A Complete Temporal and Spatial Logic for Distributed Systems*

Dirk Pattinson[1] and Bernhard Reus[2]

[1] LMU München, Institut für Informatik, 80538 München
[2] University of Sussex, Informatics, Brighton BN1 9QH

Abstract. In this paper, we introduce a spatial and temporal logic for reasoning about distributed computation. The logic is a combination of an extension of hybrid logic, that allows us to reason about the spatial structure of a computation, and linear temporal logic, which accounts for the temporal aspects. On the pragmatic side, we show the wide applicability of this logic by means of many examples. Our main technical contribution is completeness of the logic both with respect to spatial/temporal structures and a class of spatial transition systems.

1 Introduction

With the advent of the Internet, mobility and spatial distribution of information systems have established themselves as a new computational paradigm.

Distributed and mobile systems, however, require new specification and verification methodologies. Program logics have to account for space *and* time in a single, unified framework, stating where and when certain computations happen. A further challenge consists of the fact that these systems run on heterogeneous platforms using various different programming languages.

The formal modelling of distributed and mobile systems has traditionally been the domain of process calculi. Several approaches can be found in the literature, for example the π-calculus [22], the ambient calculus [9], and Klaim [12]. In all of these approaches, distributed processes are represented as terms in the language of the underlying calculus. For each of these calculi, corresponding formal logics have been proposed to reason about the behaviour of distributed computation. For example, see [23,7,4,5] for the π-calculus, [9] for the Ambient-calculus, and [24] for Klaim, to name but a few. From a practical perspective, it seems unrealistic to assume that all entities participating in a distributed (or mobile) system can be specified in a single *syntactic* framework: by its very nature, distributed computation integrates various different platforms, operating systems, and programming languages.

A single *semantic* framework is, however, desirable as it supports the analysis and comparison of different logics and calculi. This paper bridges the gap between theory and practice and introduces syntax-independent models of distributed and mobile systems together with an associated logic, that allows to reason about the behaviour of

* This work was partially sponsored by the DAAD and the British Council in the ARC project 1205 "Temporal and Spatial Logic for Mobile Systems".

B. Gramlich (Ed.): FroCoS 2005, LNAI 3717, pp. 122–137, 2005.

such systems. On a semantical level, we consider *spatial transition systems*, which encapsulate the behaviour of individual components without the need of expressing the behaviour of the component in a particular syntactic formalism. The properties of the systems under consideration are expressed using *linear spatial temporal logic* ($\mathcal{LSTL}$), a new logic that we introduce and study in this paper. It arises as a combination of two logics that reflect the two aspects of distributed computation. The first is an extension of both hybrid logic [1,3] and combinatory dynamic logic [25]. This logic, which we call $\mathcal{HL}^*$, is used to reason about the spatial (e.g. network) structure present at one particular point in time. The second is linear temporal logic [20,19] to capture the temporal aspect. This linear spatial temporal logic is independent of any concrete programming or process language. By means of examples, we show that our models and our logic capture many situations that naturally arise in distributed computation.

Our main technical contribution is the completeness of our logic, both with respect to spatio-temporal structures (which are introduced later in the paper) and a class of spatial transition systems. In more detail, we first introduce the spatial component $\mathcal{HL}^*$ of our logic, which can be viewed either as extension of hybrid logic [3,2] with iteration (Section 2) or of combinatory dynamic logic [25] with satisfaction operators. We show that the resulting logic is weakly complete with respect to named models, that is, Kripke models where every location can be referenced by a (not necessarily unique) name. We then use local formulas, a subset of hybrid formulas that only describe properties of one specific node of the distributed structure, in place of propositional atoms in a linear temporal logic. As names provide the only handle to distinguish different nodes of the system, we have to insist that names do not change over time, that is, we consider names as *physical entities* rather than logical ones. Consequently, we have to extend the technique of [14] to account for this interference between the temporal and spatial dimension of the logic. This is achieved by considering sequences with *consistent naming* as models for the combined logic, which is reflected by an additional axiom. The second main result is the completeness of linear spatial temporal logic (Section 3) w.r.t. spatio-temporal structures. This completeness result is then extended to spatial transition systems (Section 4), which can be thought of as machine models for distributed computation. In a nutshell, we obtain a new and complete logical formalism, that is capable of reasoning about distributed computation and applicable to many situations that naturally arise in distributed computation.

Related Work. We have already mentioned the work on spatial logics interpreted over process calculi, notably the π-calculus and the ambient calculus [7,4,5], where the completeness of the logic is in general neglected; however [18] proves a Hennessy-Milner property. In [23,24], modal logics with primitive modal operators for process communication are proposed, but these are also tailored towards their specific process calculi. Finally, spatial logics that are structurally similar to ours have been proposed in the context of semi-structured data, e.g. Ghelli et al.'s work on query languages for XML documents [7]. Completeness is not addressed there. An intuitionistic hybrid logic is investigated in [10] including a completeness result, but w.r.t. Kripke structures that define intuitionistic models and places having no structure at all. A temporal and spatial logic is also used in [21,26] on the basis of a less flexible model of tree sequences. There is no completeness result so far.

2 Spatial Reasoning with Hybrid Logic

This section introduces the purely spatial part of our logic in isolation. To capture the whereabouts of a computation, two ingredients are essential: names for locations where computation takes place and the topological structure that connects these locations. We use a combination of hybrid logic [1] and combinatory dynamic logic [25] that reflects precisely these criteria. Our logic is an extension of modal logic, with a name attached to each world; this feature is present both in hybrid logic and in combinatory dynamic logic. This basic setup is extended with satisfaction operators (borrowed from hybrid logic), that allow us to assert that a formula holds at a specific point of the model. Combinatory dynamic logic contributes a modality for transitive closure, which provides the linguistic means to reason about reachable nodes in a model.

While modalities for transitive closure (i.e. the $*$ of dynamic logic) is needed to have enough expressive power in the language, satisfaction operators are crucial when it comes to combining spatial and temporal aspects. A satisfaction operator $@_i$ shifts the evaluation context the node of the model that has name i. As a consequence, satisfaction operators give rise to formulas $@_i\phi$ that that are either true or false at every node of the model.

A model of our logic is a Kripke Model, where additionally every name is assigned to a unique node. In view of our intended application, we view the worlds of the model as the locations where computation happens and call them *places*. Following the hybrid tradition, place names are referred to as *nominals*. If two places p_1 and p_2 of the model are related, then we interpret this as "from p_1 one can see p_2", or "p_1 has a network connection to p_2", depending on the particular context. In particular, as we require that every node has a name, and names are drawn from a countably infinite set of nominals, all of our models will have an at most countable carrier.

We now introduce the syntax and semantics of our extension $\mathcal{HL}^*$ of hybrid logic.

Definition 1 (Syntax of $\mathcal{HL}^*$). *Suppose that* A *is a set of atomic propositions and* Nom *is a set of nominals. The language of the logic* $\mathcal{HL}^*(\mathsf{A}, \mathsf{Nom})$ *is defined to be the least set of formulas according to the grammar*

$$\phi, \psi \in \mathcal{HL}^* ::= a \mid i \mid \,\diamondminus\phi \mid \,\diamondminus^*\phi \mid \phi \wedge \psi \mid \neg\phi \mid @_i\phi$$

where $a \in \mathsf{A}$ *ranges over the atomic propositions and and* $i \in \mathsf{Nom}$ *is a nominal. We use standard abbreviations for the propositional connectives* $\vee$, $\rightarrow$ *and put* $\boxminus = \neg \diamondminus \neg$, $\boxminus^* = \neg \boxminus^* \neg$. *We call a formula* $\phi \in \mathcal{HL}^*$ local, *if* $\phi = @_i\psi$ *for some* $i \in \mathsf{Nom}$ *and* $\psi \in \mathcal{HL}^*$*; the set of local formulas is denoted by* $L(\mathcal{HL}^*)$.

As it is common in Hybrid Logics, proposition $@_i\phi$ represents a local property, i.e. the fact that ϕ holds at at the unique place with name i. Moreover, $\diamondminus\phi$ means that ϕ holds at some place directly reachable from here, whereas $\diamondminus^*\phi$ means that ϕ holds somewhere reachable from here.

Our notion of model is standard; for notational convenience, we distinguish between the valuation of propositional variables and that of nominals. The semantics of $\mathcal{HL}^*$ is as follows:

Definition 2 (Semantics of $\mathcal{HL}^*$). *A* named hybrid model *is a tuple* $(P, \rightarrow, V, N)$ *where* P *is a set of* places, $\rightarrow \subseteq P \times P$ *is an adjacency relation,* $V : \mathsf{A} \rightarrow \mathcal{P}(P)$ *and* $N : \mathsf{Nom} \rightarrow P$ *are a valuation of propositional variables and nominals, respectively, with* N *a surjection.*

Given a named model $S = (P, \rightarrow, V, N)$*, satisfaction at a point* $p \in P$ *is given inductively by*

$$
\begin{aligned}
&(S, p) \models a \textit{ iff } p \in V(a)\\
&(S, p) \models i \textit{ iff } p = N(i)\\
&(S, p) \models \Diamond\phi \textit{ iff } \exists p'.\, p \rightarrow p' \wedge (S, p') \models \phi\\
&(S, p) \models \Diamond^*\phi \textit{ iff } \exists n \in \mathbb{N}.\, (S, p) \models \Diamond^n\phi\\
&(S, p) \models @_i\phi \textit{ iff } (S, N(i)) \models \phi
\end{aligned}
$$

where the semantics of propositional connectives is as usual. We write $S \models \phi$ *iff* $(S, p) \models \phi$ *for all* $p \in P$ *and* $\mathcal{HL}^* \models \phi$ *if* $S \models \phi$ *for all named models* S*. If there is danger of confusion, we make the logic explicit in the satisfaction relation and write* $(S, p) \models_{\mathcal{HL}^*} \phi$ *to say that* S *is a named hybrid model and* $\phi \in \mathcal{HL}^*$ *and similarly for* $S \models_{\mathcal{HL}^*} \phi$*.*

With the intuition that the places $p \in P$ of the Kripke frame $(P, \rightarrow)$ represent network nodes and the transition relation $p \rightarrow p'$ represents the possibility of transferring data from p to p', we can formulate assertions on the network topology:

Example 1.
1. The fact that node j is reachable from everywhere is described by the formula $\Diamond^* j$.
2. The fact that network node i is transitively connected to node j is captured in the formula $@_i(\Diamond^* j)$. Note the use of the satisfaction operator $@_i$ to shift the evaluation of the formula $\Diamond^* j$ to the node with name i.
3. If every node of a connected component of a Kripke model is connected to every other node of this component, the model will satisfy the formula $\Diamond^* i \rightarrow \Diamond i$.
4. Finally, we can force connections to be bidirectional by means of the formula $@_i(\Diamond^* j) \rightarrow @_j(\Diamond^* i)$.

Note that only the formula in 2 is *local*.

Our key concern in this section is to analyse the relationship between syntax and semantics of $\mathcal{HL}^*$, and our main result is completeness of the axiom system that we introduce now.

2.1 The Axioms of $\mathcal{HL}^*$

Note that we cannot expect $\mathcal{HL}^*$ to be strongly complete w.r.t. named models. For example, consider the set of formulas $\{\neg i \mid i \in \mathsf{Nom}\}$. This set is consistent, as all its finite subsets are, but not satisfiable in a named model with name set Nom. We therefore have to content ourselves with weak completeness of $\mathcal{HL}^*$, stating that validity of $\phi \in \mathcal{HL}^*$ in all models implies derivability of ϕ. The deducibility predicate $\vdash \subseteq \mathcal{HL}^*$, is given by the following axioms and rules.

(taut) all propositional tautologies

$(\mathrm{K}_{\boxminus})\quad \boxminus(\phi \to \psi) \to (\boxminus\phi \to \boxminus\psi)$

$(\mathrm{K}_@)\quad @_i(\phi \to \psi) \to (@_i\phi \to @_i\psi)$

$(\text{self-dual})\quad @_i\phi \leftrightarrow \neg @_i \neg\phi$

$(\text{intro})\quad i \wedge \phi \to @_i\phi$

$(\text{ref})\quad @_i i$

$(\text{sym})\quad @_i j \leftrightarrow @_j i$

$(\text{nom})\quad @_i j \wedge @_j\phi \to @_i\phi$

$(\text{agree})\quad @_j @_i\phi \leftrightarrow @_i\phi$

$(\text{back})\quad \Diamond @_i p \to @_i p$

$(\text{iter})\quad \boxminus^* \phi \to \phi \wedge \boxminus \boxminus^* \phi$

$(\text{ind})\quad (\phi \to \boxminus\phi) \wedge \phi \to \boxminus^*\phi$

The proof rules of $\mathcal{HL}^*$ are summarised as follows.

$$(\text{mp})\ \frac{\phi \to \psi \quad \phi}{\psi} \qquad (\text{gen})\ \frac{\phi}{\boxminus\phi} \qquad (\text{gen}_@)\ \frac{\phi}{@_i\phi} \qquad (\text{subst})\ \frac{\phi}{\phi[\theta/x]}(x \in \mathsf{Nom} \cup \mathsf{A})$$

$$(\text{name})\ \frac{j \to \phi}{\phi}(j \notin \text{nom}(\phi)) \qquad (\text{paste})\ \frac{(@_i \Diamond j)\ \wedge\ (@_j\phi)\ \to \psi}{@_i \Diamond\phi \to \psi}\quad (j \notin \text{nom}(\phi, \psi))$$

where $\text{nom}(\phi)$ (resp. $\text{nom}(\phi, \psi)$) denotes the set of nominals occurring in the formula ϕ (resp. in ϕ or ψ) and in the substitution rule it is silently understood to be type correct, i.e. formulas will be substituted for atomic propositions and nominals for nominals only.

Definition 3. *If $\Phi \subseteq \mathcal{HL}^*$ is a set of formulas of $\mathcal{HL}^*$, then $\phi \in \mathcal{HL}^*$ is* derivable *from Φ, if ϕ is contained in the least set of formulas that contains Φ and the above axioms and is closed under the proof rules of $\mathcal{HL}^*$. This is denoted by $\Phi \vdash_{\mathcal{HL}^*} \phi$. We write $\mathcal{HL}^* \vdash \phi$, if ϕ is a theorem of $\mathcal{HL}^*$, i.e. $\emptyset \vdash_{\mathcal{HL}^*} \phi$.*

It is straightforward to check the following proposition.

Proposition 1. *$\mathcal{HL}^*$ is sound, that is, if $\mathcal{HL}^* \vdash \phi$, then $\mathcal{HL}^* \models \phi$ for all $\phi \in \mathcal{HL}^*$.*

2.2 Completeness of $\mathcal{HL}^*$

We now establish completeness of $\mathcal{HL}^*$. The proof follows a standard argument using a canonical model, existence lemma, and truth lemma. We just elaborate on those issues that are specific to our logic. We begin with the construction of our model.

Definition 4 (see also [1]). *Suppose $\Phi \subseteq \mathcal{HL}^*$ is maximally consistent. Φ is* named, *if $i \in \Phi$ for some nominal $i \in \mathsf{Nom}$, and Φ is* pasted, *if $@_i \Diamond\phi \in \Phi$ implies that for some $j \in \mathsf{Nom}$, $@_i \Diamond j \wedge @_j\phi \in \Phi$.*

If Φ is a named and pasted maximally $\mathcal{HL}^*$-consistent set, or a named and pasted $\mathcal{HL}^*$-MCS for short, then the model induced by *Φ is given by $M_\Phi = (P, \to_c, V, N)$, where*

- *$P = \{\Delta_i \mid i \in \mathsf{Nom}\}$ with $\Delta_i = \{\phi \mid @_i\phi \in \Phi\}$;*
- *$\to_c$ is the canonical relation defined by $u \to_c u'$ iff $\{\Diamond\phi \mid \phi \in u'\} \subseteq u$;*
- *$V(a) = \{p \in P \mid a \in p\}$ is the canonical valuation of propositional variables,*
- *$N(i) =$ the unique $p \in P$ with $i \in P$.*

The following lemma justifies the above definition.

Lemma 1. *Suppose $M_\Phi = (P, \rightarrow_c, V, N)$ is the model induced by a named and pasted $\mathcal{HL}^*$-MCS Φ. Then p is named for every $p \in P$, and moreover $i \in p \cap q$ implies $p = q$ for all $p, q \in P$ and all $i \in \mathsf{Nom}$.*

Our desire for a named model dictates that we only use named MCS-s, and the condition that the MCS-s be pasted ensures the validity of an existence lemma; see [1, Section 7.3] for more on this issue. The following is an adaptation of the classical Lindenbaum lemma guaranteeing the existence of named and pasted MCS's.

Lemma 2. *Suppose $\phi \in \mathcal{HL}^*$ is consistent. Then there exists a named and pasted $\mathcal{HL}^*$-MCS containing ϕ and $@_i\neg j$ for all $i \neq j \in M$, for some countable subset $M \subseteq \mathsf{Nom}$ with $M \cap \mathrm{nom}(\phi) = \emptyset$.*

Consequently, a model induced by a MCS of the sort described in Lemma 2 is named and countable. Note that we extend a single formula to a maximally consistent set. This allows us to avoid having to enrich the language with new nominals (cf. [1, Section 7.3]). The proof for the existence lemma now works as for the basic hybrid case, and we move straight to the truth lemma.

Lemma 3 (Truth Lemma). *Suppose $M_\Phi = (P, \rightarrow_c, V, N)$ is the model induced by a named and pasted $\mathcal{HL}^*$-MCS Φ. Then, for all $\phi \in \mathcal{HL}^*$ and all $p \in P$, we have $(M_\Phi, p) \models \phi$ iff $\phi \in p$.*

Our completeness result follows from Lemma 3 as usual:

Theorem 1. *$\mathcal{HL}^*$ is weakly complete w.r.t. countable, named models.*

There are two points to note here. First, unlike the classical case, we do not have strong completeness w.r.t. named models, as the Lindenbaum Lemma 2 would fail. Second, the preceding theorem asserts that $\mathcal{HL}^*$ is complete for models with countable carrier. This will be important for the completeness of the combined logic with respect to spatial transition systems. We conclude the section with a trivial corollary to the completeness theorem, which will be of fundamental importance later.

Corollary 1. *Suppose $\phi \in L(\mathcal{HL}^*)$ is local. Then ϕ is consistent iff $M \models \phi$ for some countable, named model M.*

This claim follows from the very nature of local formulas: a formula $@_i\phi$ is valid in a place iff it is valid in the place named i, hence local formulas are either globally true or globally false.

3 Temporalising Hybrid Logics

After having studied $\mathcal{HL}^*$ in isolation, we now add a temporal dimension to $\mathcal{HL}^*$. The logic $\mathcal{HL}^*$ allows us to reason about *where* a distributed computation happens; the temporal extension will furthermore furnish us with the expressive power to say *when* this will be the case.

The idea is quite simple: We consider linear temporal logic, but with atomic propositions replaced by local $\mathcal{HL}^*$-formulas. This is as in [15, Section 14], but with one important exception: In *loc.cit.*, the logic being temporalised is completely independent from the added temporal layer. In our case, spatial information needs to be propagated over time, leading to an entanglement of both dimensions. Semantically, this is reflected by or notion of model, which enforces consistency of names, and accounted for by an additional axiom in the proof calculus.

We call the resulting logic $\mathcal{LSTL}$. This logic naturally incorporates a temporal and a spatial aspect: the formulas of $\mathcal{HL}^*$ specify spatial properties *at a given point in time*, and temporal logic allows one to reason about the evolution of the spatial structure over time.

3.1 Linear Temporal Logic (A reminder)

Before we introduce $\mathcal{LSTL}$, let us briefly re-capitulate the syntax and semantics of propositional linear temporal logic. For a clear distinction between the propositional variables of the spatial and temporal logics, we denote the latter by T.

Definition 5. *Suppose* $\mathsf{T} = \{a_0, a_1, \dots\}$ *is a set (of atomic propositions). Then the language* $\mathcal{LTL}(\mathsf{T})$ *of* linear temporal logic *over* T *is the least set according to the grammar*

$$\phi, \psi \in \mathcal{LTL}(\mathsf{T}) ::= \mathsf{ff} \mid \phi \to \psi \mid \bigcirc\phi \mid \phi\,\mathcal{U}\,\psi \mid a$$

where $a \in \mathsf{T}$ *ranges over the set of propositional variables. As usual, the other connectives,* tt, $\vee$, $\wedge$, $\neg$ *can be defined from* ff *and* $\to$, *and we abbreviate* $\Diamond\phi = \mathsf{tt}\,\mathcal{U}\,\phi$ *and* $\Box\phi = \neg\Diamond\neg\phi$.

We call a sequence of valuations $V = (V_n)_{n\in\mathbb{N}}$ *of* T *a* temporal structure. *Given such a* $V = (V_n)_{n\in\mathbb{N}}$, *i.e. each* V_n *is of type* $\mathsf{T} \to \{\mathsf{tt}, \mathsf{ff}\}$, *the satisfaction relation is inductively given by*

$$\begin{array}{lll}(V,n) \models a & \textit{iff} & V_n(a) = \mathsf{tt} \\ (V,n) \models \bigcirc\phi & \textit{iff} & (V,n+1) \models \phi \\ (V,n) \models \phi\,\mathcal{U}\,\psi & \textit{iff} & \exists j \geq i.(V,j) \models \psi \text{ and } \forall i \leq k < j.\ (V,k) \models \phi\end{array}$$

where the semantics of propositional connectives is defined as usual. Finally, we put $V \models \phi$ *if* $(V,n) \models \phi$ *for all* $n \in \mathbb{N}$ *and* $\mathcal{LTL} \models \phi$ *if* $V \models \phi$ *for all temporal structures* V. *To distinguish satisfaction w.r.t. linear temporal logic, we sometimes write* $(V,n) \models_{\mathcal{LTL}} \phi$, *and similarly* $V \models_{\mathcal{LTL}} \phi$.

The formula $\bigcirc\phi$ is usually read as "ϕ is true in the next point in time", and $\phi\,\mathcal{U}\,\psi$ reads "ϕ is true until ψ becomes true". Similarly, $\Diamond\phi$ means that "ϕ will eventually become true", and finally $\Box\phi$ expresses that ϕ will be true in all future states. It is well known that the axioms

$$\begin{array}{llll}(\mathrm{taut}) & \text{all propositional tautologies} & (\mathrm{ltl1}) & \bigcirc\phi \wedge \bigcirc(\phi \to \psi) \to \bigcirc\psi \\ (\mathrm{ltl2}) & \phi\,\mathcal{U}\,\psi \leftrightarrow \psi \vee (\phi \wedge \bigcirc(\phi\,\mathcal{U}\,\psi)) & (\mathrm{ltl3}) & \bigcirc(\neg\phi) \to \neg\bigcirc\phi\end{array}$$

together with the inference rules

$$\text{(mp)}\ \frac{\phi, \phi \to \psi}{\psi} \qquad \text{(nex)}\ \frac{\phi}{\bigcirc\phi} \qquad \text{(ind)}\ \frac{\phi' \to \neg\psi \wedge \bigcirc\phi'}{\phi' \to \neg(\phi\, \mathcal{U}\, \psi)}$$

provide a complete axiomatisation of propositional linear temporal logic. We write $\mathcal{LTL} \vdash \phi$ if ϕ can be derived using the above axioms and rules. It is easy to check soundness of the above axioms and rules, and we have the following well-known completeness theorem [17,13,19]:

Theorem 2. *A formula $\phi \in LTL$ is valid in all temporal structures iff ϕ is derivable, i.e. $\mathcal{LTL} \models \phi \iff \mathcal{LTL} \vdash \phi$ for all $\mathcal{LTL}$-formulas ϕ.*

3.2 The Logic $\mathcal{LSTL}$

We now embark on the programme of temporalising $\mathcal{HL}^*$, which essentially amounts to replacing (temporal) propositions in $\mathcal{LTL}$-formulas by local $\mathcal{HL}^*$-formulas and the addition of an axiom that represents that names do not change over time. The resulting logic is called $\mathcal{LSTL}$, and the formal definition is as follows:

Definition 6. *The* language *of the logic $\mathcal{LSTL}$ is the language of linear temporal logic over the set $L(\mathcal{HL}^*) = \{\phi \in \mathcal{HL}^* \mid \phi \text{ local}\}$ of atoms, i.e. $\mathcal{LTL}(L(\mathcal{HL}^*))$. Note that propositional combinations of local formulas are not local anymore, but this does not matter as the propositional connectives are in $\mathcal{LTL}$ as well.*

A spatio-temporal structure *is a sequence $(S_n)_{n\in\mathbb{N}}$ of* named $\mathcal{HL}^*$*-models. The structure $(S_n)_{n\in\mathbb{N}}$ has* consistent naming, *if $S_0 \models @_i j$ iff $S_n \models @_i j$ for all $i, j \in \mathsf{Nom}$ and all $n \in \mathbb{N}$.*

Every spatio-temporal structure $(S_n)_{n\in\mathbb{N}}$ gives rise to a sequence of valuations

$$S_n^\sharp : L(\mathcal{HL}^*) \to \{\mathsf{tt}, \mathsf{ff}\}, \quad \phi \mapsto \begin{cases} \mathsf{tt} & S_n \models \phi \\ \mathsf{ff} & \textit{otherwise.} \end{cases}$$

Validity of a $\mathcal{LSTL}$ formula ϕ in a spatio-temporal structure is can now be defined by $(S_n)_{n\in\mathbb{N}} \models_{\mathcal{LSTL}} \phi$ iff $(S_n^\sharp)_{n\in\mathbb{N}} \models_{\mathcal{LTL}} \phi$, where the latter is the standard validity in linear temporal logic (Definition 5).

Finally, $\mathcal{LSTL} \models \phi$ iff $S \models_{\mathcal{LSTL}} \phi$ for all spatio-temporal structures S with consistent naming.

The reason for introducing structures with consistent naming is that in our view "names are physical", which in particular means that they do not change over time (like an IP address for example compared to a domain name that may change). Moreover, those names will provide the only glue between the models in a spatio-temporal structure. Consistent naming ensures that we can address the same physical location at different times via the same (physical) name. We conclude the section on syntax and semantics of $\mathcal{LSTL}$ with some examples.

Example 2 (Network routing). If we let places denote the nodes of a network and the spatial structure reflect the network topology, we are able to formulate assertions on the network and its routing of packets. We are only interested in a finite number of such nodes K. The packet with destination r is encoded as atomic proposition of $\mathcal{HL}^*$, denoted $\underline{r}$. We want to send it from s and thus assume that there is a spatial connection between nodes s and r (Reach). It is also assumed that the network does not change its spatial topology (Static) – and thus in particular does never lose any connections. Packet $\underline{r}$, wherever it may be, will always be broadcast to neighbour nodes (Broadcast). Finally, we have to ensure that – as messages are only broadcasted to neighbours in K – that $\underline{r}$ can reach its destination via a path that only visits nodes in K, which is implied by (Connect). In $\mathcal{LSTL}$ this reads as follows:

Reach $= @_s \diamondminus^* r$
Static $= \bigwedge_{p,q \in K} @_p \diamondminus q \;\to\; \Box @_p \diamondminus q$
Broadcast $= \bigwedge_{p,q \in K} (@_p \underline{r} \wedge @_p \diamondminus q) \to \bigcirc @_q \underline{r}$
Connect $= \bigwedge_{i \in K} @_i \boxminus \bigvee_{j \in K} j$

In such a situation one can derive that message $\underline{r}$ will eventually arrive, ie. $@_s \underline{r} \to \Diamond\, @_r \underline{r}$.

Example 3. Agents can be specified by describing the computation at various places in terms of state transitions. If agent A runs at place i and agent B runs at place j, and their state change is described by functions $\delta_A : S_A \to S_A$ and $\delta_B : S_B \to S_B$, respectively, then the system obtained by running A and B concurrently can be specified by

$$\bigwedge_{s \in S_A} @_i \varphi(s) \to \bigcirc @_i \varphi(\delta_A(s)) \;\wedge\; \bigwedge_{s \in S_B} @_i \varphi(s) \to \bigcirc @_j \varphi(\delta_B(s))$$

where $\varphi(\cdot)$ is a logical formula that characterises the respective state. If agent B "moves into" agent A after performing a state change from s_{mv} to s_e then this can be specified by

$$@_j \varphi(s_{mv}) \wedge \neg @_i \diamondminus j \wedge \neg @_j \diamondminus i \to \bigcirc @_i \diamondminus j \wedge @_j \varphi(s_e)$$

This movement is accounted for by the change of the spatial structure. This can be extended to describe behaviours of ambient like agents [8].

Example 4 (Leader election protocol). The following example is an adaptation of the IEEE 1394 Leader election protocol (see e.g. [26]). Let places again denote a finite number of network nodes. The network topology is described by a fixed *acyclic* (and finite) neighbourhood relation R. The network nodes are supposed to elect a leader.

Let the spatial structure represent the election results, i.e. how "local leaders" were chosen between each pair of connected nodes. Hence, we have $p \to q$ if p has chosen q to be its leader (and p and q are neighbours).

The protocol can be specified as follows: Initially, there are no connections between places (Init). Next$_{p,q}$ describes the situation where two nodes, p and q have not determined a leader between each other yet, and p is the only neighbour of q with that property. In such a case, q can become a subordinate of p, which is specified on the second line of Next. The first line specifies a "frame"-condition, namely that connections between places are always maintained, and for places who are not neighbours, do not even change. Goal states that for any two places in the neighbourhood relation one is the leader of the other. This implies that there is a leader for all nodes. Finally, Live axiomatises that if Goal is not (yet) true, there are places p and q that decide leadership amongst them in the next step.

The specification of the overall system then is: Init $\wedge$ Next $\wedge$ Live $\rightarrow$ Goal.

$$
\begin{array}{ll}
\text{Init} & = \bigwedge_p @_p \neg \diamondminus \mathsf{tt} \\
\text{Next}_{p,q} & = @_p \neg \diamondminus \mathsf{tt} \wedge @_q \neg \diamondminus \mathsf{tt} \wedge \bigwedge_{r \neq p, R(q,r)} @_r \diamondminus \mathsf{tt} \\
\text{Next} & = \bigwedge_{p,q} @_q \diamondminus p \rightarrow \bigcirc @_q \diamondminus p \ \wedge \bigwedge_{p,q,\neg R(p,q)} @_q \diamondminus p \iff \bigcirc @_q \diamondminus p \quad \wedge \\
 & \quad \bigwedge_{R(p,q)} (\bigcirc @_q \diamondminus p) \wedge @_q \neg \diamondminus p \rightarrow \text{Next}_{p,q} \\
\text{Goal} & = \bigwedge_{p \neq q, R(p,q)} @_p \diamondminus q \vee @_q \diamondminus p \\
\text{Live} & = \text{Goal} \vee \bigvee_{p,q} @_q \neg \diamondminus p \wedge \bigcirc @_q \diamondminus p
\end{array}
$$

We deem this formulation in $\mathcal{LSTL}$ more natural than the one given in [26].

Example 5 (XML documents). Let us specify an XML document with an active component. The spatial structure mirrors the XML document tree-structure, such that places correspond to *occurrences* of pairs of matching tags, i.e. $i \rightarrow j$ means that the XML-component at j is defined inside the one at i. The tags used and the text contained inside these tag are expressed as spatial propositions. As documents are finite, we are only interested in a finite set of places F. The document specified below has a *root* component (1), and a weather component somewhere under the root node (2). Moreover, if the weather component contains a temperature component, it will eventually fill in a valid integer representing the temperature in degrees (3).

1. $\bigwedge_{p \in F} @_{root} \diamondminus^* p \wedge \neg @_p \diamondminus^* root$
2. $@_{root} \diamondminus^* \langle \text{weather} \rangle$
3. $\bigwedge_{p \in F} @_p(\langle \text{weather} \rangle \wedge \diamondminus^* \langle \text{temp} \rangle) \rightarrow \Diamond\, @_p(\langle \text{weather} \rangle \wedge \diamondminus^* (\langle \text{temp} \rangle \wedge \text{valid\_int}))$

Note that the basic set of inference rules accounts for loops and self-reference in the structure of XML documents. While this is possible in some dialects of XML, e.g. Xlink [11] and other tree based query languages [6], it is easy to axiomatise special properties of trees in $\mathcal{LSTL}$. For example, the formula $\neg @_i \diamondminus^* i$ ensures that there are no cycles in the structure of the document.

3.3 Proof Rules of $\mathcal{LSTL}$

This section describes a complete axiomatisation of $\mathcal{LSTL}$. Extending [14], we enrich a standard and complete axiomatisation of $\mathcal{LTL}$ with the following rule and axiom scheme:

$$
(\text{emb}) \quad \frac{\mathcal{LTL} \vdash \phi}{\mathcal{LSTL} \vdash \phi} \qquad (\text{cn}) \quad @_i j \leftrightarrow \bigcirc @_i j
$$

to import spatial deduction into $\mathcal{LSTL}$ and to account for the fact that we are axiomatising structures with consistent naming, which is the main difference to [14], which presumes complete independence of the temporal component and the logic being temporalised.

Definition 7. *Suppose $\phi \in \mathcal{LSTL}$. Then $\mathcal{LSTL} \vdash \phi$ if ϕ is in the least set of formulas closed under (emb), (cn) and the axioms and rules of any complete axiomatisation of $\mathcal{LTL}$.*

It is straightforward to verify soundness of $\mathcal{LSTL}$.

Proposition 2 (Soundness of $\mathcal{LSTL}$). *Suppose $\phi \in \mathcal{LSTL}$. Then $\mathcal{LSTL} \models \phi$ if $\mathcal{LSTL} \vdash \phi$.*

3.4 Completeness of $\mathcal{LSTL}$

We now tackle completeness of $\mathcal{LSTL}$. Our construction is an extension of the construction presented in [14] that accounts for the fact that the rule (cn) axiomatises consistent naming, which is a property of spatio-temporal structures that cuts across time.

The proof of completeness fixes a fixed enumeration of a set $\mathsf{T} = \{p_0, p_1, p_2, \dots\}$ of propositional variables, that is used to encode sentences of $\mathcal{LSTL}$ in $\mathcal{LTL}$. We need the following technical terminology.

Definition 8. *For a fixed enumerations $L(\mathcal{HL}^*) = \{\phi_0, \phi_1, \phi_2 \dots\}$ we define the* correspondence mapping $\sigma : \mathcal{LTL}(L(\mathcal{HL}^*)) \to \mathcal{LTL}(\mathsf{T})$ *as the mapping $\phi_i \mapsto a_i$.*

Because we replace propositional reasoning when substituting $L(\mathcal{HL}^*)$-formulas for atoms in linear temporal logic, we need to encode the relations between the atoms on a purely propositional level in order to make use of completeness of $\mathcal{LTL}$. This is the purpose of the next definition.

Definition 9. *We inductively define the set* $\mathrm{Lit}(\phi) \subseteq \mathcal{HL}^*$ *of* literals *of $\phi \in \mathcal{LSTL}$ as follows:*

$$\mathrm{Lit}(\mathsf{ff}) = \emptyset \qquad \mathrm{Lit}(\phi \to \psi) = \mathrm{Lit}(\phi) \cup \mathrm{Lit}(\psi) \qquad \mathrm{Lit}(\bigcirc\phi) = \mathrm{Lit}(\phi)$$
$$\mathrm{Lit}(a) = \{a, \neg a\} \qquad \mathrm{Lit}(\phi\,\mathcal{U}\,\psi) = \mathrm{Lit}(\phi) \cup \mathrm{Lit}(\psi)$$

where $p \in \mathcal{HL}^*$ in the last line above. If $\phi \in \mathcal{LSTL}$, the set of inconsistencies *of ϕ is given as*

$$\mathrm{Inc}(\phi) = \{\textstyle\bigwedge \Phi \mid \Phi \subseteq \mathrm{Lit}(\phi) \text{ and } \Phi \vdash_{\mathcal{HL}^*} \mathsf{ff}\}.$$

Theorem 3. *The logic $\mathcal{LSTL}$ is weakly complete.*

Proof. Suppose $\phi \in \mathcal{LSTL}$ is consistent; we show that ϕ has a model, which is equivalent to the claim by contraposition. Let $\mathrm{nom}(\phi) = \bigcup\{\mathrm{nom}(\psi) \mid \psi \in \mathrm{Lit}(\phi)\}$ denote the set of nominals occurring in ϕ making use of nom for $\mathcal{HL}^*$- formulas (see Section 2). We now let

$$\hat{\phi} = \phi \wedge \bigwedge_{\psi \in \mathrm{Inc}(\phi)} \Box\neg\psi \wedge \bigwedge_{i,j \in \mathrm{nom}(\phi)} @_i j \leftrightarrow \bigcirc @_i j$$

Note that consistency of ϕ implies consistency of $\hat{\phi}$, which in turn implies consistency of $\sigma(\hat{\phi})$. Hence there exists a sequence $V = (V_n)$ of valuations of the propositional variables T s.t. $V \models_{\mathcal{LTL}} \sigma(\hat{\phi})$. The intuition behind the definition of $\hat{\phi}$ is that $\hat{\phi}$ encodes not only ϕ, but also all relations between its literals on a purely propositional level. This encoding ensures that propositionally valid literals are actually consistent in the logic $\mathcal{HL}^*$, a fact that is crucial for completeness, which we now address.

By construction, this valuation satisfies

$$V_0 \models \sigma(@_i j) \iff V_n \models \sigma(@_i j)$$

for all $n \in \mathbb{N}$ and all $i, j \in \mathrm{nom}(\phi)$. Take

$$G_n(\phi) = \{\psi \in \mathrm{Lit}(\phi) \mid V_n \models \sigma(\psi)\}.$$

Then all G_n are $\mathcal{HL}^*$-consistent (Lemma 14.2.17 of [15]). Moreover, we have $@_i j \in G_0(\phi) \iff @_i j \in G_n(\phi)$ for all $i, j \in \mathrm{nom}(\phi)$ and all $n \in \mathbb{N}$ by construction. As $G_n(\phi)$ consists of local formulas only, we can invoke Corollary 1 to obtain a countable named model S_n with $S_n \models G_n(\phi)$ for all $n \in \mathbb{N}$.

We can assume without loss of generality that the sequence (S_n) has constant naming, as $S_n \models @_i j \iff S_0 \models @_i j$ for $i, j \in \mathsf{Nom}(\phi)$ and $n \in \mathbb{N}$ and we can always change the valuation of nominals not occurring in ϕ (and hence G_n) without changing the validity of formulas.

Now $V \models \sigma(\hat{\phi})$ implies that $V \models \sigma(\phi)$ which implies $M \models \phi$ where the latter can be shown by induction on the structure of ϕ.

4 Spatial Transition Systems

The spatio-temporal structures of Def. 6 have one significant drawback, they are just arbitrary sequences of spatial models and there are no rules on how one spatial model evolves from its predecessors. As a remedy, and to bridge the gap between spatio-temporal structures and programming languages, spatial transition systems are introduced below. They are an abstraction of distributed programs. Completeness of $\mathcal{LSTL}$ with respect to these transitions systems will follow from the fact that every spatio-temporal structure arises as a run of a spatial transition system.

Definition 10. *A* spatial transition system *(STS) Θ consists of an enumerable set of* physical places P*, a surjective map $\eta : \mathsf{Nom} \to P$ mapping nominals – ie. (non-unique) place* names *– to physical places, and a P-indexed set of transition systems $(X_p, \to_p, \lambda_p, \mu_p, s_p^0)_{p \in P}$ such that*

- *X_p is the set of states of computations happening at place p,*
- *$\to_p \subseteq X_p \times X_p$ is the (possibly non-deterministic) state transition relation of the computation at place p. Transitions in $(X_p \times X_p)$ are autonomous transitions that can happen at place p.*
- *$\lambda_p : X_p \to \mathcal{P}(P)$ describes the spatial structure in terms of all connected neighbours of p at any state during the computation,*
- *$\mu_p : X_p \to \mathcal{P}(\mathsf{A})$ characterises the states of the computation at p by stating which (spatial) propositions hold in each state,*
- *s_p^0 is the initial state for the computation in p.*

A system state s of Θ is then a place indexed vector of states, i.e. $s \in \Pi_{p \in P}. X_p$. We write $s(p)$ for the component of s belonging to place p. A spatial transition system

$\Theta = (P, \eta, (X_p, \to_p, \lambda_p, \mu_p, s^0_p)_{p \in P})$ induces a transition relation $\to_\Theta$ on system states $s, s' \in \Pi_{p\in P}.\, X_p$ as follows:

$$s \to_\Theta s' \iff \exists Q \subseteq P(\forall p \in Q.\, s(p) \to_p s'(p) \text{ and } \forall p \notin Q. s(p) = s'(p))$$

with $s^\Theta = (s^0_p)_{p\in P}$ as initial state.

Runs of an STS are always infinite, as all the computations may be idle (choosing Q to be $\emptyset$). This provides us with a unified setting for finite and infinite computations. Moreover, the computations at different places may proceed in different speeds, reflected by the fact that at every tick of the synchronous clock describing the progress of a system state $s \in \Pi_{p\in P}.\, X_p$, some of the computations, precisely those in $P \setminus Q$, are idle. This is supposed to reflect the fact that the computations are actually running independently. Any computation in $p \in P$ can be non-deterministic if $\to_p$ is not the graph of a function.

Definition 11. *Every system state s for a STS Θ as described above gives rise to a named spatial model $Sp(s) = (P, \to_s, V_s, N_s)$ setting*

$$p \to_s q \iff q \in \lambda_p(s(p)), \quad V_s(a) = \{p \in P \mid a \in \mu_p(s(p))\}, \quad N_s = \eta.$$

The set of spatio-temporal structures generated by *the STS Θ, called $\mathsf{Run}(\Theta)$, contains all sequences of models generated by possible runs of Θ, i.e.*

$$\mathsf{Run}(\Theta) = \{(Sp(s_n))_{n\in\mathbb{N}} \mid s_0 = s^\Theta \wedge s_n \to_\Theta s_{n+1} \text{ for all } n \in \mathbb{N}\}.$$

As η in the definition of STS does not depend on the states of the STS, all spatio-temporal structures in $\mathsf{Run}(\Theta)$ have consistent naming. Validity for an STS is defined via a detour through the spatio-temporal structures:

$$\Theta \models \phi \iff \forall (S_n) \in \mathsf{Run}(\Theta).\, (S_n) \models \phi.$$

Due to the independent definition of the computations at places P, there cannot be any communication between them. Therefore, we will refine the notion of an STS shortly, but the present definition is sufficient to prove a completeness result.

Before we embark on completeness, we need one little technical lemma on consistent naming, which uses the following terminology: For a function $f : X \to Y$, the *kernel* of f is the set $\mathrm{Ke}(f) = \{(x, x') \in X \times X \mid f(x) = f(x')\}$. Note that $\mathrm{Ke}(f)$ is an equivalence relation.

Lemma 4. *Suppose $(S_n)_{n\in\mathbb{N}}$ is a spatio-temporal structure with consistent naming and $S_n = (P_n, \to_n, V_n, N_n)$. Then $\mathrm{Ke}(N_k) = \mathrm{Ke}(N_l)$ for all $k, l \in \mathbb{N}$ and $P \cong \mathsf{Nom}/\mathrm{Ke}(N_k)$ for all $k \in \mathbb{N}$.*

Lemma 5. *For an $\mathcal{LSTL}$ formula ϕ, if $\Theta \models \phi$ for all STS Θ, then $S \models \phi$ for all spatio-temporal structures S with consistent naming (according to Def. 6).*

Proof. Assume $\Theta \models \phi$ for all STS Θ and let a spatio-temporal structure S with consistent naming be given. Assume $S = (S_n, \to_n, V_n, P_n)$. By the last lemma, we can assume without loss of generality that $S_0 = \mathsf{Nom}/\mathrm{Ke}(N_0) = S_k$ for all $k \in \mathbb{N}$. We now show that S can be generated by a spatial transition system. We let $P = S_0$ and put $\eta(i) = N_0(i)$ for $i \in \mathsf{Nom}$. The components at each place $p \in P$ are given by:

- $X_p = \mathbb{N}$
- $n \rightarrow_p m$ iff $m = n + 1$
- $\lambda_p(n) = \{q \in S_n \mid p \rightarrow_n q\}$
- $\mu_p(n) = \{a \in \mathsf{A} \mid p \in V_n(a)\}$
- $s_p^0 = 0$.

Clearly $S \in \mathsf{Run}(\Theta)$, hence $S \models \phi$ by assumption.

Corollary 2. *The logic $\mathcal{LSTL}$ is weakly complete w.r.t. spatial transition systems.*

The transition systems defined above still do not provide means for programming synchronisation between computations (which can be used to program communication). Therefore we define *synchronised* spatial transition system as a superset of the spatial ones, ensuring that the completeness result above is not jeopardised. The main idea of a synchronised spatial transition system is the following. We equip the transition systems $(X_p, \rightarrow_p, \lambda_p, \mu_p, s_p^0)$ that model the system behaviour at place p with a *labelled* transition relation $\rightarrow_p \subseteq X_p \times \sigma \times X_p$, where σ is a set of labels that contains the distinguished label τ. We now stipulate that the system state s can evolve into a system state s' if either some of the processes make an internal transition (labelled with τ) or all processes capable of performing ℓ transitions participate in a synchronous transition, labelled with $\ell \neq \tau$. The formal definition reads as follows.

Definition 12. *A* synchronised spatial transition system *(SSTS) Σ is defined like a STS with an additional enumerable set of synchronisation labels σ with distinguished element $\tau \in \sigma$ and slightly changed transition systems $\rightarrow_p \subseteq (X_p \times \sigma \times X_p)$. We write $x \xrightarrow{\ell}_p y$ to indicate the $(x, \ell, y) \in \rightarrow_p$. For all $p \in P$ define* labels : $P \rightarrow \mathcal{P}(\sigma)$ *by* $\mathrm{labels}(p) = \{\ell \in \sigma \mid \ell \neq \tau \wedge \exists s, t \in X_p.\ s \xrightarrow{\ell}_p t\}$ *to denote all labels for which there are synchronised transitions for the computation at p. The system transitions for such a SSTS are now defined below, making sure that ℓ-synchronised transitions (for $\ell \neq \tau$) can only be performed if all processes with ℓ-labelled transitions actually fire ℓ-transitions synchronously. We stipulate $s \rightarrow_\Sigma s'$ if one of the following two conditions are satisfied:*

1. *$\exists Q \subseteq P(\forall p \in Q.\ s(p) \xrightarrow{\tau}_p s'(p)$ and $\forall p \notin Q. s(p) = s'(p))$, or*
2. *$\exists \ell \neq \tau \in \sigma(\forall p \in P_\ell . s(p) \xrightarrow{\ell}_p s'(p)$ and $\forall p \notin P_\ell . s(p) = s'(p))$.*

where $P_\ell = \mathrm{labels}^{-1}(\{\ell\})$ denotes the set $\{p \in P \mid \ell \in \mathrm{labels}(p)\}$ of places that can fire an ℓ-transition.

Example 6 (Leader Election Protocol IEEE 1394). We present an SSTS that fulfils the specification given in Example 4. The SSTS is defined as follows: Set $P = \mathsf{Nom}$ and $\eta = id$. For $p \in \mathsf{Nom}$ let $X_p = (\mathbb{N}, \mathcal{P}(\mathsf{Nom}))$ such that $\lambda_p(\_, x) = x$. The first component keeps track of the number of neighbours with which p has yet to decide about the leadership. Let σ contain a label $\ell_{\{i,j\}}$ for each pair of names, such that $R(i, j)$. Remember that R is the fixed neighbourhood relation describing the topology of an acyclic network. The initial state for each p is now $(\mathrm{card}\{j \in \mathsf{Nom} \mid R(j, p)\}, \emptyset)$.

For every $i, j \in \mathsf{Nom}$ such that $R(i, j)$ we have transitions

$$(1, x) \xrightarrow{\ell_{\{i,j\}}}_i (0, x \cup \{j\}) \qquad (n + 1, x) \xrightarrow{\ell_{\{i,j\}}}_j (n, x)$$

According to this definition, any node i can only chose j to be its leader if j is a neighbour, and j is the only neighbour that has not yet become a subordinate to another node.

Corollary 3. *The logic $\mathcal{LSTL}$ is weakly complete w.r.t. synchronised spatial transition system.*

This follows from Corollary 2 and the fact that every STS is also an SSTS.

5 Conclusions

By blending well-known ingredients, hybrid logic and linear temporal logic, extending a recipe from [14], we obtained a logic for reasoning about time and space for distributed computations that we proved to be complete. Our model is capable of representing many situations that naturally arise in distributed computing, including the behaviour of distributed agents (Example 3). Further research is necessary to investigate whether the spatial formulas can be extended, e.g. by hybrid quantifiers that could replace the finite conjunctions in our examples; this work will be guided by [16]. On the spatio-temporal side the question remains how to reflect synchronisation on the logical level.

References

1. Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.
2. Patrick Blackburn and Miroslava Tzakova. Hybrid completeness. *Logic Journal of the IGPL*, 6(4):625–650, 1998.
3. Patrick Blackburn and Miroslava Tzakova. Hybrid languages and temporal logic. *Logic Journal of the IGPL*, 7(1):27–54, 1999.
4. Luís Caires and Luca Cardelli. A spatial logic for concurrency (part I). *Inf. Comput.*, 186(2):194–235, 2003.
5. Luís Caires and Luca Cardelli. A spatial logic for concurrency - II. *Theor. Comput. Sci.*, 322(3):517–565, 2004.
6. Cristiano Calcagno, Philippa Gardner, and Uri Zarfaty. Context logic and tree update. In *POPL '05: Proceedings of the 32nd symposium on Principles of programming languages*, pages 271–282, New York, NY, USA, 2005. ACM Press.
7. Luca Cardelli, Philippa Gardner, and Giorgio Ghelli. A spatial logic for querying graphs. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 597–610. Springer, 2002.
8. Luca Cardelli and Andrew D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *POPL*, pages 365–377, 2000.
9. Luca Cardelli and Andrew D. Gordon. *Mobile ambients*, pages 198–229. Cambridge University Press, New York, NY, USA, 2001.
10. Rohit Chadha, Damiano Macedonio, and Vladimiro Sassone. A distributed Kripke semantics. Technical Report 2004:04, University of Sussex, 2004.
11. W3C consortium. Xlink language version 1.0.
12. Rocco de Nicola, Gian Luigi Ferrari, and Rosario Pugliese. Klaim: A kernel language for agents interaction and mobility. *IEEE Trans. Softw. Eng.*, 24(5):315–330, 1998.

13. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
14. Marcelo Finger and Dov Gabbay. *Adding a Temporal Dimension to a Logic System*, chapter 14, pages 524–552. Volume 1 of *Oxford Logic Guides* [15], 1994.
15. Dov Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects: Volume I.* Number 28 in Oxford Logic Guides. Oxford University Press, 1994.
16. Dov Gabbay, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. *Many-dimensional Modal logics: Theory and Applications*. Elsevier, 2003.
17. Dov Gabbay, Amir Pnueli, Saharon Shela, and Johnatan Stavi. On the temporal analysis of fairness. In *Proc. of the 7th ACM Symp. on Principles of Programming Languages*, pages 163–173. ACM press, 1980.
18. Daniel Hirschkoff. An extensional spatial logic for mobile processes. In Philippa Gardner and Nobuko Yoshida, editors, *Proc. of 15th Int. Conf. CONCUR 2004*, volume 3170 of *Lecture Notes in Computer Science*, pages 325–339. Springer, 2004.
19. Fred Kröger. *Temporal Logic of Programs*, volume 8 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1987.
20. Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, 1992.
21. Stephan Merz, Martin Wirsing, and Júlia Zappe. A spatio-temporal logic for the specification and refinement of mobile systems. In Mauro Pezzè, editor, *Proc. of 6th Int. Conf. Fundamental Approaches to Software Engineering (FASE) 2003*, volume 2621 of *Lecture Notes in Computer Science*, pages 87–101. Springer, 2003.
22. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes. *Inf. Comput.*, 100(1):1–40, 1992.
23. Robin Milner, Joachim Parrow, and David Walker. Modal logic for mobile processes. *Theoretical Computer Science*, 1(114):149–171, 1993.
24. Rocco De Nicola and Michele Loreti. A modal logic for mobile agents. *ACM Trans. Comput. Logic*, 5(1):79–128, 2004.
25. Solomon Passy and Tinko Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93, 1991.
26. Júlia Zappe. Towards a mobile TLA. In M. Nissim, editor, *ESSLI Student Workshop on Logic*, 2002.