✧}\mathcal{A}$
$(\Diamond\ \mathrm{K})$	$\}\ \Diamond\mathcal{A}\Rightarrow\Diamond\mathcal{B} \vdash \Diamond(\mathcal{A}\Rightarrow\mathcal{B})$	$(\text{✧}\ \mathrm{K})$	$\}\ \text{✧}\mathcal{A}\Rightarrow\text{✧}\mathcal{B} \vdash \text{✧}(\mathcal{A}\Rightarrow\mathcal{B})$
$(\Diamond\ 4)$	$\}\ \Diamond\Diamond\mathcal{A} \vdash \Diamond\mathcal{A}$	$(\text{✧}\ 4)$	$\}\ \text{✧}\text{✧}\mathcal{A} \vdash \text{✧}\mathcal{A}$
$(\Diamond\ \vee)$	$\}\ \Diamond(\mathcal{A}\vee\mathcal{B}) \dashv\vdash \Diamond\mathcal{A}\vee\Diamond\mathcal{B}$	$(\text{✧}\ \vee)$	$\}\ \text{✧}(\mathcal{A}\vee\mathcal{B}) \dashv\vdash \text{✧}\mathcal{A}\vee\text{✧}\mathcal{B}$
$(\Diamond\ \mathbf{F})$	$\}\ \Diamond\mathbf{F} \vdash \mathbf{F}$	$(\text{✧}\ \mathbf{F})$	$\}\ \text{✧}\mathbf{F} \vdash \mathbf{F}$

1. We say that two binary operators $\text{⬠},\text{⬡}$ are logical adjuncts if $\mathcal{A}\text{⬠}\mathcal{C} \vdash \mathcal{B}\ \{\}\ \mathcal{A} \vdash \mathcal{C}\text{⬡}\mathcal{B}$. The main adjunction of logic is given by the pair $\wedge,\Rightarrow$. Moreover, we say that two unary operators $\text{⬠},\text{⬡}$ are logical adjuncts if $\text{⬠}\mathcal{A} \vdash \mathcal{B}\ \{\}\ \mathcal{A} \vdash \text{⬡}\mathcal{B}$.

$(\square ✧)$	$\} \; \square ✧ \mathcal{A} \vdash ✧ \square \mathcal{A}$	
$(\square\ n[])$	$\} \; \square n[\mathcal{A}] \vdash n[\square \mathcal{A}]$	
$(\square\ @)$	$\} \; (\square \mathcal{A})@n \vdash \mathcal{A}@n$	
$(\Diamond\ @)$	$\} \; \mathcal{A}@n \vdash (\Diamond \mathcal{A})@n.$	$\} \; \Diamond(\mathcal{A}@n) \dashv\vdash (\Diamond \mathcal{A})@n$
$(\square \rhd)$	$\} \; \mathcal{A} \rhd \mathcal{B} \vdash (\square \mathcal{A}) \rhd \mathcal{B}$	
$(\Diamond \rhd)$	$\} \; (\Diamond \mathcal{A}) \rhd \mathcal{B} \vdash \mathcal{A} \rhd \mathcal{B}.$	$\} \; \Diamond(\mathcal{A} \rhd \mathcal{B}) \vdash (\Diamond \mathcal{A}) \rhd (\Diamond \mathcal{B})$

4.2.5 Satisfiability

Validity and satisfiability can be reflected into the logic by means of the $\mathcal{A}^{\mathbf{F}}$ operator (here we use $\mathcal{A}^{\neg}$ for $\neg\mathcal{A}$):

$\mathcal{A}^{\mathbf{F}}$	$\triangleq$	$\mathcal{A} \rhd \mathbf{F}$	$\mathcal{A}$ is unsatisfiable
$Vld\, \mathcal{A}$	$\triangleq$	$\mathcal{A}^{\neg\mathbf{F}}$	$\mathcal{A}$ is valid
$Sat\, \mathcal{A}$	$\triangleq$	$\mathcal{A}^{\mathbf{F}\neg}$	$\mathcal{A}$ is satisfiable
$P \vDash \mathcal{A}^{\mathbf{F}}$			iff $\forall P'{:}\Pi.\ \neg P' \vDash \mathcal{A}$
$P \vDash Vld\, \mathcal{A}$			iff $\forall P'{:}\Pi.\ P' \vDash \mathcal{A}$
$P \vDash Sat\, \mathcal{A}$			iff $\exists P'{:}\Pi.\ P' \vDash \mathcal{A}$

From the definitions of $\rhd$ and $\mathbf{F}$, we obtain that $P \vDash \mathcal{A}^{\mathbf{F}} \Leftrightarrow (\forall P'{:}\Pi.\ P' \vDash \mathcal{A} \Rightarrow P|P' \vDash \mathbf{F}) \Leftrightarrow (\forall P'{:}\Pi.\ \neg P' \vDash \mathcal{A})$. I.e., $P \vDash \mathcal{A}^{\mathbf{F}}$ iff $\mathcal{A}$ is unsatisfiable, independently of P.

One of the main properties of $\mathcal{A}^{\mathbf{F}}$ is that $\mathcal{A} \mid \mathcal{A}^{\mathbf{F}} \vdash \mathbf{F}$, by $(\rhd \mid)$. That is, $\mathcal{A}$ cannot be both satisfiable and unsatisfiable. In addition we obtain, from the model, the following rules, from which it is possible to show within the logic that *Vld* and *Sat* obey the rules of S5 modal operators:

Satisfiability Rules

$(\rhd\mathbf{F}\ \neg)$	$\} \; \mathcal{A}^{\mathbf{F}} \vdash \mathcal{A}^{\neg}$	if $\mathcal{A}$ is unsatisfiable then $\mathcal{A}$ is false
$(\neg \rhd\mathbf{F})$	$\} \; \mathcal{A}^{\mathbf{F}\neg} \vdash \mathcal{A}^{\mathbf{F}\mathbf{F}}$	if $\mathcal{A}$ is satisfiable then $\mathcal{A}^{\mathbf{F}}$ is not

Some Satisfiability Corollaries

$(\mid \rhd\mathbf{F})$	$\} \; \mathcal{A} \mid \mathcal{A}^{\mathbf{F}} \vdash \mathbf{F}$	
$(\rhd\mathbf{F} \vdash)$	$\mathcal{B} \vdash \mathcal{A} \; \} \; \mathcal{A}^{\mathbf{F}} \vdash \mathcal{B}^{\mathbf{F}}$	
$(\rhd\mathbf{F} \rhd)$	$\} \; \mathcal{B} \rhd \mathcal{A} \vdash \mathcal{A}^{\mathbf{F}} \rhd \mathcal{B}^{\mathbf{F}}$	
$(\mathbf{F} \rhd\mathbf{F})$	$\} \; \mathbf{T} \dashv\vdash \mathbf{F}^{\mathbf{F}}$	
$(\mathbf{T} \rhd\mathbf{F})$	$\} \; \mathbf{F} \dashv\vdash \mathbf{T}^{\mathbf{F}}$	
$(\neg \rhd\mathbf{F})$	$\} \; \mathcal{A}^{\neg\mathbf{F}} \vdash \mathcal{A}^{\neg\neg}.$	$\} \; \mathcal{A}^{\mathbf{F}\mathbf{F}} \vdash \mathcal{A}^{\mathbf{F}\neg}$
	$\} \; \mathcal{A}^{\neg\mathbf{F}} \vdash \mathcal{A}^{\mathbf{F}\mathbf{F}}.$	$\} \; \mathcal{A}^{\neg\neg} \vdash \mathcal{A}^{\mathbf{F}\neg}$

4.2.6 Predicates

So far we have considered only propositional validity; when considering quantifiers, we need to extend our notion of validity. If $fv(\mathcal{A})=\{x_1, ..., x_k\}$ are the free variables of $\mathcal{A}$ and $\varphi \in fv(\mathcal{A}) \rightarrow \Lambda$ is a substitution of variables for names, we write $\mathcal{A}_\varphi$ for $\mathcal{A}\{x_1 \leftarrow \varphi(x_1), ..., x_k \leftarrow \varphi(x_k)\}$, and we define:

$$\boldsymbol{vld}(\mathcal{A}) \ \triangleq\ \forall \varphi \in fv(\mathcal{A}) \rightarrow \Lambda.\ \forall P{:}\Pi.\ P \vDash \mathcal{A}_\varphi$$

This definition of predicate validity generalizes the previous definition of $\boldsymbol{vld}$, which was restricted to the case of $fv(\mathcal{A}) = \emptyset$. It similarly generalizes the definitions of sequents and rules.

We can now introduce quantifiers and their rules:

Quantifier Rules

$(\forall\text{-L})$	$\mathcal{A}\{x \leftarrow \eta\} \vdash \mathcal{B} \; \} \; \forall x.\mathcal{A} \vdash \mathcal{B}$	(η a name or a variable)
$(\forall\text{-R})$	$\mathcal{A} \vdash \mathcal{B} \; \} \; \mathcal{A} \vdash \forall x.\mathcal{B}$	where $x \notin fv(\mathcal{A})$

As an example, $✧\forall x.\neg(x[\mathbf{T}]^{\exists})$ is the formula for "somewhere there are no ambients". Since there are no infinite spatial paths $P_1 \downarrow P_2 \downarrow P_3 \downarrow ...$, we can show in the model that this formula is valid. On the other hand, its temporal dual, "sometime there are no ambients", $\Diamond\forall x.\neg(x[\mathbf{T}]^{\exists})$, is invalid; for instance, it is not satisfied by $n[]$.

The following lemma yields a substitution principle for predicate validity, allowing us to replace logically equivalent formulas in larger contexts. Let $\mathcal{B}\{-\}$ be a formula with a set of formula holes, indicated by $-$, and let $\mathcal{B}\{\mathcal{A}\}$ denote the capture-avoiding substitution of $\mathcal{A}$ for the holes in $\mathcal{B}\{-\}$.

4-1 Lemma (Substitution)

$\boldsymbol{vld}(\mathcal{A}' \Leftrightarrow \mathcal{A}'') \Rightarrow \boldsymbol{vld}(\mathcal{B}\{\mathcal{A}'\} \Leftrightarrow \mathcal{B}\{\mathcal{A}''\})$

□

4-2 Corollary (Substitution Principle)

$\mathcal{A}' \dashv\vdash \mathcal{A}'' \Rightarrow \mathcal{B}\{\mathcal{A}'\} \dashv\vdash \mathcal{B}\{\mathcal{A}''\}$

□

4.2.7 Name Equality

It is possible to encode name equality within the logic in terms of location adjuncts, by taking:

$$\eta = \mu \ \triangleq\ \eta[\mathbf{T}]@\mu$$

We obtain, for all $\varphi \in fv(\eta) \cup fv(\mu) \rightarrow \Lambda$ and all $P{:}\Pi$:

$$P \vDash (\eta = \mu)_\varphi \ \Leftrightarrow\ \varphi(\eta) = \varphi(\mu)$$

As an example, the following formula means "any two ambients here have different names", which can be read as a no-spoofing security property:

$$\forall x.\ \forall y.\ x[\mathbf{T}] \mid y[\mathbf{T}] \mid \mathbf{T} \Rightarrow \neg\, x = y$$

4.2.8 Lifting Propositional Validity

Using equality, we can extend propositional validity to predicate validity in the sense of the proposition proved at the end of this section, Proposition 4-9. This way, we can systematically extend to predicate logic the rules we have derived so far for propositional logic.

To prove this proposition, we need renaming lemmas for satisfaction, Lemma 4-6, and for validity, Lemmas 4-7 and 4-8. First, we state three auxiliary lemmas.

4-3 Lemma (Fresh renaming preserves ≡)

Consider any process P and names m, m', with $m' \notin fn(P)$. For all P', if $P \equiv P'$ then $m' \notin fn(P')$ and $P\{m \leftarrow m'\} \equiv P'\{m \leftarrow m'\}$. Moreover, for all Q, if $P\{m \leftarrow m'\} \equiv Q$ then there is a P' with $P \equiv P'$, $m' \notin fn(P')$ and $Q = P'\{m \leftarrow m'\}$.

□

4-4 Lemma (Fresh renaming preserves →)

Consider any process P and names m, m', with $m' \notin fn(P)$. For all P', if $P \rightarrow P'$ then $m' \notin fn(P')$ and $P\{m \leftarrow m'\} \rightarrow P'\{m \leftarrow m'\}$. Moreover, for all Q, if $P\{m \leftarrow m'\} \rightarrow Q$ then there is a P' with $P \rightarrow P'$, $m' \notin fn(P')$ and $Q = P'\{m \leftarrow m'\}$.

□

4-5 Lemma (Fresh renaming preserves ↓)

Consider any process P and names m, m', with $m' \notin fn(P)$. For all P', if $P \downarrow P'$ then $m' \notin fn(P')$ and $P\{m \leftarrow m'\} \downarrow P'\{m \leftarrow m'\}$. Moreover, for all Q, if $P\{m \leftarrow m'\} \downarrow Q$ then there is a P' with $P \downarrow P'$, $m' \notin fn(P')$ and $Q = P'\{m \leftarrow m'\}$.

□

4-6 Lemma (Fresh renaming preserves $\vDash$)

For all closed formulas $\mathcal{A}$, processes P, and names m, m', if $m'\notin fn(P)\cup fn(\mathcal{A})$ then $P \vDash \mathcal{A} \Leftrightarrow P\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}$.

Proof

The proof is by induction on the number of symbols in the closed formula $\mathcal{A}$. Note that the number of symbols in a formula is unchanged by substituting a name for a variable or another name. Consider an arbitrary process P, and any names m and m'. If $m=m'$ the lemma holds trivially, so we may assume that $m\neq m'$. We show only the case for parallel composition and the case for universal quantification.

Case for |: We prove each half of the following separately, where $m'\notin fn(P)\cup fn(\mathcal{A} \mid \mathcal{B})$.

$$P \vDash \mathcal{A} \mid \mathcal{B} \Leftrightarrow P\{m\leftarrow m'\} \vDash (\mathcal{A} \mid \mathcal{B})\{m\leftarrow m'\}.$$

($\Rightarrow$) Assume $P \vDash \mathcal{A} \mid \mathcal{B}$. We are to show that there are Q', Q'' such that $P\{m\leftarrow m'\} \equiv Q' \mid Q''$, $Q' \vDash \mathcal{A}\{m\leftarrow m'\}$, and $Q'' \vDash \mathcal{B}\{m\leftarrow m'\}$. By assumption, there are P', P'' such that $P \equiv P' \mid P''$, $P' \vDash \mathcal{A}$, and $P'' \vDash \mathcal{B}$. Let $Q' = P'\{m\leftarrow m'\}$ and $Q'' = P''\{m\leftarrow m'\}$. By Lemma 4-3, $P \equiv P' \mid P''$ and $m'\notin fn(P)$ imply that $m'\notin fn(P')\cup fn(P'')$ and $P\{m\leftarrow m'\} \equiv Q' \mid Q''$. By induction hypothesis, $m'\notin fn(P')\cup fn(\mathcal{A})$ and $P' \vDash \mathcal{A}$ imply that $Q' \vDash \mathcal{A}\{m\leftarrow m'\}$, and also $m'\notin fn(P'')\cup fn(\mathcal{B})$ and $P'' \vDash \mathcal{B}$ imply that $Q'' \vDash \mathcal{B}\{m\leftarrow m'\}$.

($\Leftarrow$) Assume $P\{m\leftarrow m'\} \vDash (\mathcal{A} \mid \mathcal{B})\{m\leftarrow m'\}$. We are to show that there are P', P'' such that $P \equiv P' \mid P''$, $P' \vDash \mathcal{A}$, and $P'' \vDash \mathcal{B}$. By assumption, there are Q', Q'' such that $P\{m\leftarrow m'\} \equiv Q' \mid Q''$, $Q' \vDash \mathcal{A}\{m\leftarrow m'\}$, and $Q'' \vDash \mathcal{B}\{m\leftarrow m'\}$. By Lemma 4-3, $P\{m\leftarrow m'\} \equiv Q' \mid Q''$ and $m'\notin fn(P)$ imply there is R with $P \equiv R$, $m'\notin fn(R)$ and $Q' \mid Q'' = R\{m\leftarrow m'\}$, and hence that there are P', P'' such that $R = P' \mid P''$, $m'\notin fn(P')$, $m'\notin fn(P'')$, $Q' = P'\{m\leftarrow m'\}$, and $Q'' = P''\{m\leftarrow m'\}$. By induction hypothesis, $m'\notin fn(P')\cup fn(\mathcal{A})$ and $Q' \vDash \mathcal{A}\{m\leftarrow m'\}$ imply that $P' \vDash \mathcal{A}$, and also $m'\notin fn(P'')\cup fn(\mathcal{B})$ and $Q'' \vDash \mathcal{B}\{m\leftarrow m'\}$ imply that $P'' \vDash \mathcal{B}$.

Case for $\forall$: We prove each direction of the following separately, where $m'\notin fn(P)\cup fn(\forall x.\mathcal{A})$.

$$P \vDash \forall x.\mathcal{A} \Leftrightarrow P\{m\leftarrow m'\} \vDash (\forall x.\mathcal{A})\{m\leftarrow m'\}.$$

($\Rightarrow$) Assume $P \vDash \forall x.\mathcal{A}$. Pick any name n. We are to show that $P\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow n\}$. We split the proof into three cases.

First, suppose that $m=n$. Pick a fresh name m'' such that $m''\notin fn(P)\cup fn(\mathcal{A})\cup\{m,m'\}$. By assumption, $P \vDash \mathcal{A}\{x\leftarrow m''\}$. Since $m'\notin fn(P)\cup fn(\mathcal{A}\{x\leftarrow m''\})$, the induction hypothesis implies that $P\{m\leftarrow m'\} \vDash \mathcal{A}\{x\leftarrow m''\}\{m\leftarrow m'\}$. Recall that $m\neq m'$. Then, since $m\notin fn(P\{m\leftarrow m'\})$ and $m\notin fn(\mathcal{A}\{x\leftarrow m''\}\{m\leftarrow m'\})$, we get that $P\{m\leftarrow m'\}\{m''\leftarrow m\} \vDash \mathcal{A}\{x\leftarrow m''\}\{m\leftarrow m'\}\{m''\leftarrow m\}$ by a second application of the induction hypothesis. But because of the freshness of m'', we have $P\{m\leftarrow m'\}\{m''\leftarrow m\} = P\{m\leftarrow m'\}$ and $\mathcal{A}\{x\leftarrow m''\}\{m\leftarrow m'\}\{m''\leftarrow m\} = \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow m\}$. Since $m=n$, we have shown $P\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow n\}$.

Second, suppose that $m\neq n$ and $m'=n$. By assumption, $P \vDash \mathcal{A}\{x\leftarrow m\}$. In general we know that $m'\notin fn(P)\cup fn(\mathcal{A})$ and $m\neq m'$. Therefore, we can apply the induction hypothesis to obtain $P\{m\leftarrow m'\} \vDash \mathcal{A}\{x\leftarrow m\}\{m\leftarrow m'\}$. We have $\mathcal{A}\{x\leftarrow m\}\{m\leftarrow m'\} = \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow m'\}$. Since $m'=n$, we have shown $P\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow n\}$.

Third, suppose that $m\neq n$ and $m'\neq n$. By assumption, $P \vDash \mathcal{A}\{x\leftarrow n\}$. We have that $m'\notin fn(P)\cup fn(\mathcal{A})$ and in this case we know that $m'\neq n$. Therefore, we can apply the induction hypothesis to obtain $P\{m\leftarrow m'\} \vDash \mathcal{A}\{x\leftarrow n\}\{m\leftarrow m'\}$. Since $m\neq n$ we have $\mathcal{A}\{x\leftarrow n\}\{m\leftarrow m'\} = \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow n\}$. So we have shown $P\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow n\}$.

($\Leftarrow$) Assume $P\{m\leftarrow m'\} \vDash (\forall x.\mathcal{A})\{m\leftarrow m'\}$. Pick any name n. We are to show that $P \vDash \mathcal{A}\{x\leftarrow n\}$. We split the proof into three cases.

First, suppose $n=m'$. Pick a fresh name m'' such that $m''\notin fn(P)\cup fn(\mathcal{A})\cup\{m,m'\}$. By assumption, we have $P\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow m''\}$. We can calculate $\mathcal{A}\{m\leftarrow m'\}\{x\leftarrow m''\} = \mathcal{A}\{x\leftarrow m''\}\{m\leftarrow m'\}$ since $m\neq m''$. Then, since $m'\notin fn(P)\cup fn(\mathcal{A}\{x\leftarrow m''\})$, the induction hypothesis implies $P \vDash \mathcal{A}\{x\leftarrow m''\}$. Again, since $m'\notin fn(P)$ and $m'\notin fn(\mathcal{A}\{x\leftarrow m''\})$, the induction hypothesis implies $P\{m''\leftarrow m'\} \vDash \mathcal{A}\{x\leftarrow m''\}\{m''\leftarrow m'\}$. But because of the freshness of m'', this is $P \vDash \mathcal{A}\{x\leftarrow m'\}$. Therefore, since $n=m'$, we have shown $P \vDash \mathcal{A}\{x\leftarrow n\}$.

Second, take $n\neq m'$ but $n=m$. By assumption, $P\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow m'\}$. From $m\neq m'$, we get $\mathcal{A}\{m\leftarrow m'\}\{x\leftarrow m'\} = \mathcal{A}\{x\leftarrow m'\}\{m\leftarrow m'\}$. Moreover, we also get $m\notin fn(P\{m\leftarrow m'\})$ and $m\notin fn(\mathcal{A}\{x\leftarrow m'\}\{m\leftarrow m'\})$. Hence, the induction hypothesis implies $P\{m\leftarrow m'\}\{m'\leftarrow m\} \vDash \mathcal{A}\{x\leftarrow m'\}\{m\leftarrow m'\}\{m'\leftarrow m\}$. Since $m'\notin fn(P)\cup fn(\mathcal{A})$, we can calculate $P\{m\leftarrow m'\}\{m'\leftarrow m\} = P$ and $\mathcal{A}\{x\leftarrow m'\}\{m\leftarrow m'\}\{m'\leftarrow m\} = \mathcal{A}\{x\leftarrow m\}$. Therefore, we have shown $P \vDash \mathcal{A}\{x\leftarrow n\}$.

Third, suppose $n\neq m'$ and $n\neq m$. By assumption, $P\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}\{x\leftarrow n\}$. Since $n\neq m$ we have $\mathcal{A}\{m\leftarrow m'\}\{x\leftarrow n\} = \mathcal{A}\{x\leftarrow n\}\{m\leftarrow m'\}$. Since $n\neq m'$, $m'\notin fn(P)\cup fn(\mathcal{A}\{x\leftarrow n\})$. Hence, the induction hypothesis implies $P \vDash \mathcal{A}\{x\leftarrow n\}$.

□

4-7 Lemma (Fresh renaming preserves validity)

If $\mathcal{A}$ is closed and valid and $m'\notin fn(\mathcal{A})$ then $\mathcal{A}\{m\leftarrow m'\}$ is closed and valid.

Proof

We can assume that $m'\neq m$. Take any P and two distinct names $n,n'\notin fn(P)\cup fn(\mathcal{A})\cup\{m,m'\}$. Since $\mathcal{A}$ is valid we have, in particular, that $P\{m\leftarrow n\}\{m'\leftarrow m\} \vDash \mathcal{A}$. By Lemma 4-6, since $m'\notin fn(P\{m\leftarrow n\}\{m'\leftarrow m\})\cup fn(\mathcal{A})$, we obtain $P\{m\leftarrow n\}\{m'\leftarrow m\}\{m\leftarrow m'\} \vDash \mathcal{A}\{m\leftarrow m'\}$. This is the same as $P\{m\leftarrow n\} \vDash \mathcal{A}\{m\leftarrow m'\}$. Again by Lemma 4-6, since $m\notin fn(P\{m\leftarrow n\})\cup fn(\mathcal{A}\{m\leftarrow m'\})$, we obtain $P\{m\leftarrow n\}\{n\leftarrow m\} \vDash \mathcal{A}\{m\leftarrow m'\}\{n\leftarrow m\}$. This is the same as $P \vDash \mathcal{A}\{m\leftarrow m'\}$. Hence $\mathcal{A}\{m\leftarrow m'\}$ is valid. Since $\mathcal{A}$ is closed, so is $\mathcal{A}\{m\leftarrow m'\}$.

□

4-8 Lemma (Injective complete renaming preserves validity)

If $\mathcal{A}$ is closed and valid and $\rho\in fn(\mathcal{A})\to\Lambda$ is an injective renaming, then $\mathcal{A}_\rho$ is closed and valid.

Proof

Let $\rho = \{m_1\leftarrow n_1, ..., m_k\leftarrow n_k\}$, where $\{m_1, ..., m_k\} = fn(\mathcal{A})$ and all the n_i are distinct. Take fresh $p_1, ..., p_k \notin \{m_1, n_1, ..., m_k, n_k\}$. By induction on i ranging from 1 to k, since $\mathcal{A}$ is closed and valid and $p_i\notin fn(\mathcal{A}\{m_1\leftarrow p_1\}...\{m_{i-1}\leftarrow p_{i-1}\})$, by using Lemma 4-7 at each step, we obtain that $\mathcal{A}' \triangleq \mathcal{A}\{m_1\leftarrow p_1\}...\{m_k\leftarrow p_k\}$ is closed and valid. Note that $fn(\mathcal{A}') = \{p_1, ..., p_k\}$. Then again, by induction on i ranging from 1 to k, since $n_i\notin fn(\mathcal{A}'\{p_1\leftarrow n_1\}...\{p_{i-1}\leftarrow n_{i-1}\})$, by using Lemma 4-7 at each step, we obtain that $\mathcal{A}'' \triangleq \mathcal{A}'\{p_1\leftarrow n_1\}...\{p_k\leftarrow n_k\}$ is closed and valid. Since $p_1, ..., p_k$ are fresh, $\mathcal{A}'' = \mathcal{A}_\rho$.

□

4-9 Proposition (Lifting propositional validity)

If $\mathcal{A}$ is closed and valid, then for any injective map $\psi\in fn(\mathcal{A})\to\vartheta$ from names to variables, the formula $(dfn(\mathcal{A})\Rightarrow\mathcal{A})_\psi$ is valid, where $dfn(\mathcal{A})$ is the conjunction of all inequalities $\neg n=m$ such that n,m are distinct names in $fn(\mathcal{A})$.

Proof

Assume that $\mathcal{A}$ is closed and valid and that $\psi\in fn(\mathcal{A})\to\vartheta$ is injective. By construction, we also have that $dfn(\mathcal{A})\Rightarrow\mathcal{A}$ is closed and valid. Take any $\varphi\in fv((dfn(\mathcal{A})\Rightarrow\mathcal{A})_{\psi})\to\Lambda$ (with $rng(\psi)=dom(\varphi)$) and consider $\varphi\circ\psi$. There are two cases. If φ is not injective then $dfn(\mathcal{A})_{\varphi\circ\psi}$ is equivalent to $\mathbf{F}$, and therefore $(dfn(\mathcal{A})\Rightarrow\mathcal{A})_{\varphi\circ\psi}$ is valid. Otherwise, if φ is injective, then $\varphi\circ\psi$ is also injective with $dom(\varphi\circ\psi) = fn(\mathcal{A}) = fn(dfn(\mathcal{A})\Rightarrow\mathcal{A})$. By Lemma 4-8, since $dfn(\mathcal{A})\Rightarrow\mathcal{A}$ is closed and valid, we have that $(dfn(\mathcal{A})\Rightarrow\mathcal{A})_{\varphi\circ\psi}$ is closed and valid. We have shown that $\forall\varphi\in fv((dfn(\mathcal{A})\Rightarrow\mathcal{A})_{\psi})\to\Lambda.\ \forall P{:}\Pi.\ P\vDash(dfn(\mathcal{A})\Rightarrow\mathcal{A})_{\varphi\circ\psi}$; that is, $\boldsymbol{vld}((dfn(\mathcal{A})\Rightarrow\mathcal{A})_{\psi})$.

□

For example, the valid proposition: $n[\mathbf{T}] \Rightarrow \neg m[\mathbf{T}]$ is transformed into the valid predicate $\neg x{=}y \Rightarrow (x[\mathbf{T}] \Rightarrow \neg y[\mathbf{T}])$. However, without the assumption $\neg x{=}y$, the predicate $x[\mathbf{T}] \Rightarrow \neg y[\mathbf{T}]$ is not valid: for predicate validity one must consider also the substitutions that map x and y to the same name.

4.2.9 *Case Analysis Principle*

When reasoning about equality, it is often convenient to reason by cases on whether the equality is true or false. To this end, we introduce a case analysis principle.

4-10 Definition (Classical Predicates)

$\mathcal{A}$ is classical iff $\forall\varphi\in fv(\mathcal{A})\to\Lambda.\ \{P \,\|\, P\vDash\mathcal{A}_{\varphi}\}\in\{\Pi,\varnothing\}$.

□

The predicates $\mathbf{T}$, $\mathbf{F}$, and $\eta{=}\mu$ are classical. So is the disjunction and negation of classical predicates.

4-11 Proposition (Case Analysis Principle)

Let $\mathcal{S}\{-\}$ be a sequent with a set of formula holes, and $\mathcal{A}$ be a classical predicate. Then $\mathcal{S}\{\mathbf{T}\}\wedge\mathcal{S}\{\mathbf{F}\}\Rightarrow\mathcal{S}\{\mathcal{A}\}$.

Proof

Taking $\mathcal{S}\{-\} = \mathcal{B}'\{-\}\vdash\mathcal{B}''\{-\}$ and $\mathcal{B}\{-\}\triangleq\mathcal{B}'\{-\}\Rightarrow\mathcal{B}''\{-\}$, it is sufficient to show that $\boldsymbol{vld}(\mathcal{B}\{\mathbf{T}\})\wedge\boldsymbol{vld}(\mathcal{B}\{\mathbf{F}\})\Rightarrow\boldsymbol{vld}(\mathcal{B}\{\mathcal{A}\})$. Assume $\boldsymbol{vld}(\mathcal{B}\{\mathbf{T}\})\wedge\boldsymbol{vld}(\mathcal{B}\{\mathbf{F}\})$. Take any $\varphi\in fv(\mathcal{B}\{\mathcal{A}\})\to\Lambda$ and $P{:}\Pi$. By assumption we have $P\vDash\mathcal{B}_{\varphi}\{\mathbf{T}\}$ and $P\vDash\mathcal{B}_{\varphi}\{\mathbf{F}\}$. Since $\mathcal{A}$ is classical, we have also that $\{Q \,\|\, Q\vDash\mathcal{A}_{\varphi}\}\in\{\Pi,\varnothing\}$. Consider the case where $\{Q \,\|\, Q\vDash\mathcal{A}_{\varphi}\}=\Pi$ so that for any P, $P\vDash\mathcal{A}_{\varphi}$ iff $P\vDash\mathbf{T}$. By Lemma 4-1, $P\vDash\mathcal{B}_{\varphi}\{\mathcal{A}_{\varphi}\}$ iff $P\vDash\mathcal{B}_{\varphi}\{\mathbf{T}\}$, hence we obtain $P\vDash\mathcal{B}_{\varphi}\{\mathcal{A}_{\varphi}\}$. Consider the case where $\{Q \,\|\, Q\vDash\mathcal{A}_{\varphi}\}=\varnothing$ so that for any P, $P\vDash\mathcal{A}_{\varphi}$ iff $P\vDash\mathbf{F}$. By Lemma 4-1, $P\vDash\mathcal{B}_{\varphi}\{\mathcal{A}_{\varphi}\}$ iff $P\vDash\mathcal{B}_{\varphi}\{\mathbf{F}\}$, hence we have $P\vDash\mathcal{B}_{\varphi}\{\mathcal{A}_{\varphi}\}$. In both cases, we have shown that $\forall\varphi\in fv(\mathcal{B}\{\mathcal{A}\})\to\Lambda.\ \forall P{:}\Pi.\ P\vDash\mathcal{B}_{\varphi}\{\mathcal{A}_{\varphi}\}$, that is, $\boldsymbol{vld}(\mathcal{B}\{\mathcal{A}\})$.

□

4.3 Logical Properties of Type Systems

In this section we briefly discuss applications of our logic to express properties guaranteed by type systems, beyond the standard statements of subject reduction. This section assumes knowledge of type systems for the Ambient Calculus [5].

Consider the system of locking and mobility types for the Ambient Calculus [5], recast for the calculus of this paper. The assumption $p{:}Amb^{\bullet}[S]$ ensures that ambients named p are locked, that is, they cannot be dissolved by an *open*. We can prove that if E, $p{:}Amb^{\bullet}[S], E'\vdash P : T$, then $P\vDash\Box(\diamond(p[\mathbf{T}]^{\exists})\Rightarrow\Box\diamond(p[\mathbf{T}]^{\exists}))$. This expresses that in a well-typed process, once a locked ambient named p somewhere comes into being, ever after there will somewhere be an ambient named p.

Moreover, the assumption $q{:}Amb^{\bullet}[^{\curlyvee}S']$ ensures that ambients named q are locked and immobile, that is, they cannot be moved by *in* or *out*, nor dissolved by *open*. We can prove that if E, $q{:}Amb^{\bullet}[^{\curlyvee}S'], E'\vdash P : T$, then $P\vDash\Box(q[\mathbf{T}]^{\exists}\Rightarrow\Box q[\mathbf{T}]^{\exists})$. This expresses that in a well-typed process, once a locked, immobile ambient appears at the top-level of the process, it will stay there ever after. Moreover, we can prove that if E, $p{:}Amb^{\bullet}[S]$, $q{:}Amb^{\bullet}[^{\curlyvee}S'], E'\vdash P : T$, then $P\vDash\Box(\diamond(p[q[\mathbf{T}]^{\exists}]^{\exists})\Rightarrow\Box\diamond(p[q[\mathbf{T}]^{\exists}]^{\exists}))$. This expresses that in a well-typed process, once a locked, immobile ambient named q is somewhere a child of a locked ambient named p, ever after there will somewhere be a q child of p.

4.4 An Example

In this example we use the laws of $\Diamond$, $|$, and $\triangleright$, to analyze the consequences of composing two logical specifications.

The specifications describe two subsystems: a *Shopper* and a *Thief*, and focus on what happens to the shopper's wallet. The wallet is described simply by the formula Wallet[$\mathbf{T}$], leaving the contents of the wallet unspecified. The absence of a wallet in a given location is described by the formula *NoWallet*, defined as $\neg$(Wallet[$\mathbf{T}$] | $\mathbf{T}$), meaning that it is not possible to decompose the current location into a part containing a wallet and some other part.

A thief is somebody who, in the direct presence of a wallet, can make the wallet disappear. Its specification is Wallet[$\mathbf{T}$] $\triangleright$ $\Diamond$*NoWallet*, and its implementation in the Ambient Calculus could simply be given by *open* Wallet.

A shopper is, initially, a person with a wallet (a *Looker*) who is later likely to become a *Buyer*. A buyer is a person who has pulled out the wallet, presumably to buy something. When a wallet has been pulled out, it becomes vulnerable to a nearby thief.

In the following derivation, we show that the interaction of a shopper with a thief (possibly in some larger context) may result in a *CrimeScene*, which is a situation in which the shopper has no wallet, and also there is no wallet to be found nearby.

$NoWallet \triangleq \neg(\text{Wallet}[\mathbf{T}] \mid \mathbf{T})$
$Looker \triangleq \text{Person}[\text{Wallet}[\mathbf{T}] \mid \mathbf{T}]$
$Buyer \triangleq \text{Person}[NoWallet] \mid \text{Wallet}[\mathbf{T}]$
$Shopper \triangleq Looker \wedge \Diamond Buyer$
$Thief \triangleq \text{Wallet}[\mathbf{T}] \triangleright \Diamond NoWallet$
$CrimeScene \triangleq \text{Person}[NoWallet] \mid NoWallet$

We begin with the system *Buyer* | *Thief*; using the rules ($\triangleright$ |) and (| $\vdash$) we obtain:

$Buyer \mid Thief$
$= \text{Person}[NoWallet] \mid \text{Wallet}[\mathbf{T}] \mid (\text{Wallet}[\mathbf{T}] \triangleright \Diamond NoWallet)$
$\vdash \text{Person}[NoWallet] \mid \Diamond NoWallet$

From the rules ($\Diamond$T) $\}$ $\mathcal{A}\vdash\Diamond\mathcal{A}$, (Id), and (| $\vdash$) we obtain, in general, $\mathcal{A}\mid(\Diamond\mathcal{B})\vdash(\Diamond\mathcal{A})\mid(\Diamond\mathcal{B})$. Then, by ($\Diamond$ |) $\}$ $(\Diamond\mathcal{A})\mid(\Diamond\mathcal{B})\vdash\Diamond(\mathcal{A}\mid\mathcal{B})$ and transitivity (derivable from (Cut)) we obtain $\mathcal{A}\mid(\Diamond\mathcal{B})\vdash\Diamond(\mathcal{A}\mid\mathcal{B})$. Using this fact in our example we obtain, by transitivity:

$Buyer \mid Thief \vdash \Diamond(\text{Person}[NoWallet] \mid NoWallet)$
$= \Diamond CrimeScene$

Using the rules ($\Diamond$ $\vdash$) $\mathcal{A}\vdash\mathcal{B}$ $\}$ $\Diamond\mathcal{A}\vdash\Diamond\mathcal{B}$, and ($\Diamond$ 4) $\}$ $\Diamond\Diamond\mathcal{A}\vdash\Diamond\mathcal{A}$, we derive:

$\Diamond(Buyer \mid Thief) \vdash \Diamond\Diamond CrimeScene$
$\Diamond\Diamond CrimeScene \vdash \Diamond CrimeScene$

As before, we can derive $(\Diamond\mathcal{A})\mid\mathcal{B}\vdash\Diamond(\mathcal{A}\mid\mathcal{B})$; therefore:

$(\Diamond Buyer) \mid Thief \vdash \Diamond(Buyer \mid Thief)$

and, by transitivity from above:

$(\Diamond Buyer) \mid Thief \vdash \Diamond CrimeScene$

then, by weakening (W-L):

$(Looker \mid Thief) \wedge ((\Diamond Buyer) \mid Thief) \vdash \Diamond CrimeScene$

Now let's consider the system $Shopper \mid Thief$. By the distribution of $\mid$ over $\wedge$ (($\mid \wedge$), from section 4.2.2) we have:

$Shopper \mid Thief = (Looker \wedge \Diamond Buyer) \mid Thief$

$\vdash (Looker \mid Thief) \wedge ((\Diamond Buyer) \mid Thief)$

and finally, by transitivity from above, we obtain:

$Shopper \mid Thief \vdash \Diamond CrimeScene$

5 A Decidable Sublogic

A model checker is an algorithm that determines the truth of an assertion $P \vDash \mathcal{A}$, given process P and formula $\mathcal{A}$ as input. We describe a model checker for the case where P is replication-free and $\mathcal{A}$ is $\triangleright$-free. The model checker depends on putting any replication-free process into a normal form, given by a finite product of prime processes:

Products, Primes, and Normal Forms

$\Pi_{i\in 1..k} P_i \triangleq P_1 \mid ... \mid P_k \mid \mathbf{0}$	product
$\pi ::= M[P] \parallel n.P \parallel in\ M.P \parallel out\ M.P \parallel open\ M.P \parallel (n).P \parallel \langle M\rangle$	prime process
$\Pi_{i\in 1..k} \pi_i$	normal form

The following recursive algorithm maps any replication-free process to a list of prime processes representing a normal form structurally congruent to the original process. We write lists of processes in the notation $[P_1, ..., P_k]$.

Normal Form for a Replication-Free Process

$Norm(\mathbf{0}) \triangleq []$
$Norm(P \mid P') \triangleq [\pi_1, ..., \pi_k, \pi'_1, ..., \pi'_{k'}]$
  if $Norm(P) = [\pi_1, ..., \pi_k]$ and $Norm(P') = [\pi'_1, ..., \pi'_{k'}]$
$Norm(M[P]) \triangleq [M[P]]$
$Norm(M.P) \triangleq [M.P]$ if $M \in \{n, in\ N, out\ N, open\ N\}$
$Norm(\varepsilon.P) \triangleq Norm(P)$
$Norm((M.N).P) \triangleq Norm(M.(N.P))$
$Norm((n).P) \triangleq [(n).P]$
$Norm(\langle M\rangle) \triangleq [\langle M\rangle]$

5-1 Lemma

If $Norm(P) = [\pi_1, ..., \pi_k]$ then $P \equiv \Pi_{i\in 1..k}\, \pi_i$.

□

To check the sometime and somewhere modalities, we depend on two routines *Reachable* and *SubLocations* that given a process P compute a representation of the sets of processes Q such that $P \longrightarrow^* Q$ and $P \downarrow^* Q$, respectively. We omit the straightforward definitions of these routines. Instead, we state their desired properties, which are proved using techniques developed previously [12].

5-2 Lemma

If $Reachable(P) = [P_1, ..., P_k]$ then for all $i\in 1..k$, $P \longrightarrow^* P_i$, and for all Q, if $P \longrightarrow^* Q$ then $Q \equiv P_i$ for some $i\in 1..k$.

If $SubLocations(P) = [P_1, ..., P_k]$ then for all $i\in 1..k$, $P \downarrow^* P_i$, and for all Q, if $P \downarrow^* Q$ then $Q \equiv P_i$ for some $i\in 1..k$.

□

Next, we define our model checking algorithm, and state its correctness property, Proposition 5-4, together with the main lemmas used in its proof.

Checking Whether Process P Satisfies Closed Formula $\mathcal{A}$

$Check(P, \mathbf{T}) \triangleq \mathbf{T}$
$Check(P, \neg\mathcal{A}) \triangleq \neg Check(P, \mathcal{A})$
$Check(P, \mathcal{A} \vee \mathcal{B}) \triangleq Check(P, \mathcal{A}) \vee Check(P, \mathcal{B})$
$Check(P, \mathbf{0}) \triangleq$ if $Norm(P) = []$ then $\mathbf{T}$ else $\mathbf{F}$
$Check(P, n[\mathcal{A}]) \triangleq$
  if $Norm(P) = [n[Q]]$ for some Q, then $Check(Q, \mathcal{A})$, else $\mathbf{F}$
$Check(P, \mathcal{A} \mid \mathcal{B}) \triangleq$
  let $Norm(P) = [\pi_1, ..., \pi_k]$
  in $\exists I,J.\ I\cup J=1..k \wedge I\cap J=\emptyset \wedge$
    $Check(\Pi_{i\in I}\, \pi_i, \mathcal{A}) \wedge Check(\Pi_{i\in J}\, \pi_i, \mathcal{B})$
$Check(P, \forall x.\mathcal{A}) \triangleq$
  let $\{m_1, ..., m_k\} = fn(P)\cup fn(\mathcal{A})$ and $m_0 \notin \{m_1, ..., m_k\}$
  in $\forall i\in 0..k.\ Check(P, \mathcal{A}\{x\leftarrow m_i\})$
$Check(P, \Diamond\mathcal{A}) \triangleq$
  let $[P_1, ..., P_k] = Reachable(P)$ in $\exists i\in 1..k.\ Check(P_i, \mathcal{A})$
$Check(P, \diamond\mathcal{A}) \triangleq$
  let $[P_1, ..., P_k] = SubLocations(P)$ in $\exists i\in 1..k.\ Check(P_i, \mathcal{A})$
$Check(P, \mathcal{A}@n) \triangleq Check(n[P], \mathcal{A})$

5-3 Lemmas

(1) For all replication-free processes P and Q, and all replication-free primes $\pi_1, ..., \pi_k$, $P \mid Q \equiv \Pi_{i\in 1..k}\, \pi_i$ if and only if there are sets I and J such that $I\cup J = 1..k$, $I\cap J = \emptyset$, $P \equiv \Pi_{i\in I}\, \pi_i$, and $Q \equiv \Pi_{i\in J}\, \pi_i$.

(2) For all replication-free processes P, and all replication-free primes $\pi_1, ..., \pi_k$, $n[P] \equiv \Pi_{i\in 1..k}\, \pi_i$ if and only if $k = 1$ and there is Q with $\pi_1 = n[Q]$ and $P \equiv Q$.

(3) For all replication-free processes P and $\triangleright$-free closed formulas $\forall x.\mathcal{A}$, if $\{m_1, ..., m_k\} = fn(P)\cup fn(\mathcal{A})$ and $m_0 \notin \{m_1, ..., m_k\}$, then: $P \vDash \forall x.\mathcal{A}$ if and only if $\forall i\in 0..k.\ P \vDash \mathcal{A}\{x\leftarrow m_i\}$.

□

5-4 Proposition

For all replication-free processes P and $\triangleright$-free closed formulas $\mathcal{A}$, $P \vDash \mathcal{A}$ if and only if $Check(P, \mathcal{A}) = \mathbf{T}$.

□

Since all the recursive calls are on subformulas of the original formula, the algorithm always terminates. When computing $Check(P, \mathcal{A} \mid \mathcal{B})$ with $Norm(P) = [\pi_1, ..., \pi_k]$ there are 2^k different subsets of $1..k$, and so 2^k different choices of the sets I and J. Therefore, in general the time complexity of $Check(P, \mathcal{A})$ is at least exponential in the size of P. (The practical performance of this algorithm can be greatly improved by special-casing and heuristics.)

Examples: define $an\ n \triangleq n[\mathbf{T}]^{\exists}$, and $p\ parents\ q \triangleq p[q[\mathbf{T}]^{\exists}]^{\exists}$, and let $P = a[p[out\ a.\ in\ b.\ \langle m\rangle]] \mid b[open\ p.\ (x).\ x[]]$, as in Section 2.2. The algorithm returns the following results on various example formulas:

$Check(P, an\ a) = \mathbf{T}$	$Check(P, \Diamond\diamond an\ m) = \mathbf{T}$
$Check(P, an\ b) = \mathbf{T}$	$Check(P, a\ parents\ p) = \mathbf{T}$
$Check(P, an\ p) = \mathbf{F}$	$Check(P, b\ parents\ p) = \mathbf{F}$
$Check(P, \diamond an\ p) = \mathbf{T}$	$Check(P, \Diamond b\ parents\ p) = \mathbf{T}$

In summary, Proposition 5-4 shows that the model checking problem for the sublogic without $\triangleright$ and the subcalculus without **!** is decidable. It is not clear in general how to extend this algorithm to include either **!** or $\triangleright$, because in principle an unbounded number of processes needs to be considered. For example, checking the truth of $P \vDash \mathbf{T} \triangleright \mathcal{A}$ in principle requires showing for all processes P' that $P \mid P' \vDash \mathcal{A}$. Similarly, checking the truth of $\mathbf{!}P \vDash \neg(\mathcal{A} \mid \mathbf{T})$ in principle requires showing that neither $\mathbf{!}P \vDash \mathcal{A}$ nor $P^k \vDash \mathcal{A}$ for all $k \geq 0$.

6 Connections with Other Logics

In this final section we compare our logic with well known substructural logics.

6.1 Relevant Logic

The shape of our definition of the satisfaction relation turns out to be very similar to Urquhart's semantics of relevant logic [19]. (Thanks to Peter O'Hearn and David Pym for pointing this out.) In particular $\mathcal{A} \mid \mathcal{B}$ is similar to *intensional conjunction*, and $\mathcal{A} \triangleright \mathcal{B}$ is similar to *relevant implication* in that semantics. The main difference with standard formulations of relevant logic is that we do not have contraction: this rule is not sound for process calculi, because $P \mid P \neq P$ under any reasonable equivalence.

Moreover, we use an equivalence, $\equiv$, instead of a Kripke-style partial order as in Urquhart's general case. If we were to adopt a partial order (perhaps some asymmetric form of structural congruence), then the classical fragment of our logic would have to be replaced by an intuitionistic fragment, in order to maintain the analogue of Proposition 3-1. This seems to be the deep reason why we can get by with classical implication.

6.2 Bunched Logic

Peter O'Hearn and David Pym study *bunched logics* [18], where sequents have two structural combinators, instead of the standard single "," combinator (usually meaning $\wedge$ or $\otimes$ on the left) found in most presentations of logic. Thus, sequents are *bunches* of formulas, instead of lists of formulas. Correspondingly, there are two implications that arise as the adjuncts of the two structural combinators.

The situation is very similar to our combinators $\mid$ and $\wedge$, which can combine to irreducible bunches of formulas in sequents, and to our two implications $\Rightarrow$ and $\triangleright$. However, we have a classical and a linear implication, while bunched logics have so far had an intuitionistic and a linear implication.

6.3 Linear Logic

We now relate a fragment of our logic to intuitionistic linear logic. Although the connections with some parts of linear logic are slightly degenerate, we can make them quite precise.

First note that, when considering $\mid$ as a structural connective, we must reject weakening, which entails $\mathcal{A} \vdash \mathbf{0}$, and contraction, which entails $\mathcal{A} \vdash \mathcal{A} \mid \mathcal{A}$: both are unsound in our process model. Therefore, we are at least somewhat close in spirit to linear logic. Our sequents are linear in the sense that we must have the same number of process components on the left and right of $\vdash$. In other words, space cannot be instantaneously created or destroyed. Consequently, the implication $\triangleright$ arising as an adjunct of $\mid$ is a linear implication: note that in the definition of $\mathcal{A} \triangleright \mathcal{B}$ the attacker that satisfies $\mathcal{A}$ is used exactly once in the system that satisfies $\mathcal{B}$.

Multiplicative intuitionistic linear logic (MILL) can be captured faithfully by identifying $\multimap_{\text{MILL}} = \triangleright$, $\otimes_{\text{MILL}} = \mid$, and $\mathbf{1}_{\text{MILL}} = \mathbf{0}$: the rules of MILL and the subset of our rules that involve only those connectives (plus a derivable cut rule for $\mid$ corresponding to the MILL cut rule) are interderivable. However, this precise match is obtained by paring down both linear logic and our logic. We can go further and draw a connection with full intuitionistic linear logic, both syntactically and semantically.

First, syntactically, intuitionistic linear logic (ILL) [13,16,8] can be embedded in our logic by the mapping:

$$
\begin{array}{lll@{\qquad\qquad}lll}
\mathcal{A} \oplus \mathcal{B} & \triangleq & \mathcal{A} \vee \mathcal{B} & \mathbf{1}_{\text{ILL}} & \triangleq & \mathbf{0} \\
\mathcal{A} \,\&\, \mathcal{B} & \triangleq & \mathcal{A} \wedge \mathcal{B} & \bot_{\text{ILL}} & \triangleq & \mathbf{F} \\
\mathcal{A} \otimes \mathcal{B} & \triangleq & \mathcal{A} \mid \mathcal{B} & \top_{\text{ILL}} & \triangleq & \mathbf{T} \\
\mathcal{A} \multimap \mathcal{B} & \triangleq & \mathcal{A} \triangleright \mathcal{B} & \mathbf{0}_{\text{ILL}} & \triangleq & \mathbf{F} \\
\mathbf{!}\mathcal{A} & \triangleq & \mathbf{0} \wedge (\mathbf{0} \Rightarrow \mathcal{A})^{\neg\mathbf{F}} & & &
\end{array}
$$

This mapping is such that the rules of ILL can be derived within our logic, so $\mathcal{A}_1, ..., \mathcal{A}_n \vdash_{\text{ILL}} \mathcal{B}$ implies $\mathcal{A}_1 \mid ... \mid \mathcal{A}_n \vdash \mathcal{B}$. In particular, we can derive the "strong" rules for $\mathbf{!}\mathcal{A}$ that correspond to an interpretation of **!** as a maximal fixpoint [13,16,8]:

$$
\begin{array}{ll}
(!\ \text{L1}) & \}_{\text{ILL}}\ \mathbf{!}\mathcal{A} \vdash_{\text{ILL}} \mathbf{1}_{\text{ILL}} \\
(!\ \text{L2}) & \}_{\text{ILL}}\ \mathbf{!}\mathcal{A} \vdash_{\text{ILL}} \mathcal{A} \\
(!\ \text{L3}) & \}_{\text{ILL}}\ \mathbf{!}\mathcal{A} \vdash_{\text{ILL}} \mathbf{!}\mathcal{A} \otimes \mathbf{!}\mathcal{A} \\
(!\ \text{R}) & \mathcal{B} \vdash_{\text{ILL}} \mathbf{1}_{\text{ILL}};\ \mathcal{B} \vdash_{\text{ILL}} \mathcal{A};\ \mathcal{B} \vdash_{\text{ILL}} \mathcal{B} \otimes \mathcal{B}\ \ \}_{\text{ILL}}\ \ \mathcal{B} \vdash_{\text{ILL}} \mathbf{!}\mathcal{A}
\end{array}
$$

We omit the proof of correctness of the embedding; this is not hard, but it requires gradual build-up and some experience with our logic.

The semantic connection is made through quantales [8]. We recall that a (commutative) quantale $\mathcal{Q}$ is a structure $<S{:}\text{Set},\ \leq{:}\ S^2 \to \text{Bool},\ \bigvee{:}\mathcal{P}(S) \to S,\ \otimes{:}S^2 \to S,\ 1{:}S>$ such that $\leq$ and $\bigvee$ form a complete join semilattice, $\otimes$ and 1 form a commutative monoid, and $p \otimes \bigvee Q = \bigvee\{p \otimes q \parallel q \in Q\}$ for all $p \in S$ and $Q \subseteq S$. It is folklore that quantales are sound and complete models of intuitionistic linear logic, according to the following interpretation $[\![\mathcal{A}]\!]_{\mathcal{Q}}$ (we omit the subscript when $\mathcal{Q}$ is unambiguous):

$$
\begin{array}{lll}
[\![\mathcal{A} \oplus \mathcal{B}]\!] & \triangleq & \bigvee\{[\![\mathcal{A}]\!], [\![\mathcal{B}]\!]\} \\
[\![\mathcal{A} \,\&\, \mathcal{B}]\!] & \triangleq & \bigvee\{C \parallel C \leq [\![\mathcal{A}]\!] \wedge C \leq [\![\mathcal{B}]\!]\} \\
[\![\mathcal{A} \otimes \mathcal{B}]\!] & \triangleq & [\![\mathcal{A}]\!] \otimes [\![\mathcal{B}]\!] \\
[\![\mathcal{A} \multimap \mathcal{B}]\!] & \triangleq & \bigvee\{C \parallel C \otimes [\![\mathcal{A}]\!] \leq [\![\mathcal{B}]\!]\} \\
[\![\mathbf{!}\mathcal{A}]\!] & \triangleq & \upsilon X.\ [\![\mathbf{1}_{\text{ILL}} \,\&\, \mathcal{A} \,\&\, (X \otimes X)]\!] \\
[\![\mathbf{1}_{\text{ILL}}]\!] & \triangleq & 1 \\
[\![\bot_{\text{ILL}}]\!] & \triangleq & \text{any element of } S \\
[\![\top_{\text{ILL}}]\!] & \triangleq & \bigvee S \\
[\![\mathbf{0}_{\text{ILL}}]\!] & \triangleq & \bigvee \varnothing
\end{array}
$$

where $\upsilon X.\ A\{X\} \ \triangleq\ \bigvee\{C \parallel C \leq A\{C\}\}$

The validity of ILL sequents and the soundness and completeness properties are stated as follows:

$$
\begin{array}{l}
\boldsymbol{vld}_{\text{ILL}}(\mathcal{A}_1, ..., \mathcal{A}_n \vdash_{\text{ILL}} \mathcal{B})_{\mathcal{Q}} \triangleq \\
\qquad [\![\mathcal{A}_1]\!]_{\mathcal{Q}} \otimes_{\mathcal{Q}} ... \otimes_{\mathcal{Q}} [\![\mathcal{A}_n]\!]_{\mathcal{Q}} \leq_{\mathcal{Q}} [\![\mathcal{B}]\!]_{\mathcal{Q}} \\
\mathcal{A}_1, ..., \mathcal{A}_n \vdash_{\text{ILL}} \mathcal{B} \ \Leftrightarrow \\
\qquad \text{for all quantales } \mathcal{Q},\ \boldsymbol{vld}_{\text{ILL}}(\mathcal{A}_1, ..., \mathcal{A}_n \vdash_{\text{ILL}} \mathcal{B})_{\mathcal{Q}}
\end{array}
$$

Now, sets of Ambient Calculus processes closed under structural congruence form a quantale. More precisely, the structure $\Theta \triangleq <\Phi, \subseteq, \bigcup, \otimes, 1>$ is a quantale, where, for $A,B \subseteq \Pi$, and for $A^{\equiv} \triangleq \{P \parallel \exists Q \in A.\ P \equiv Q\}$, we take $\Phi \triangleq \{A^{\equiv} \parallel A \subseteq \Pi\}$, $A \otimes B \triangleq \{P \mid Q \parallel P \in A \wedge Q \in B\}^{\equiv}$, and $1 \triangleq \{\mathbf{0}\}^{\equiv}$. Our logic is interpreted as follows: $[\![\mathcal{A}]\!] \triangleq \{P \in \Pi \parallel P \vDash \mathcal{A}\}$; note that, by Proposition 3-1, $[\![\mathcal{A}]\!] = [\![\mathcal{A}]\!]^{\equiv}$.

6-1 Proposition (Soundness of the ILL interpretation)

The syntactically defined ILL constants and operators correspond to their quantale definitions in Θ.

Proof

We detail the most interesting cases, for $\otimes$, $\multimap$, and $\mathbf{!}$.

Case for $\otimes$: $[\![\mathcal{A} \otimes \mathcal{B}]\!] = [\![\mathcal{A}]\!] \otimes [\![\mathcal{B}]\!]$.

$(P \in [\![\mathcal{A} \otimes \mathcal{B}]\!]) \Leftrightarrow (P \in [\![\mathcal{A} \mid \mathcal{B}]\!]) \Leftrightarrow (P \vDash \mathcal{A} \mid \mathcal{B}) \Leftrightarrow (\exists P', P'' {:}\Pi.\ P \equiv P' \mid P'' \wedge P' \vDash \mathcal{A} \wedge P'' \vDash \mathcal{B}) \Leftrightarrow (P \in \{P' \mid P'' \Vert P' \vDash \mathcal{A} \wedge P'' \vDash \mathcal{B}\}^{\equiv}) \Leftrightarrow (P \in \{P' \mid P'' \Vert P' \in [\![\mathcal{A}]\!] \wedge P'' \in [\![\mathcal{B}]\!]\}^{\equiv}) \Leftrightarrow (P \in [\![\mathcal{A}]\!] \otimes [\![\mathcal{B}]\!])$

Case for $\multimap$: $[\![\mathcal{A} \multimap \mathcal{B}]\!] = [\![\mathcal{A}]\!] \multimap [\![\mathcal{B}]\!]$.

Let $A = [\![\mathcal{A}]\!]$ and $B = [\![\mathcal{B}]\!]$. $(P \in [\![\mathcal{A}]\!] \multimap [\![\mathcal{B}]\!]) \Leftrightarrow (P \in A \multimap B) \Leftrightarrow (P \in \bigcup\{C \Vert C \otimes A \subseteq B\}) \Leftrightarrow (\exists C.\ P \in C \wedge C \otimes A \subseteq B) \Leftrightarrow (\exists C.\ P \in C \wedge \forall Q.\ (\exists Q', Q''.\ Q \equiv Q' \mid Q'' \wedge Q' \in C \wedge Q'' \in A) \Rightarrow Q \in B) \Leftrightarrow (\forall Q''.\ Q'' \in A \Rightarrow P \mid Q'' \in B)$. The last step is derived as follows:

1) Assume $\exists C.\ P \in C \wedge \forall Q.\ (\exists Q', Q''.\ Q \equiv Q' \mid Q'' \wedge Q' \in C \wedge Q'' \in A) \Rightarrow Q \in B$. Take any R and assume $R \in A$. Instantiate the assumption with $P \mid R$ for Q and take $Q'=P$ and $Q''=R$; we obtain $P \mid R \in B$.

2) Conversely, assume $\forall R.\ R \in A \Rightarrow P \mid R \in B$. Take $C=\{P\}^{\equiv}$, take any Q, and assume $(\exists Q', Q''.\ Q \equiv Q' \mid Q'' \wedge Q' \in \{P\}^{\equiv} \wedge Q'' \in A)$. Instantiating the assumption with Q'' for R, we obtain $P \mid Q'' \in B$. Now, $Q' \equiv P$ by assumption, hence $P \mid Q'' \equiv Q' \mid Q'' \equiv Q$. Since B is $\equiv$-closed, we obtain $Q \in B$.

Hence, $(P \in [\![\mathcal{A}]\!] \multimap [\![\mathcal{B}]\!]) \Leftrightarrow (\forall Q''.\ Q'' \in A \Rightarrow P \mid Q'' \in B) \Leftrightarrow (\forall Q''.\ Q'' \vDash \mathcal{A} \Rightarrow P \mid Q'' \vDash \mathcal{B}) \Leftrightarrow (P \vDash \mathcal{A} \triangleright \mathcal{B}) \Leftrightarrow (P \in [\![\mathcal{A} \triangleright \mathcal{B}]\!]) \Leftrightarrow (P \in [\![\mathcal{A} \multimap \mathcal{B}]\!])$.

Case for $\mathbf{!}$: $[\![\mathbf{!}\mathcal{A}]\!] = \mathbf{!}[\![\mathcal{A}]\!]$.

First we show that $\forall P.\ \mathbf{0} \vDash \mathcal{A} \Leftrightarrow P \vDash (\mathbf{0} \Rightarrow \mathcal{A})^{\neg\mathbf{F}}$.

Take any P; by definition of $\triangleright$, we have $P \vDash (\mathbf{0} \Rightarrow \mathcal{A})^{\neg\mathbf{F}} \Leftrightarrow (\forall Q.\ Q \vDash \mathbf{0} \Rightarrow \mathcal{A})$. Then, $(\forall Q.\ Q \vDash \mathbf{0} \Rightarrow \mathcal{A}) \Leftrightarrow (\forall Q.\ Q \vDash \mathbf{0} \Rightarrow Q \vDash \mathcal{A}) \Leftrightarrow \mathbf{0} \vDash \mathcal{A}$. The last step is by instantiation of Q with $\mathbf{0}$, in one direction, and by Proposition 3-1, in the other direction.

Then we compute: $(P \in [\![\mathbf{!}\mathcal{A}]\!]) \Leftrightarrow (P \in [\![\mathbf{0} \wedge (\mathbf{0} \Rightarrow \mathcal{A})^{\neg\mathbf{F}}]\!]) \Leftrightarrow (P \equiv \mathbf{0} \wedge P \vDash (\mathbf{0} \Rightarrow \mathcal{A})^{\neg\mathbf{F}}) \Leftrightarrow (P \equiv \mathbf{0} \wedge \mathbf{0} \vDash \mathcal{A})$.

Now, in a quantale $\mathbf{!}A = \upsilon X.\ 1\ \&\ A\ \&\ (X \otimes X)$, which in Θ means $\upsilon X.\ \{\mathbf{0}\}^{\equiv} \cap A \cap (X \mid X)$. If $\mathbf{0} \notin A$ then $\{\mathbf{0}\}^{\equiv} \cap A = \emptyset$, and $\mathbf{!}A = \emptyset$. If instead $\mathbf{0} \in A$, then $\{\mathbf{0}\}^{\equiv} \cap A = \{\mathbf{0}\}^{\equiv}$, and $\mathbf{!}A = \upsilon X.\ \{\mathbf{0}\}^{\equiv} \cap (X \mid X)$. We have that $\{\mathbf{0}\}^{\equiv}$ is a fixpoint of $\lambda X.\ \{\mathbf{0}\}^{\equiv} \cap (X \mid X)$; moreover, if $B = \{\mathbf{0}\}^{\equiv} \cap (B \mid B)$ then $B \subseteq \{\mathbf{0}\}^{\equiv}$, hence $\{\mathbf{0}\}^{\equiv}$ is the greatest fixpoint, and $\mathbf{!}A = \{\mathbf{0}\}^{\equiv}$. In conclusion: if $\mathbf{0} \notin A$ then $\mathbf{!}A = \emptyset$ else if $\mathbf{0} \in A$ then $\mathbf{!}A = \{\mathbf{0}\}^{\equiv}$ and, by contrapositive, if $\mathbf{!}A \neq \emptyset$ then $\mathbf{0} \in A$.

Hence $P \in \mathbf{!}[\![\mathcal{A}]\!] \Rightarrow \mathbf{!}[\![\mathcal{A}]\!] \neq \emptyset \Rightarrow \mathbf{0} \in [\![\mathcal{A}]\!] \Rightarrow \mathbf{!}[\![\mathcal{A}]\!] = \{\mathbf{0}\}^{\equiv} \Rightarrow P \in \{\mathbf{0}\}^{\equiv}$; that is $P \in \mathbf{!}[\![\mathcal{A}]\!] \Rightarrow P \equiv \mathbf{0} \wedge \mathbf{0} \vDash \mathcal{A}$. Conversely, if $P \equiv \mathbf{0} \wedge \mathbf{0} \vDash \mathcal{A}$, then $\mathbf{0} \in [\![\mathcal{A}]\!] \Rightarrow \mathbf{!}[\![\mathcal{A}]\!] = \{\mathbf{0}\}^{\equiv} \Rightarrow P \in \mathbf{!}[\![\mathcal{A}]\!]$.

In conclusion $P \in \mathbf{!}[\![\mathcal{A}]\!] \Leftrightarrow P \equiv \mathbf{0} \wedge \mathbf{0} \vDash \mathcal{A} \Leftrightarrow P \in [\![\mathbf{!}\mathcal{A}]\!]$.

□

Moreover, in our model the linear notion of validity matches our notion of validity:

6-2 Proposition

Let $\mathcal{A}_1, ..., \mathcal{A}_n, \mathcal{B}$ be formulas in ILL.

$\boldsymbol{vld}_{\mathrm{ILL}}(\mathcal{A}_1, ..., \mathcal{A}_n \vdash_{\mathrm{ILL}} \mathcal{B})_{\Theta} \Leftrightarrow \boldsymbol{vld}(\mathcal{A}_1 \mid ... \mid \mathcal{A}_n \vdash \mathcal{B})$

(For n=0 this means: $\boldsymbol{vld}_{\mathrm{ILL}}(\vdash_{\mathrm{ILL}} \mathcal{B})_{\Theta} \Leftrightarrow \boldsymbol{vld}(\mathbf{0} \vdash \mathcal{B})$.)

Proof

$(\forall P.\ P \vDash \mathcal{A}_1 \mid ... \mid \mathcal{A}_n \Rightarrow \mathcal{B}) \Leftrightarrow (\forall P.\ P \vDash \mathcal{A}_1 \mid ... \mid \mathcal{A}_n \Rightarrow P \vDash \mathcal{B}) \Leftrightarrow (\forall P.\ P \in [\![\mathcal{A}_1 \mid ... \mid \mathcal{A}_n]\!] \Rightarrow P \in [\![\mathcal{B}]\!]) \Leftrightarrow [\![\mathcal{A}_1 \mid ... \mid \mathcal{A}_n]\!] \subseteq [\![\mathcal{B}]\!] \Leftrightarrow [\![\mathcal{A}_1]\!] \otimes ... \otimes [\![\mathcal{A}_n]\!] \subseteq [\![\mathcal{B}]\!]$.

The last step is as in the $\otimes$ case of Proposition 6-1.

□

The discrepancies with ILL are as follows. We identify $\bot_{\mathrm{ILL}}$ and $\mathbf{0}_{\mathrm{ILL}}$ (as $\mathbf{F}$); therefore, $\mathcal{A}^{\bot}$ acquires special properties. The additives $\oplus$ and & distribute over each other (both semantically and as a derived rule). The semantic interpretation of $\mathbf{!}\mathcal{A}$ is rather degenerate; in particular, $\mathbf{!}\mathcal{A} \multimap \mathcal{B}$ does not seem to have an interesting meaning.

Conclusions and Further Work

We have introduced an expressive logic that can describe properties of spatial configurations and of mobile computation, including security properties. Although some attack scenarios can already be described, many interesting security properties require the use of name restriction (which is already present in our full Ambient Calculus): we intend to study extensions of our logic in that direction. We also intend to study recursive modal formulas. Finally, we should consider issues of logical completeness: these have not been looked at because our focus has been on studying properties of the model. The only sense in which we feel we have a "large enough" set of rules is that we can logically derive the rules of intuitionistic linear logic.

We have previously developed type systems for mobility; now we have a model-checking algorithm for a decidable sublogic, and a more complete logic of mobility. These can be seen as three progressive stages in the screening of mobile code, corresponding to bytecode verification by type checking, by model checking, and by proof checking (as in proof-carrying code). In all these cases, it is possible to express and verify properties of mobile code that allow the code to move around after verification, safely removing the constraints of rigid sandboxing policies.

Acknowledgments

Giorgio Ghelli participated in the initial discussions leading to this logic. Barney Hilken and Peter O'Hearn both pointed out that '$\triangleright$' is adjunct to '|'. Glynn Winskel directed us to the connection with intuitionistic linear logic. Peter O'Hearn and David Pym pointed us to literature on relevant logic. Gordon Plotkin suggested the new axioms for structural congruence. Useful comments were made by Martín Abadi and Todd Knoblock.

References

[1] Abadi, M., Plotkin, G.D.: **A Logical View of Composition**. TCS **114**(1), 3-30, 1993.

[2] Amadio, R.M., Prasad, S.: **Localities and failures**. FST & TCS '94, LNCS 880, 205-216, Springer, 1994.

[3] Cardelli, L., Gordon, A.D.: **Mobile Ambients**. FoSSaCS'98, LNCS 1378, 140-155, Springer, 1998.

[4] Cardelli, L., Gordon, A.D.: **Types for Mobile Ambients.** POPL'99, 79-92, 1999.

[5] Cardelli, L., Ghelli, G., Gordon, A.D.: **Mobility Types for Mobile Ambients.** ICALP'99. LNCS 1644, 230-239, Springer, 1999.

[6] Dam, M.: **Relevance Logic and Concurrent Composition**. LICS'88, 178-185, 1988.

[7] Došen, K.: **A Historical Introduction to Substructural Logics**. In: Schroeder-Heistler, P., Došen, K. (eds.): Substructural Logics. Studies in Logic and Computation 2, 1-30, Clarendon Press, 1994.

[8] Engberg, U.H., Winskel, G.: **Linear Logic on Petri Nets**. BRICS Report RS-94-3, 1994.

[9] Engelfriet, J.: **A Multiset Semantics for the π-calculus with Replication**. TCS **153**, 65-94, 1996.

[10] Fournet, C., Gonthier, G.: **A Calculus of Mobile Agents**. CONCUR'96, LNCS 1119, Springer, 1999.

[11] Gentzen, G.: **Untersuchungen über das logische Schließen**. Mathematische Zeitschrift 39, 176-210, 405-431, 1935. English translation in: The collected papers of Gerhard Gentzen. M.E.Szabo (ed.), 132-213, North Holland, 1969.

[12] Gordon, A.D., Cardelli, L.: **Equational Properties of Mobile Ambients**. FoSSaCS'99, LNCS 1578, 212-226, Springer, 1996.

[13] Girard, J.-Y., Lafont, Y.: **Linear Logic and Lazy Computation**. TAPSOFT 87, LNCS 250 vol 2, 53-66, Springer, 1987.

[14] Girard, J.-Y., Lafont, Y., Taylor, P.: **Proofs and Types**. Cambridge University Press, 1989.

[15] Hennessy, H., Milner, R.: **Algebraic Laws for Nondeterminism and Concurrency**. JACM, **32**(1)137-161, 1985.

[16] Lafont, Y.: **The Linear Abstract Machine**. TCS (**59**)157-180, 1988.

[17] Milner, R.: **Flowgraphs and Flow Algebras**. JACM **26**(4), 1979.

[18] O'Hearn, P.W., Pym, D.: **The Logic of Bunched Implications**. Bulletin of Symbolic Logic. To appear, 1999.

[19] Urquhart, A.: **Semantics for Relevant Logics.** Journal of Symbolic Logic **37**(1)159-169, 1972.

[20] Vitek, J., Castagna, G.: **Seal: A Framework for Secure Mobile Computations**. Internet Programming Languages, LNCS 1686, 47-77, Springer, 1999.

Appendix: Rules of the Ambient Logic

This appendix collects information already presented in the paper.

Sequents:	$\mathcal{A} \vdash \mathcal{B}$	
Rules:	$\mathcal{A}_1 \vdash \mathcal{B}_1; \ldots; \mathcal{A}_n \vdash \mathcal{B}_n \} \mathcal{A} \vdash \mathcal{B}$	$(n \geq 0)$

Abbreviations: $\dashv\vdash$ means $\vdash$ in both directions; $\{\}$ means $\}$ in both directions.

Propositional Rules

(A-L)	$\mathcal{A} \wedge (\mathcal{C} \wedge \mathcal{D}) \vdash \mathcal{B} \;\{\}\; (\mathcal{A} \wedge \mathcal{C}) \wedge \mathcal{D} \vdash \mathcal{B}$
(A-R)	$\mathcal{A} \vdash (\mathcal{C} \vee \mathcal{D}) \vee \mathcal{B} \;\{\}\; \mathcal{A} \vdash \mathcal{C} \vee (\mathcal{D} \vee \mathcal{B})$
(X-L)	$\mathcal{A} \wedge \mathcal{C} \vdash \mathcal{B} \;\}\; \mathcal{C} \wedge \mathcal{A} \vdash \mathcal{B}$
(X-R)	$\mathcal{A} \vdash \mathcal{C} \vee \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B} \vee \mathcal{C}$
(C-L)	$\mathcal{A} \wedge \mathcal{A} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}$
(C-R)	$\mathcal{A} \vdash \mathcal{B} \vee \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}$
(W-L)	$\mathcal{A} \vdash \mathcal{B} \;\}\; \mathcal{A} \wedge \mathcal{C} \vdash \mathcal{B}$
(W-R)	$\mathcal{A} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{C} \vee \mathcal{B}$
(Id)	$\}\; \mathcal{A} \vdash \mathcal{A}$
(Cut)	$\mathcal{A} \vdash \mathcal{C} \vee \mathcal{B};\; \mathcal{A}' \wedge \mathcal{C} \vdash \mathcal{B}' \;\}\; \mathcal{A} \wedge \mathcal{A}' \vdash \mathcal{B} \vee \mathcal{B}'$
(**T**)	$\mathcal{A} \wedge \mathbf{T} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}$
(**F**)	$\mathcal{A} \vdash \mathbf{F} \vee \mathcal{B} \;\}\; \mathcal{A} \vdash \mathcal{B}$
(¬-L)	$\mathcal{A} \vdash \mathcal{C} \vee \mathcal{B} \;\}\; \mathcal{A} \wedge \neg\mathcal{C} \vdash \mathcal{B}$
(¬-R)	$\mathcal{A} \wedge \mathcal{C} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \neg\mathcal{C} \vee \mathcal{B}$

Quantifier Rules

(∀-L)	$\mathcal{A}\{x \leftarrow \eta\} \vdash \mathcal{B} \;\}\; \forall x.\mathcal{A} \vdash \mathcal{B}$	(η is a name or a variable)
(∀-R)	$\mathcal{A} \vdash \mathcal{B} \;\}\; \mathcal{A} \vdash \forall x.\mathcal{B}$	where $x \notin fv(\mathcal{A})$

Composition Rules

(| **0**)	$\}\; \mathcal{A} \mid \mathbf{0} \dashv\vdash \mathcal{A}$
(| ¬**0**)	$\}\; \mathcal{A} \mid \neg\mathbf{0} \vdash \neg\mathbf{0}$
(A |)	$\}\; \mathcal{A} \mid (\mathcal{B} \mid \mathcal{C}) \dashv\vdash (\mathcal{A} \mid \mathcal{B}) \mid \mathcal{C}$
(X |)	$\}\; \mathcal{A} \mid \mathcal{B} \vdash \mathcal{B} \mid \mathcal{A}$
(| ⊢)	$\mathcal{A}' \vdash \mathcal{B}';\; \mathcal{A}'' \vdash \mathcal{B}'' \;\}\; \mathcal{A}' \mid \mathcal{A}'' \vdash \mathcal{B}' \mid \mathcal{B}''$
(| ∨)	$\}\; (\mathcal{A} \vee \mathcal{B}) \mid \mathcal{C} \vdash \mathcal{A} \mid \mathcal{C} \vee \mathcal{B} \mid \mathcal{C}$
(| ‖)	$\}\; \mathcal{A}' \mid \mathcal{A}'' \vdash (\mathcal{A}' \mid \mathcal{B}'') \vee (\mathcal{B}' \mid \mathcal{A}'') \vee (\neg\mathcal{B}' \mid \neg\mathcal{B}'')$
(| ▷)	$\mathcal{A} \mid \mathcal{C} \vdash \mathcal{B} \;\{\}\; \mathcal{A} \vdash \mathcal{C} \triangleright \mathcal{B}$
(▷**F** ¬)	$\}\; \mathcal{A}^{\mathbf{F}} \vdash \mathcal{A}^{\neg}$
(¬ ▷**F**)	$\}\; \mathcal{A}^{\mathbf{F}\neg} \vdash \mathcal{A}^{\mathbf{FF}}$

Location Rules

(n[] ¬**0**)	$\}\; n[\mathcal{A}] \vdash \neg\mathbf{0}$
(n[] ¬ |)	$\}\; n[\mathcal{A}] \vdash \neg(\neg\mathbf{0} \mid \neg\mathbf{0})$
(n[] ⊢)	$\mathcal{A} \vdash \mathcal{B} \;\{\}\; n[\mathcal{A}] \vdash n[\mathcal{B}]$
(n[] ∧)	$\}\; n[\mathcal{A}] \wedge n[\mathcal{B}] \vdash n[\mathcal{A} \wedge \mathcal{B}]$
(n[] ∨)	$\}\; n[\mathcal{A} \vee \mathcal{B}] \vdash n[\mathcal{A}] \vee n[\mathcal{B}]$
(n[] @)	$n[\mathcal{A}] \vdash \mathcal{B} \;\{\}\; \mathcal{A} \vdash \mathcal{B}@n$
(¬ @)	$\}\; \mathcal{A}@n \dashv\vdash \neg((\neg\mathcal{A})@n)$

Time and Space Modality Rules

(◇)	$\}\; \Diamond\mathcal{A} \dashv\vdash \neg\Box\neg\mathcal{A}$
(□ K)	$\}\; \Box(\mathcal{A} \Rightarrow \mathcal{B}) \vdash \Box\mathcal{A} \Rightarrow \Box\mathcal{B}$
(□ T)	$\}\; \Box\mathcal{A} \vdash \mathcal{A}$
(□ 4)	$\}\; \Box\mathcal{A} \vdash \Box\Box\mathcal{A}$
(□ **T**)	$\}\; \mathbf{T} \vdash \Box\mathbf{T}$
(□ ⊢)	$\mathcal{A} \vdash \mathcal{B} \;\}\; \Box\mathcal{A} \vdash \Box\mathcal{B}$
(✧)	$\}\; \text{✧}\mathcal{A} \dashv\vdash \neg\text{¤}\neg\mathcal{A}$
(¤ K)	$\}\; \text{¤}(\mathcal{A} \Rightarrow \mathcal{B}) \vdash \text{¤}\mathcal{A} \Rightarrow \text{¤}\mathcal{B}$
(¤ T)	$\}\; \text{¤}\mathcal{A} \vdash \mathcal{A}$
(¤ 4)	$\}\; \text{¤}\mathcal{A} \vdash \text{¤}\text{¤}\mathcal{A}$
(¤ **T**)	$\}\; \mathbf{T} \vdash \text{¤}\mathbf{T}$
(¤ ⊢)	$\mathcal{A} \vdash \mathcal{B} \;\}\; \text{¤}\mathcal{A} \vdash \text{¤}\mathcal{B}$
(◇ n[])	$\}\; n[\Diamond\mathcal{A}] \vdash \Diamond n[\mathcal{A}]$
(◇ |)	$\}\; \Diamond\mathcal{A} \mid \Diamond\mathcal{B} \vdash \Diamond(\mathcal{A} \mid \mathcal{B})$
(✧ n[])	$\}\; n[\text{✧}\mathcal{A}] \vdash \text{✧}\mathcal{A}$
(✧ |)	$\}\; \text{✧}\mathcal{A} \mid \mathcal{B} \vdash \text{✧}(\mathcal{A} \mid \mathbf{T})$
(✧◇)	$\}\; \text{✧}\Diamond\mathcal{A} \vdash \Diamond\text{✧}\mathcal{A}$