MODEL CHECKING DYNAMIC-EPISTEMIC SPATIAL LOGIC

Radu Mardare and Corrado Priami

March 2006

Technical Report # DIT-06-011

Model checking Dynamic Epistemic Spatial Logics*

Radu Mardare[1] and Corrado Priami[1,2]
[1]University of Trento, Italy
[2]Microsoft Research - University of Trento Center
for Computational and Systems Biology

Abstract

We propose a new class of logics for specifying and model-checking properties of distributed systems - Dynamic Epistemic Spatial Logics. They have been designed as extensions of Hennessy-Milner logic with spatial operators (inspired by Cardelli-Caires spatial logic) and epistemic operators (inspired by dynamic-epistemic logics). Our logics focus on observers, agents placed in different locations of the system having access to some subsystems. Treating them as epistemic agents, we develop completely axiomatized and decidable logics that express the information flow between them in a dynamic and distributed environment. The knowledge of an epistemic agent, is understood as the information, locally available to our observer, about the overall-global system. By combining the knowledge of different observers we can specify properties of the whole system.
Dynamic Epistemic Spatial Logics are decidable against a semantics based on a fragment of CCS for which the classical spatial logics have been proved to be undecidable. Eventually model-checking and satisfiability/validity-checking algorithms are presented.

1 Introduction

The notions of *calculational process* or *algorithm* are not new in mathematics. They were studied long before the development of computing technology. Still, the invention of modern computers and latter the development of computer networks came with new challenges and new paradigms of computation.

The concept of *monolithic computational systems* (one-agent system) was replaced by the *concurrent distributed computing systems* (multi-agent systems), which represent programs or processors running in parallel and organized in networks of subsystems, each subsystem having its own identity. They interact, collaborate, communicate and interrupt each other. Underlying this new paradigm is the assumption that each part of such a system has its own identity, which persists through time. We shall associate to such a part (subsystem) an *agent*, that might be understood as an observer placed in a given point of our system and having access to this subsystem.

We need the notion of agents in order to discriminate between the events of the systems behavior. Indeed, if we wish to identify a particular event we have little choice but to identify the agents involved. Hence the agents might be understood as (associated with) separate and independently observable units of behavior and computation. They evolve in a given environment, following some primitive rules, their evolution influencing the structure of the whole (multi-agent) system. The main feature of the agents is their ability to communicate, that is to exchange information inside their environment.

These agents might not be topologically bound in the network, but able to change their relative positions. A laptop, for example, can be connected to the computer network at some

*Work partially supported by EU-IST project 016004 SENSORIA

point, it can start running some programs that interact with the network and, further, it might be unplugged and plugged back at a different point. Meanwhile, the laptop is running its programs independently of the whole system.

Such a multi-agent system reflects interactive, concurrent and distributed behaviors and computations of agents, thus is extremely complex. The success in dealing with this complexity depends on the mathematical model we choose to abstract the system. Further we focus on two major paradigms.

The agent is nothing more but its behavior

Process Algebra [3] abstracts the agents of the system on the level of their behavior and using some algebraic calculi and operational semantics [26] describes the evolution of the whole system. Largely used in applications, this paradigm succeeds in modelling complex computational scenarios. Further, as the behavior of a concurrent system is a succession of affine states in (possibly branching) time, was considered the possibility of applying modal (especially temporal) logics for specifying properties of the processes that modelled distributed systems.

In studying security problems, for example, we may want to be able to specify systems composed by agents that deal with fresh or secret resources. We may want to express properties such as "*the agent has the key*", "*eventually the agent crosses the firewall*" or "*there is always at most one agent here able to decrypt the message*".

In systems biology [9] we need to handle big complex systems having extreme dimensions and variable environments. We need to express properties such as "*somewhere there is a virus*", "*if the virus will meet the macrophage cell then it will be engulfed and eventually destroyed*", or "*the presence of the protein x will stimulate the reaction X*", etc.

Hennessy-Milner logic [19] is one of the first modal logics that proposes some modal operators, indexed by actions, to describe the behavior of the systems in CCS. It introduces, in top of the classical propositional logic, a dynamic operator $\langle\alpha\rangle\phi$ to express the property of a system that can perform the sequence of computations α and then reach a state described by ϕ. The idea was further developed in combination with temporal operators [27] and applied to other process calculi [25, 14, 16]. Latter, Mads Dam introduced a tensor that can express properties of modularity in the system [15], i.e. it can identify subsystems of a system. All these logics are characterized by their *extensional nature*, meaning that they cannot distinguish between processes that behave the same, even if these processes are different.

An increased degree of expressiveness is needed for specifying and reasoning about notions such as locations, resources, independence, distribution, connectivity and freshness. The specific applications of mobile computing call for properties that hold at particular locations, and it becomes natural to consider spatial modalities for expressing properties that hold at a certain location, at some locations or at every location. Thus, *Spatial logics* [6, 5, 11] propose, in addition to the modal temporal operators, some modal spatial operators such as the *parallel operator* $\phi|\psi$ (meaning that the current system can be split into a parallel composition of two subsystems, one satisfying ϕ and the other satisfying ψ), and its adjoint - the *guarantee operator* $\phi \triangleright \psi$ (if we compose in parallel any system P that satisfies ϕ with a system Q that satisfies $\phi \triangleright \psi$, then the composed system $P|Q$ satisfies ψ), or *location operators*[1] such as $n[\phi]$ (meaning that the current system can be described as a box $n[P]$ containing a subsystem P that satisfies ϕ), etc. A formula in a spatial logic describes a property of a particular part of the system at a particular time. These spatial modalities have an *intensional flavor*, the properties they express being invariant only for simple spatial rearrangements of the system.

Still most of the spatial logics face with decidability problems: it was proved that the basic spatial operators, in combination with temporal operators, generate undecidable logics [7, 13, 12] even against a finite piece of CCS. The situation is caused by the presence of the

[1]These operators are characteristic for Ambient Logic [11], a special spatial logic developed for Ambient Calculus [10].

guarantee operator, which involves a universal quantifier over the class of processes. In the light of the results presented in literature we have two alternatives for avoiding undecidability: either we choose a logic based on a static calculus [8], thus the logic cannot specify properties of our system in evolution, or we choose a dynamic calculus, but we have to avoid the use of a guarantee operator [4, 22, 21], hence we can express only local properties of the system.

The second alternative is useful only if our system is an isolated one (there is no upper-system for it) and we have a full description of it. In this sense the possible applications are quite limited. In modelling, for example, the scenario of the laptop connected to Internet, such a solution is not acceptable, as the whole system (Internet) is, theoretically, an infinite environment. We may consider upper-systems of our subsystem (laptop), but we cannot decide how far up we should go with modeling in order to obtain the information we are looking for, such as security issues.

Hence a spatial operator to express global properties within the limits of decidability is needed. To the best of our knowledge, no such alternative to guarantee operator has been proposed in the literature.

An agent is defined by its "knowledge"

The other paradigm of modelling multi-agent systems is inspired by logics and philosophy: reasoning about systems in terms of *knowledge of the agents* [17]. The knowledge of an agent is understood as the sum of actions the agent (subsystem) may take as a function of its local state in a given environment. Thus the agent "knows" its *protocol* in a given system. If we think to the agent as being an observer placed in our system which has access to a subsystem, its knowledge is the information related to evolution of this subsystem in an unknown environment.

Epistemic logics [17] formalize, in a direct manner, notions of knowledge, possessed by an agent, or a group of agents, using modalities like $K_A\phi$ (*A knows* ϕ), or $Ck\phi$ (*all the agents knows* ϕ*, i.e.* ϕ *is a common knowledge*). These logics supports Kripke-model based semantics, each basic modality being associated with a binary *accessibility relation* in these models. Thus for each epistemic agent A we devise an accessibility relation $\xrightarrow{A}$, called *indistinguishability relation for* A, expressing the agent's uncertainty about the current state. The states s' such that $s \xrightarrow{A} s'$ are the epistemic alternatives of s to agent A: if the current state of the whole system is s, A thinks that any of the alternatives s' may be the current state (as it doesn't have enough information to distinguish them). These logics have been extensively studied and applied to multi-agent systems.

Within computer science, reasoning about knowledge plays an extremely important role in contemporary theories of (intelligent) agents and it has been proved to be useful in modelling and understanding complex communication-based multi-agent systems.

Dynamic logics [18] are closer to process calculi, in that they have names for programs (*actions*) and operators to combine them. Accessibility relations are interpreted as transitions induced by programs, and a dynamic modality $[\pi]\phi$ captures the weakest precondition of such a program w.r.t. a given post-specification ϕ. Modalities in a dynamic logic form an algebraical structure: programs are built using basic program constructors such as *sequential composition* $\pi.\pi'$ or *iteration* π^*.

By mixing dynamic and epistemic formalisms Dynamic Epistemic Logics have been developed [1, 2, 20, 28, 29, 30], aiming to capture properties of information flow, such as communication, in multi-agent systems. These logics combine a rich expressivity with low complexity ensuring decidability and complete axiomatizations.

Our approach

The two paradigms of modelling concurrent distributed systems - the process algebraical paradigm and the epistemic one - were developed in parallel, but to our knowledge, there has been no unified paradigm. We propose such a paradigm in this paper, used for constructing a new logic for concurrency completely axiomatized and decidable. The main idea is to combine the features of spatial logics with the epistemic logics thus obtaining a special type of dynamic epistemic logic equipped with spatial operators. We call it Dynamic Epistemic Spatial Logic.

More concretely, our logic extends Hennessy-Milner logic with the parallel operator (hence it is a spatial logic) and epistemic operators. The role of the epistemic operators is to do most of the job of the guarantee operator while maintaining decidability. In our logics the epistemic agents are named by the processes they are related with. Thus $K_P\phi$ means *the agent related with P knows ϕ* and it holds iff ϕ is satisfied by any process having P as subprocess. The intuition is that the agent related with P is an observer inside our system that can see only P. So, as epistemic agent, it cannot differentiate between the global states P, $P|Q$ or $P|R$ of the whole system, as in all these states it sees only P. Thus its knowledge rests on the properties ϕ that are satisfied by each of these states (processes).

We prove, for Dynamic Epistemic Spatial Logic, the finite model property with respect to the chosen semantics. Thus, we have decidability for satisfiability, validity and model-checking problems. In proving the finite model property we use a new congruence on processes - *the structural bisimulation*. Informally, it is an approximation of the structural congruence bound by two dimensions: the *height* and the *weight* of a process.

For the logic we propose a complete Hilbert-style axiomatic system, which helps in understanding the basic algebraical behavior of the classical process operators. We prove its soundness and completeness with respect to the piece of CCS for which the classic spatial logic has been proved to be undecidable in [7]. Thus, many properties can be syntactically verified and proved. Moreover the interplay of our logical operators allows to express, in the syntax, validity and satisfiability for formulas. We also have characteristic formulas able to identify a process (agent) up to structural congruence (cloned copies).

The sound-complete axiomatic system in the context of decidability allows the development of algorithms for solving, in a finite manner, the satisfiability, validity and model-checking problems.

Concluding, the novelty of our logic with respect to the classical spatial logics is the use of the epistemic operators, as alternative to guarantee operator, for expressing global properties while ensuring decidability. The epistemic operators allow to refer directly to agents of our system by mean of their knowledge. An epistemic agent is, thus, an observer that can be placed in different places in our system and has access to partial information. By combining these partial information ("points of view" of different observers) we can specify complex properties of distributed systems. Further, due to decidability, we can syntactically verify and prove these properties.

From the epistemic logics perspective, we propose a new class of epistemic logics by imposing an algebraical structure (CCS-like) on the class of epistemic agents. In this way we may assume compositional and hierarchically organized agents. Thus P and Q are epistemic agents, but also $P|Q$ may be another agent. As they are ontologically related (P and Q are ontological subsidiary of $P|Q$), our logic allows to derive relations between their knowledge and dynamics from their ontological relations. In the classical epistemic logics [17] the agents are assumed to be ontologically independent entities, while our logics accepts dependencies. Other peculiarities of our epistemic logic comes from the fact that we can activate and deactivate agents: thus in a system having the current state described by $\alpha.P$, the agent that sees P is not active, but it might be activated in a future state. Our logic allows also cloned agents. Thus in a system described by $P|Q|P$ we have two clones of the agent seeing P.

Thus, we can model simultaneously, as agents in a system, individuals, societies of individuals, societies of societies of individuals, etc and their evolutions. All these features are new for epistemic logics.

$$P|0 \equiv P \qquad P|Q \equiv Q|P \qquad P|(Q|R) \equiv (P|Q)|R$$

Table 1: The axioms the structural congruence

Outline of the paper

The paper is organized as follows. In section 2 we present the process calculus on which we will focus for the rest of the paper. This calculus provides a semantics against which the classical spatial logic is undecidable. Section 3 defines some concepts in process algebra which will be used further, such as structural bisimulation and pruning processes and sets of processes. Starting with section 4 we define our logics. Two such systems will be introduced $\mathcal{L}_{DS}$ and its extension $\mathcal{L}_{DES}$. For both we will prove the finite model property that entails the decidability against the process semantics for which the classic spatial logic was proved to be undecidable. We also develop sound complete Hilbert-style axiomatic systems that comprehend the behavior of the logical operators involved. In section 6, underpinning on finite model property, we develop finite algorithms for satisfiability, validity and model-checking properties of distributed systems. We end the paper with some concluding remarks.

In addition we added an appendix where some examples related with the new concepts introduced in our paper are presented together with an example where our logic is used to specify properties for a small system.

For the proofs of the theorems presented in this paper, and for additional results the reader is referred to [24] for Dynamic Epistemic Spatial Logic and to [23] for Dynamic Spatial Logic.

2 Undecidability results in Spatial Logics

The main reason for introducing spatial logics is to provide appropriate techniques for specifying and model-checking properties of concurrent distributed systems, therefore most of the work done in this field points to decidability-related problems. We briefly present hereafter the (un)decidability results for spatial logics, proved in [7], which motivated our work.

Definition 2.1 (Processes). Consider the fragment of CCS generated by the next syntax, where $\mathbb{A}$ is a denumerable set of actions and $\alpha \in \mathbb{A}$:

$$P ::= 0 \mid \alpha.P \mid P|P$$

Hereafter this calculus[2] is the object of our paper. We will use α, β to range over $\mathbb{A}$ and we will denote by $\mathfrak{P}$ the class of processes.

Definition 2.2 (Structural congruence). The *relation of structural congruence* is defined as the least congruence $\equiv$ on processes satisfying the axioms in table 1.

Definition 2.3 (Transition systems). The *transition system* for the previously defined calculus is $ITS = \langle \mathfrak{P}, \mathbb{A}, \longrightarrow \rangle$, where $\longrightarrow \subset \mathfrak{P} \times \mathbb{A} \times \mathfrak{P}$ is the *transition relation* defined by the rules in table 2, with the assumption that $P \xrightarrow{\alpha} Q$ denotes $\langle P, \alpha, Q \rangle \in \longrightarrow$. We denote by $\longrightarrow^*$ the transitive closure of $\longrightarrow$.

For this calculus, in [7], were considered two spatial logics:

- $\mathcal{L}_{spat}$ given by the syntax

$$\phi ::= \top \mid 0 \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1|\phi_2 \mid \phi_1 \triangleright \phi_2 \mid \diamond\phi$$

[2]We can, additionally, consider an involution on $\mathbb{A}$ that associate to each action $\alpha \in \mathbb{A}$ an action $\overline{\alpha} \in \mathbb{A}$, as usual in CCS, and also to take into consideration the silent action τ. But all these represent just syntactic sugar, irrelevant from the point of view of the logic we discuss.

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P \equiv Q \qquad P \xrightarrow{\alpha} P'}{Q \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

Table 2: The transition system

$P, v \models_M \top$ for any process P
$P, v \models_M \neg\phi$ iff $P, v \not\models \phi$
$P, v \models_M \phi \wedge \psi$ iff $P, v \models_M \phi$ and $P, v \models_M \psi$
$P, v \models_M 0$ iff $P \equiv 0$
$P, v \models_M \phi|\psi$ iff $P \equiv Q|R$, $Q, v \models_M \phi$ and $R, v \models_M \psi$
$P, v \models_M \phi \triangleright \psi$ iff for any process $Q, v \models_M \phi$ we have $P|Q, v \models_M \psi$
$P, v \models_M \exists x.\phi$ iff $\exists \alpha \in \mathbb{A}$ such that $P, (v\{x \leftarrow \alpha\}) \models_M \phi$
$P, v \models_M \langle x \rangle \phi$ iff $\exists Q.P \xrightarrow{v(x)} Q$ and $Q, v \models_M \phi$

Table 3: Semantics of Spatial Logics

- $\mathcal{L}_{mod}$ given, over an infinite set of variables $X \ni x$, by the syntax

$$\phi ::= \top \mid 0 \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1|\phi_2 \mid \phi_1 \triangleright \phi_2 \mid \diamond\phi \mid \langle x \rangle \phi \mid \exists x.\phi$$

A *valuation* is a mapping from a finite subset of X to $\mathbb{A}$. For any valuation v, we write $v\{x \leftarrow \alpha\}$ for the valuation v' such that $v'(x) = \alpha$, and $v'(y) = v(y)$ if $y \neq x$.
The semantics for the two spatial logics, defined by the *satisfaction relation* $P, v \models_M \phi$ where P is a process, M is a set of processes that contains P, ϕ a formula, and v is a valuation for the free variables of ϕ, is presented in Table 3.

In [7] it is proved that $\mathcal{L}_{spat}$ can encode $\mathcal{L}_{mod}$, hence they are equally expressive. Then it is proved that model-checking and validity/satisfiability checking for $\mathcal{L}_{spat}$ with respect to this finite fragment of CCS are all undecidable. But $\mathcal{L}_{spat}$ is the core of all Spatial Logics.

Concluding, though expressive and useful, most of the spatial logics proved to be undecidable, even in the absence of quantifiers. Unlike in static spatial logics, the composition adjunct adds to the expressiveness of the logic, so that adjunct elimination is not possible for dynamic spatial logics, even quantifier-free [7].

To the best of our knowledge, no alternative operator, to replace the guarantee one in order to express global properties and still ensuring decidability, has been studied. We propose further such an alternative.

3 Processes and contexts

In this section, focusing on the fragment of CCS introduced in definition 2.1, we develop some concepts on which we will base the further constructs.

Assumption (Representativeness modulo structural congruence). *As the structural congruence is the ultimate level of expressivity we want for our logic, hereafter in the paper we will speak about processes up to structural congruence.*

Definition 3.1. We call a process P *guarded* iff $P \equiv \alpha.Q$ for $\alpha \in \mathbb{A}$. We introduce the notation $P^k \overset{def}{=} \underbrace{P|...|P}_{k}$, and convey to denote $P^0 \equiv 0$.

We extend the operators from processes to sets of processes.

Definition 3.2. For any sets of processes $M, N \subset \mathfrak{P}$ and any $\alpha \in \mathbb{A}$ we define:

$$\alpha.M \stackrel{def}{=} \{\alpha.P \mid P \in M\} \qquad\qquad M|N \stackrel{def}{=} \{P|Q \mid P \in M, Q \in N\}$$

As we speak about processes up to structural congruence, the parallel operator on sets of processes will be commutative, associative and will have $\{0\}$ as null.

Now we define the *contexts*. The intuition is that a *context* $\mathcal{M}$ is a (possibly infinite) set of processes that contains, in a maximal manner, any process representing a possible state of our system or of a subsystem of our system. Hence if a process belongs to a context then any process obtained by pruning its syntactic tree should belong to the context, as it might represent a subsystem or the information collected by an external observer in a bound time. For the same reason, the context should be also closed to transitions.

We associate to each process P the set $\pi(P)$ of all processes obtained by pruning the syntactic tree of P.

Definition 3.3 (Pruning the syntactic tree). For $P \in \mathfrak{P}$ we define[3] $\pi(P) \subset \mathfrak{P}$ by:

1. $\pi(0) \stackrel{def}{=} \{0\}$ 2. $\pi(\alpha.P) \stackrel{def}{=} \{0\} \cup \alpha.\pi(P)$ 3. $\pi(P|Q) \stackrel{def}{=} \pi(P)|\pi(Q)$

We extend the definition of π to sets of processes $M \subset \mathfrak{P}$ by $\pi(M) \stackrel{def}{=} \bigcup_{P \in M} \pi(P)$.

Definition 3.4 (Context). *A context is a nonempty set $\mathcal{M} \subseteq \mathfrak{P}$ of processes such that*

1. if $P \in \mathcal{M}$ and $P \longrightarrow P'$ then $P' \in \mathcal{M}$ 2. if $P \in \mathcal{M}$ then $\pi(P) \subset \mathcal{M}$

3.1 Size of a process

Further we define the *size of a process*, following a similar idea developed in [8] for sizes of trees. The intuition is that the process has a *height* given by the vertical size of its syntactic tree, and a *width* equal to the maximum number of bisimilar subprocesses that can be identified in a node of the syntactic tree.

Definition 3.5 (Size of a process). We define, inductively, the *size* (h, w) (h stays for *height* and w for *width*) of a process P, denoted by $[\![P]\!]$:

1. $[\![0]\!] \stackrel{def}{=} (0, 0)$ 2. $[\![P]\!] \stackrel{def}{=} (h, w)$ iff
   – $P = (\alpha_1.Q_1)^{k_1}|(\alpha_2.Q_2)^{k_2}|...|(\alpha_j.Q_j)^{k_j}$ and $[\![Q_i]\!] = (h_i, w_i)$, $i \in 1..j$
   – $h = 1 + max(h_1, ..., h_k)$, $w = max(k_1, ..., k_j, w_1, ..., w_j)$

We convey to write $(h_1, w_1) \leq (h_2, w_2)$ for $h_1 \leq h_2$ and $w_1 \leq w_2$ and $(h_1, w_1) < (h_2, w_2)$ for $h_1 < h_2$ and $w_1 < w_2$.

In example A.1, in appendix, we show the sizes for some processes.

Definition 3.6 (Size of a set of processes). Let $M \subset \mathfrak{P}$. We write $[\![M]\!] = (h, w)$ iff $(h, w) = max\{[\![P]\!] \mid P \in M\}$[4].

3.2 Substitutions

For the future constructs is also useful to introduce the substitutions of actions in a process.

Definition 3.7 (The set of actions of a process). We define $Act(P) \subset \mathbb{A}$, inductively by:

1. $Act(0) \stackrel{def}{=} \emptyset$ 2. $Act(\alpha.P) \stackrel{def}{=} \{\alpha\} \cup Act(P)$ 3. $Act(P|Q) \stackrel{def}{=} Act(P) \cup Act(Q)$

For a set $M \subset \mathfrak{P}$ of processes we define $Act(M) \stackrel{def}{=} \bigcup_{P \in M} Act(P)$.

[3]We consider also $\pi(P)$ defined up to structural congruence.

[4]Observe that not all the sets of processes have a size, as for an infinite one it might be not possible to have the maximum required.

Definition 3.8 (Action substitution). We call *action substitution* any function $\sigma : \mathbb{A} \longrightarrow \mathbb{A}$. We extend it further, syntactically, from actions to processes, $\sigma : \mathfrak{P} \longrightarrow \mathfrak{P}$, by

1. $\sigma(0) \overset{def}{=} 0$ 2. $\sigma(P|Q) \overset{def}{=} \sigma(P)|\sigma(Q)$ 3. $\sigma(\alpha.P) \overset{def}{=} \sigma(\alpha).\sigma(P)$

For $M \subset \mathfrak{P}$ let $\sigma(M) \overset{def}{=} \{\sigma(P) \mid P \in M\}$. We also use notation M^σ, P^σ for $\sigma(M)$ and $\sigma(P)$. The *set of actions of* σ, $act(\sigma)$, is defined as $act(\sigma) \overset{def}{=} \{\alpha, \beta \in \mathbb{A} \mid \alpha \neq \beta,\ \sigma(\alpha) = \beta\}$.

3.3 Structural bisimulation

The *structural bisimulation* is a congruence on processes (then extended to contexts) defined as an approximation of the structural congruence bound by two sizes: the *height* (the depth of the syntactic tree) and the *weight* (the maximum number of bisimilar subprocesses that can be found in a node of the syntactic tree) of a process. A conceptually similar congruence was proposed in [8] for analyzing trees of location for the static ambient calculus.

The structural bisimulation analyzes the behavior of a process focusing on a boundary (h, w) of its syntactic tree. The intuition is that $P \approx_h^w Q$ (P and Q are structurally bisimilar on size (h, w)) iff when we consider for both processes their syntactic trees up to the depth h only (we prune them on the height h) and we ignore the presence of more than w parallel bisimilar subprocesses in any node of the syntactic trees (we prune the trees on weight w), we obtain identical syntactic trees.

Definition 3.9 (Structural bisimulation). Let $P, Q \in \mathfrak{P}$. We define $P \approx_h^w Q$ by:
$P \approx_0^w Q$ always
$P \approx_{h+1}^w Q$ iff for any $i \in 1..w$ and any $\alpha \in \mathbb{A}$ we have

- if $P \equiv \alpha.P_1|...|\alpha.P_i|P'$ then $Q \equiv \alpha.Q_1|...|\alpha.Q_i|Q'$ with $P_j \approx_h^w Q_j$, for $j = 1..i$
- if $Q \equiv \alpha.Q_1|...|\alpha.Q_i|Q'$ then $P \equiv \alpha.P_1|...|\alpha.P_i|P'$ with $Q_j \approx_h^w P_j$, for $j = 1..i$

In example A.2, in appendix, we exemplify this relation for some processes.

Theorem 3.1 (Congruence). *$\approx_h^w$ is a congruence on processes.*

We extend the definitions of structural bisimulation from processes to contexts.

Definition 3.10 (Structural bisimulation over contexts). Let $\mathcal{M}, \mathcal{N}$ be two contexts. We write $\mathcal{M} \approx_h^w \mathcal{N}$ iff

1. for any $P \in \mathcal{M}$ there is a $Q \in \mathcal{N}$ with $P \approx_h^w Q$
2. for any $Q \in \mathcal{N}$ there is a $P \in \mathcal{M}$ with $P \approx_h^w Q$

We convey to write $(\mathcal{M}, P) \approx_h^w (\mathcal{N}, Q)$ for the case when $P \in \mathcal{M}$, $Q \in \mathcal{N}$, $P \approx_h^w Q$ and $\mathcal{M} \approx_h^w \mathcal{N}$.

3.4 Pruning processes and contexts

We introduce an effective method to construct, given a process P, a minimal process Q that has an established size (h, w) and is structurally bisimilar to P on this size. Because the construction is based on pruning the syntactic tree of P on a given size, we call this method *bound pruning*, and we refer to Q as *the pruned of P on the size* (h, w).

Theorem 3.2 (Bound pruning theorem). *For any process $P \in \mathfrak{P}$ and any (h, w) exists a process $Q \in \mathfrak{P}$ with $P \approx_h^w Q$ and $[\![Q]\!] \leq (h, w)$.*

Proof. We describe the construction[5] of Q by induction on h.

For $h = 0$**:** we just take $Q \equiv 0$, because $P \approx_0^w Q$ and $[\![0]\!] = (0, 0)$.

[5]This construction is not necessarily unique.

For $h+1$**:** suppose that $P \equiv \alpha_1.P_1|...|\alpha_n.P_n$.
Let P'_i be the result of pruning P_i by (h,w) (we use the inductive step of construction) and $P' \equiv \alpha_1.P'_1|...|\alpha_n.P'_n$. As for any $i = 1..n$ we have $P_i \approx^w_h P'_i$ (by the inductive hypothesis), we obtain, using theorem 3.1, that $\alpha_i.P_i \approx^w_{h+1} \alpha_i.P'_i$ and further $P \approx^w_{h+1} P'$.
Consider the canonical representation of $P' \equiv (\beta_1.Q_1)^{k_1}|...|(\beta_m.Q_m)^{k_m}$.
Let $l_i = min(k_i, w)$ for $i = 1..m$. Then we define $Q \equiv (\beta_1.Q_1)^{l_1}|...|(\beta_m.Q_m)^{l_m}$. Obviously $Q \approx^w_{h+1} P'$ and as $P \approx^w_{h+1} P'$, we obtain $P \approx^w_{h+1} Q$. By construction, $[\![Q]\!] \leq (h+1, w)$. $\square$

Definition 3.11 (Bound pruning processes). For a process P and for a tuple (h,w) we denote by $P_{(h,w)}$ the process obtained by pruning P to the size (h,w) by the method described in the proof of theorem 3.2.

In example A.3, in appendix, we show how a process can be pruned on different sizes.

Further we define the bound pruning of a context $\mathcal{M}$ as the context generated by the set of pruned processes of $\mathcal{M}$.

Definition 3.12 (System of generators for a context). We say that the set $M \subset \mathfrak{P}$ is a system of generators for the context $\mathcal{M}$ if $\mathcal{M}$ is the smallest context that contains M. We denote this by $\overline{M} = \mathcal{M}$.

Definition 3.13 (Bound pruning contexts). For any context $\mathcal{M}$ and any (h,w) we define

$$\mathcal{M}_{(h,w)} \overset{def}{=} \overline{\{P_{(h,w)} \mid P \in \mathcal{M}\}}$$

Theorem 3.3. *For any context $\mathcal{M}$, any $P \in \mathcal{M}$, and any size (h,w) we have*

$$(\mathcal{M}, P) \approx^h_w (\mathcal{M}_{(h,w)}, P_{(h,w)})$$

Definition 3.14. Let $A \subset \mathbb{A}$. We denote by $\mathfrak{P}^A_{(h,w)}$ the set of all processes with the size at most (h,w) and the actions in A, and by $\mathfrak{M}^A_{(h,w)}$ the set of all contexts generated by subsets of $\mathfrak{P}^A_{(h,w)}$:

$$\mathfrak{P}^A_{(h,w)} \overset{def}{=} \{P \in \mathfrak{P} \mid Act(P) \subseteq A,\ [\![P]\!] \leq (h,w)\},\ \mathfrak{M}^A_{(h,w)} \overset{def}{=} \{\overline{M} \subset \mathfrak{P} \mid Act(M) \subseteq A,\ [\![M]\!] \leq (h,w)\}$$

Theorem 3.4. *If $A \subset \mathbb{A}$ is a finite set of actions, then the following hold:*

1. $\mathfrak{P}^A_{(h,w)}$ is finite *2. If $\mathcal{M} \in \mathfrak{M}^A_{(h,w)}$ then $\mathcal{M}$ is a finite context* *3. $\mathfrak{M}^A_{(h,w)}$ is finite*

Theorem 3.5 (Bound pruning theorem). *Let $\mathcal{M}$ be a context. Then for any (h,w) there is a context $\mathcal{N} \in \mathfrak{M}^{Act(\mathcal{M})}_{(h,w)}$ such that $\mathcal{M} \approx^w_h \mathcal{N}$. Moreover, $\mathcal{N} = \mathcal{M}_{(h,w)}$ has this property.*

4 Logics for specifying distributed systems

In this section we introduce Dynamic Spatial Logic, $\mathcal{L}_{DS}$, as an extension of Hennessy-Milner logic with the parallel operator and Dynamic Epistemic Spatial Logic, $\mathcal{L}_{DES}$, which extends $\mathcal{L}_{DS}$ with the epistemic operators. The intuition is to define the knowledge of the process P in the context $\mathcal{M}$ as the common properties of the processes in $\mathcal{M}$ that contain P as subprocess. If we think to the epistemic agent as to an observer that can see only the process P, then its knowledge about any state of global system concerns only P. Thus, for it, the global states $P|Q$ and $P|R$ looks indistinguishable. Hence the knowledge implies a kind of universal quantifier over $\mathcal{M}$, since $K_P\phi$, if is satisfied by a process $P|Q$, then it is satisfied by any process $P|R \in \mathcal{M}$. We find this enough for expressing most of the properties considered in the spatial logic literature, which required the use of the guarantee operator.

The satisfiability relations will evaluate a formula to a process in a context.

For our logics, we propose Hilbert-style axiomatic systems proved to be sound and complete with respect to process semantics. $\mathcal{L}_{DS}$ and $\mathcal{L}_{DES}$ satisfy the finite model property against the process semantics that entails the decidability for satisfiability, validity and model checking for both logics.

4.1 Syntax

Definition 4.1 (Languages). We define the language of Dynamic Spatial Logic, $\mathcal{F}_{DS}$, and the language of Dynamic Epistemic Spatial Logic, $\mathcal{F}_{DES}$, for $\alpha \in \mathbb{A}$, by:
$\phi := \ 0 \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid \phi|\phi \mid \langle\alpha\rangle\phi$ $\qquad (\mathcal{F}_{DS})$
$\phi := \ 0 \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid \phi|\phi \mid \langle\alpha\rangle\phi \mid K_Q\phi$ $\qquad (\mathcal{F}_{DES})$

Definition 4.2 (Derived operators). In addition we introduce some derived operators:

1. $\bot \stackrel{def}{=} \neg\top$
2. $\phi \vee \psi \stackrel{def}{=} \neg((\neg\phi) \wedge (\neg\psi))$
3. $\phi \rightarrow \psi \stackrel{def}{=} (\neg\phi) \vee \psi$
4. $[\alpha]\phi \stackrel{def}{=} \neg(\langle\alpha\rangle(\neg\phi))$
5. $1 \stackrel{def}{=} \neg((\neg 0) \mid (\neg 0))$
6. $\langle !\alpha\rangle\psi \stackrel{def}{=} (\langle\alpha\rangle\psi) \wedge 1$
7. $\widetilde{K}_Q\phi \stackrel{def}{=} \neg K_Q \neg\phi$

Anticipating the semantics, we outline the intuition that motivates the choice of the operators.

The formula 0 is meant to characterize any process structurally congruent with 0 in any context, expressing *"there is no activity here"*. It should not be confused with "*false*".[6]

$\top$ will be satisfied by any process in any context. $\bot$ will be used to express the inconsistency.

The reason for introducing the parallel operator $\phi|\psi$ is that we want to be able to express, as in other spatial logics, the situation in which our system is composed by two parallel subsystems, one satisfying ϕ and the other satisfying ψ.

The dynamic-like operator $\langle\alpha\rangle\phi$ is meant to be used to speak about the transitions of our system. It expresses *"the system may perform the action α meeting a state satisfying ϕ"*.

The dynamic-like operator $[\alpha]\phi$, the dual operator of $\langle\alpha\rangle\phi$, expresses the situation where either the system cannot perform α, or if the system can perform α then any future state that can be reached by performing α satisfies ϕ.

The formula 1 is meant to describe the situation in which the system cannot be decomposed into two non-trivial subsystems. 1 can describe also the trivial system 0.

The formula $\langle !\alpha\rangle\psi$ expresses a process guarded by α, which, after consuming α, satisfies ψ.

The formula $K_Q\phi$ express a global property of the context: *"in our context any system having Q as subsystem has the property ϕ"*. Observe the universal quantifier involved.

We could also introduce, for each action α, a derived operator[7] $\langle\alpha,\overline{\alpha}\rangle$ to express communication by α, supposing that we have defined an involution $co : \mathbb{A} \longrightarrow \mathbb{A}$ which associates to each action α its co-action $\overline{\alpha}$:

$$\langle\alpha,\overline{\alpha}\rangle\phi \stackrel{def}{=} \bigvee_{\phi\leftrightarrow\phi_1|\phi_2} \langle\alpha\rangle\phi_1|\langle\overline{\alpha}\rangle\phi_2$$

4.2 Process semantics

A formula of $\mathcal{F}_{DS}$, or of $\mathcal{F}_{DES}$, will be evaluated to processes in a given context, by mean of a satisfaction relation $\mathcal{M}, P \models \phi$.

[6]We insist on this aspect as some syntaxes of classical logic use 0 for denoting *false*. This is not our intention. We use $\bot$ to denote *false*.

[7]The disjunction is considered up to logically-equivalent decompositions $\phi \leftrightarrow \phi_1|\phi_2$ that ensures the use of a finitary formula.

Definition 4.3 (Models and satisfaction). A model of $\mathcal{L}_{DS}$ or of $\mathcal{L}_{DES}$ is a context $\mathcal{M}$ for which we define the satisfaction relation, for $P \in \mathcal{M}$, as follows:
$\mathcal{M}, P \models \top$ always
$\mathcal{M}, P \models 0$ iff $P \equiv 0$
$\mathcal{M}, P \models \neg\phi$ iff $\mathcal{M}, P \not\models \phi$
$\mathcal{M}, P \models \phi \wedge \psi$ iff $\mathcal{M}, P \models \phi$ and $\mathcal{M}, P \models \psi$
$\mathcal{M}, P \models \phi|\psi$ iff $P \equiv Q|R$ and $\mathcal{M}, Q \models \phi$, $\mathcal{M}, R \models \psi$
$\mathcal{M}, P \models \langle\alpha\rangle\phi$ iff there exists a transition $P \xrightarrow{\alpha} P'$ and $\mathcal{M}, P' \models \phi$
$\mathcal{M}, P \models K_Q\phi$ iff $P \equiv Q|R$ and $\forall Q|R' \in \mathcal{M}$ we have $\mathcal{M}, Q|R' \models \phi$

Then the semantics of the derived operators will be:
$\mathcal{M}, P \models [\alpha]\phi$ iff for any $P' \in \mathcal{M}$ such that $P \xrightarrow{\alpha} P'$ (if any), $\mathcal{M}, P' \models \phi$
$\mathcal{M}, P \models 1$ iff $P \equiv 0$ or $P \equiv \alpha.Q$ (P is null or guarded)
$\mathcal{M}, P \models \langle!\alpha\rangle\phi$ iff $P \equiv \alpha.Q$ and $\mathcal{M}, Q \models \phi$
$\mathcal{M}, P \models \widetilde{K}_Q\phi$ iff either $P \not\equiv Q|R$ for any R, or it exists $Q|S \in \mathcal{M}$ such that $\mathcal{M}, Q|S \models \phi$

Remark the interesting semantics of the operators K_0 and $\widetilde{K}_0$:
$\mathcal{M}, P \models K_0\phi$ iff for any $Q \in \mathcal{M}$ we have $\mathcal{M}, Q \models \phi$
$\mathcal{M}, P \models \widetilde{K}_0\phi$ iff it exists a process $Q \in \mathcal{M}$ such that $\mathcal{M}, Q \models \phi$

If a process $P \in \mathcal{M}$ satisfies $K_0\phi$ then ϕ is valid in $\mathcal{M}$ (the same about $K_0\phi$) and vice versa. Hence we can encode, in the syntax, the validity with respect to a given context.

If a process $P \in \mathcal{M}$ satisfies $\widetilde{K}_0\phi$ (then all the processes in $\mathcal{M}$ satisfy $\widetilde{K}_0\phi$) then it exists a process $Q \in \mathcal{M}$ that satisfies ϕ and vice versa. Hence $\widetilde{K}_0\phi$ provides a way to encode the satisfiability with respect to a given model.

4.3 Characteristic formulas for processes

In this subsection we use the peculiarities of the dynamic and epistemic operators to define characteristic formulas for processes and for finite contexts. Such formulas will be useful in providing appropriate axiomatic systems for our logics and, eventually, for proving the completeness.

Definition 4.4 (Characteristic formulas for processes). In $\mathcal{F}_{DS}$ we define a class of formulas $(c_P)_{P \in \mathfrak{P}}$, indexed by ($\equiv$-equivalence classes of) processes, as follows:

1. $c_0 \overset{def}{=} 0$ 2. $c_{P|Q} \overset{def}{=} c_P|c_Q$ 3. $c_{\alpha.P} \overset{def}{=} \langle!\alpha\rangle c_P$

Theorem 4.1. *$\mathcal{M}, P \models c_Q$ iff $P \equiv Q$.*

As $\mathcal{F}_{DES}$ is an extension of $\mathcal{F}_{DS}$, $(c_P)_{P \in \mathfrak{P}}$ characterize processes also in $\mathcal{F}_{DES}$.

Specific for $\mathcal{F}_{DES}$ only is the possibility to exploit the semantics of the operators K_0 and $\widetilde{K}_0$, as they can describe validity and satisfiability w.r.t a model, in defining characteristic formulas for finite contexts.

Definition 4.5 (Characteristic formulas for contexts). In $\mathcal{F}_{DES}$, if $\mathcal{M}$ is a finite context, we can define its *characteristic formula* by:

$$c_{\mathcal{M}} = K_0(\bigvee_{Q \in \mathcal{M}} c_Q) \wedge (\bigwedge_{Q \in \mathcal{M}} \widetilde{K}_0 c_Q)$$

Suppose that $\mathcal{N}, P \models c_{\mathcal{M}}$. Then the first conjunct $K_0(\bigvee_{Q \in \mathcal{M}} c_Q)$ tells us that $\bigvee_{Q \in \mathcal{M}} c_Q$ is a validity in $\mathcal{N}$, hence each element of $\mathcal{N}$ is an element of $\mathcal{M}$, $\mathcal{N} \subseteq \mathcal{M}$. The second conjunct tells us that for each $Q \in \mathcal{M}$, $\mathcal{N}, P \models \widetilde{K}_0 c_Q$. By the semantics of $\widetilde{K}_0$ this means that it exists a process $P' \in \mathcal{N}$ such that $\mathcal{N}, P' \models c_Q$, i.e. $P' \equiv Q$. As the processes are identified up to structural congruence, $\mathcal{M} \subseteq \mathcal{N}$. Hence $\mathcal{M} = \mathcal{N}$.

Theorem 4.2. *If $\mathcal{M}$ is a finite context and $P \in \mathcal{M}$ then $\mathcal{M}, P \models c_{\mathcal{N}}$ iff $\mathcal{N} = \mathcal{M}$.*

4.4 Finite model property and decidability

Now we prove the finite model property for Dynamic Epistemic Spatial Logic that will entail the decidability against the process semantics. As a consequence, we obtain decidability for Dynamic Spatial Logic (being less expressive). Anticipating, we define a size for formulas ϕ; then we prove that if $\mathcal{M}, P \models \phi$ then substituting, by σ, all the actions in $\mathcal{M}$ (and implicitly in P) that are not in the syntax of ϕ (as indexes of dynamic or epistemic operators) by a fixed action with the same property, and then pruning $\mathcal{M}^\sigma$ and P^σ to the size of ϕ we will obtain a couple $(\mathcal{N}, Q)$ such that $\mathcal{N}, Q \models \phi$. The fixed action of substitution can be chosen as the successor[8] of the maximum action of ϕ, which is unique. Hence $\mathcal{N} \in \mathfrak{M}^A_{(h,w)}$ where (h, w) is the size of ϕ and A is the set of actions of ϕ augmented with the successor of its maximum, thus A is finite. But then theorem 3.4 ensures that the set of pairs $(\mathcal{N}, Q)$, with this property, is finite.

Definition 4.6 (Size of a formula). We define *the sizes of a formula*, $(\!|\phi|\!)$ (*height* and *width*), inductively on $\mathcal{F}_{DES}$, by:

1. $(\!|0|\!) = (\!|\top|\!) \overset{def}{=} (0, 0)$ 2. $(\!|\neg\phi|\!) \overset{def}{=} (\!|\phi|\!)$

and supposing that $(\!|\phi|\!) = (h, w)$, $(\!|\psi|\!) = (h', w')$ and $[\![R]\!] = (h_R, w_R)$ we define further:

3. $(\!|\phi \wedge \psi|\!) \overset{def}{=} (max(h, h'), max(w, w'))$ 4. $(\!|\phi|\psi|\!) \overset{def}{=} (max(h, h'), w + w')$
5. $(\!|\langle\alpha\rangle\phi|\!) \overset{def}{=} (1 + h, 1 + w)$ 6. $(\!|K_R\phi|\!) \overset{def}{=} (1 + max(h, h_R), 1 + max(w, w_R))$

The next theorem states that ϕ is *"sensitive"* via satisfaction only up to size $(\!|\phi|\!)$. In other words, the relation $\mathcal{M}, P \models \phi$ is conserved by substituting the couple (M, P) with any other couple (N, P) structurally bisimilar to it at the size $(\!|\phi|\!)$.

Theorem 4.3. *If $(\!|\phi|\!) = (h, w)$, $\mathcal{M}, P \models \phi$ and $(\mathcal{M}, P) \approx^w_h (\mathcal{N}, Q)$ then $\mathcal{N}, Q \models \phi$.*

Using this theorem, we conclude that if a process, in a context, satisfies ϕ then by pruning the process and the context on the size $(\!|\phi|\!)$, we still have satisfiability for ϕ. Indeed the theorems 3.2 and 3.3 prove that if $(\!|\phi|\!) = (h, w)$ then $(\mathcal{M}, P) \approx^h_w (\mathcal{M}_{(\!|\phi|\!)}, P_{(\!|\phi|\!)})$. Hence $\mathcal{M}, P \models \phi$ implies $\mathcal{M}_{(\!|\phi|\!)}, P_{(\!|\phi|\!)} \models \phi$.

Definition 4.7 (The set of actions of a formula). We define the set of actions of a formula ϕ, $act(\phi) \subset \mathbb{A}$, inductively by:

1. $act(0) \overset{def}{=} \emptyset$ 3. $act(\phi \wedge \psi) = act(\phi|\psi) \overset{def}{=} act(\phi) \cup act(\psi)$ 5. $act(K_R\phi) \overset{def}{=} Act(R) \cup act(\phi)$
2. $act(\top) \overset{def}{=} \emptyset$ 4. $act(\neg\phi) = act(\phi)$ 6. $act(\langle\alpha\rangle\phi) \overset{def}{=} \{\alpha\} \cup act(\phi)$

The next result states that a formula ϕ does not reflect properties that involves more then the actions in its syntax. Thus if $\mathcal{M}, P \models \phi$ then any substitution σ having the elements of $act(\phi)$ as fix points preserves the satisfaction relation, i.e. $\mathcal{M}^\sigma, P^\sigma \models \phi$.

Theorem 4.4. *If $\mathcal{M}, P \models \phi$ and σ is a substitution with $act(\sigma) \bigcap act(\phi) = \emptyset$ then $\mathcal{M}^\sigma, P^\sigma \models \phi$.*

We suppose to have defined on $\mathbb{A}$ a lexicographical order $\ll$. So, for a finite set $A \subset \mathbb{A}$ we can identify a maximal element that is unique. Hence the successor of this element is unique as well. We convey to denote by A_+ the set obtained by adding to A the successor of its maximal element. Moreover, for a context $\mathcal{N} \ni P$, for a size (h, w) and for a finite set of actions $A \subset \mathbb{A}$ we denote by $\mathcal{N}^A_{(h,w)}$ (and by $P^A_{(h,w)}$ respectively) the context (the process) obtained by substituting all the actions $\alpha \in Act(\mathcal{N}) \setminus A$ ($\alpha \in Act(P) \setminus A$ respectively) by the successor of the maximum element of A and then pruning the context (the process) obtained to size (h, w).

[8]We consider defined, on the class of actions $\mathbb{A}$, a lexicographical order.

Theorem 4.5 (Finite model property).

$$\textit{If } \mathcal{M}, P \models \phi \textit{ then } \exists \mathcal{N} \in \mathfrak{M}^{act(\phi)_+}_{(\!|\phi|\!)} \textit{ and } Q \in \mathcal{N} \textit{ such that } \mathcal{N}, Q \models \phi.$$

Moreover $\mathcal{N} = \mathcal{M}^{act(\phi)}_{(\!|\phi|\!)}$ *and* $Q = P^{act(\phi)}_{(\!|\phi|\!)}$ *fulfill the requirements of the theorem.*

Because $act(\phi)$ is finite implying $act(\phi)_+$ finite, we apply theorem 3.4 ensuring that $\mathfrak{M}^{act(\phi)_+}_{(\!|\phi|\!)}$ is finite and any context $\mathcal{M} \in \mathfrak{M}^{act(\phi)_+}_{(\!|\phi|\!)}$ is finite as well. Thus we obtain the finite model property for our logic. A consequence of theorem 4.5 is the decidability for satisfiability, validity and model checking against the process semantics.

Theorem 4.6 (Decidability of $\mathcal{L}_{DES}$). *For $\mathcal{L}_{DES}$ validity, satisfiability and model checking are decidable against the process semantics.*

Corollary 4.7 (Decidability of $\mathcal{L}_{DS}$). *For $\mathcal{L}_{DS}$ validity, satisfiability and model checking are decidable against the process semantics.*

4.5 Axiomatic Systems

In table 4 is proposed a Hilbert-style axiomatic system for $\mathcal{L}_{DS}$. We assume the axioms and the rules of propositional logic. In addition we will have a set of spatial axioms and rules, and a set of dynamic axioms and rules.

Spatial axioms			
S1:	$\vdash \top \vert \bot \to \bot$	S4:	$\vdash \phi \vert (\psi \vee \rho) \to (\phi \vert \psi) \vee (\phi \vert \rho)$
S2:	$\vdash (\phi \vert \psi) \vert \rho \to \phi \vert (\psi \vert \rho)$	S5:	$\vdash \phi \vert \psi \to \psi \vert \phi$
S3:	$\vdash \phi \vert 0 \leftrightarrow \phi$	S6:	$\vdash (c_P \wedge \phi \vert \psi) \to \bigvee_{P \equiv Q \vert R} (c_Q \wedge \phi) \vert (c_R \wedge \psi)$
Spatial rules			
SR1:	$\vdash \phi \to \psi$ then $\vdash \phi \vert \rho \to \psi \vert \rho$		
Dynamic axioms			
D7:	$\vdash \langle \alpha \rangle \phi \vert \psi \to \langle \alpha \rangle (\phi \vert \psi)$	D10:	For $\alpha_i \neq \beta$, $\vdash \langle !\alpha_1 \rangle \top \vert ... \vert \langle !\alpha_n \rangle \top \to [\beta] \bot$
D8:	$\vdash [\alpha](\phi \to \psi) \to ([\alpha]\phi \to [a]\psi)$	D11:	$\vdash \langle !\alpha \rangle \phi \to [\alpha] \phi$
D9:	$\vdash 0 \to [\alpha] \bot$		
Dynamic rules			
DR2:	$\vdash \phi$ then $\vdash [\alpha]\phi$	DR3:	If $\vdash \phi_1 \to [\alpha]\phi_1'$ and $\vdash \phi_2 \to [\alpha]\phi_2'$ *then* $\vdash \phi_1 \vert \phi_2 \to [\alpha](\phi_1' \vert \phi_2 \vee \phi_1 \vert \phi_2')$
DR4:	$\vdash \bigvee_{P \in \mathfrak{P}^{act(\phi)_+}_{(\!\vert\phi\vert\!)}} c_P \to \phi$ then $\vdash \phi$		

Table 4: The axiomatic system $\mathcal{L}_{DS}$

Concerning the axioms and rules we make two observations. The disjunction involved in axiom S6 is finitary, as we considered the processes up to structural congruence level. Also the disjunction involved in rule DR4 has a finite number of terms, as a consequence of the finite model property.

The axiomatic system for $\mathcal{L}_{DES}$ is just an extension of the system of $\mathcal{L}_{DS}$ with the set of epistemic axioms and rules presented in the table 5. Observe that also the rule DR4 has been

Dynamic rule

DR'4: $\vdash \bigvee_{\mathcal{M}\in\mathfrak{M}^{act(\phi)+}_{(\!|\phi|\!)}} c_{\mathcal{M}} \to \phi$ then $\vdash \phi$

Epistemic axioms

E12: $\vdash K_Q\phi \wedge K_Q(\phi \to \psi) \to K_Q\psi$
E13: $\vdash K_Q\phi \to \phi$
E14: $\vdash K_Q\phi \to K_Q K_Q\phi$
E15: $\vdash K_Q\top \to (\neg K_Q\phi \to K_Q\neg K_Q\phi)$

E16: If $P \in \mathfrak{S}$ then $\vdash K_P\top \leftrightarrow c_P|\top$
E17: $\vdash K_Q\phi \leftrightarrow (K_Q\top \wedge K_0(K_Q\top \to \phi))$
E18: $\vdash K_0\phi \wedge \psi|\rho \to (K_0\phi \wedge \psi)|(K_0\phi \wedge \rho)$
E19: $\vdash K_0\phi \to [\alpha]K_0\phi$
E20: $\vdash K_0\phi \to (K_Q\top \to K_Q K_0\phi)$

Epistemic rules

ER5: $\vdash \phi$ then $\vdash K_Q\top \to K_Q\phi$

ER6: If $\mathcal{M} \ni P$ is a finite context and
$\vdash c_{\mathcal{M}} \wedge c_P \to K_0\phi$ then $\vdash c_{\mathcal{M}} \to \phi$

Table 5: The axiomatic system $\mathcal{L}^{\mathfrak{S}}_{DES}$

replaced by DR'4, as this logic is sensitive to contexts (due to universal quantifier involved by the semantics of the epistemic operator).

For the epistemic axioms and rules we point on their similarities with the classic axioms of knowledge. Thus axiom E12 is the classical (K)-axiom stating that our epistemic operator is a normal one, while axiom E13 is just the necessity axiom, for the epistemic operator. Also axiom E14 is well known in epistemic logics. It states that our epistemic agents satisfy *the positive introspection property*: if P knows ϕ then it knows that it knows ϕ. Axiom E15 states a variant of the *negative introspection*, saying that if an agent P is active and if it doesn't know ϕ, then it knows that it doesn't know ϕ. These axioms are present in all the epistemic logics [17]. Axiom E16 is also interesting as it states the equivalence between *to be active* and *to know* for our epistemic agents.

4.6 Soundness and Completeness

The choice of the axioms is motivated by the soundness theorem.

Theorem 4.8 (Soundness). *The systems $\mathcal{L}_{DS}$ and $\mathcal{L}_{DES}$ are sound w.r.t. process semantics.*

Hence everything expressed by our axioms and rules about the process semantics is correct and, in conclusion, using our system, we can derive only theorems that can be meaningfully interpreted in CCS.

Further we state the completeness of $\mathcal{L}_{DS}$ and of $\mathcal{L}_{DES}$ with respect to process semantics. The intuition is that, because c_P is a characteristic formulas, we should have an equivalence between $\mathcal{M}, P \models \phi$ and $\vdash c_P \to \phi$ for $\mathcal{L}_{DS}$, and between $\mathcal{M}, P \models \phi$ and $\vdash c_{\mathcal{M}} \wedge c_P \to \phi$ for $\mathcal{L}_{DES}$. Using this intuition we proved the completeness theorem. Observe that $\mathcal{L}_{DS}$ logic is not sensitive to contexts, while $\mathcal{L}_{DES}$ is, because of the universal quantifier involved in the semantics of the epistemic operator.

Theorem 4.9 (Completeness). *The $\mathcal{L}_{DS}$ and $\mathcal{L}_{DES}$ are complete with respect to process semantics.*

The completeness ensures that everything that can be derived in the semantics can be proved as theorem. In this way we have the possibility to syntactically verify (prove) properties of distributed systems.

5 Validity, satisfiability and model-checking

In this section, underpinning on finite model property, we present some finite algorithms for validity, satisfiability and model-checking properties of concurrent distributed systems. The complexity of these algorithms remain to be studied in future works.

We begin with the case of model checking finite models as being a basic case to which all the other problems will be reduced.

Model checking on finite models: Given a finite model $\mathcal{M}$, a process $P \in \mathcal{M}$ and a formula ϕ, algorithm 5 decides, in finite time, if is the case that $\mathcal{M}, P \models \phi$, or equivalently if $CheckFin(\mathcal{M}, P : \phi) = \top$.

Algorithm 1 Model checking on finite models

```
$CheckFin(\mathcal{M}, P : \top) := \top$
$CheckFin(\mathcal{M}, P : 0) := \top$ if $P \equiv 0$
                        $:= \bot$ else
$CheckFin(\mathcal{M}, P : \neg\phi) := \neg CheckFin(\mathcal{M}, P : \phi)$
$CheckFin(\mathcal{M}, P : \phi \wedge \psi) := CheckFin(\mathcal{M}, P : \phi) \wedge CheckFin(\mathcal{M}, P : \psi)$
$CheckFin(\mathcal{M}, P : \phi|\psi) := \bigvee_{P \equiv Q|R}(\ CheckFin(\mathcal{M}, Q : \phi) \wedge CheckFin(\mathcal{M}, R : \psi)\ )$
$CheckFin(\mathcal{M}, P : \langle\alpha\rangle\phi) := \bigvee_{P \xrightarrow{\alpha} Q} CheckFin(\mathcal{M}, Q : \phi)$
$CheckFin(\mathcal{M}, P : K_Q\phi) := \bot$ if there is no $S$ such that $P \equiv Q|S$
                              $:= \bigwedge_{Q|R \in \mathcal{M}} CheckFin(\mathcal{M}, Q|R : \phi)$ else
```

Observe that the disjunctions involved in the algorithm are finitary, as the processes are considered up to structural congruence. Also the conjunction involved in the evaluation of epistemic formulas is finitary, as $\mathcal{M}$ is finite.

Theorem 5.1. *If $\mathcal{M}$ is a finite context and $P \in \mathcal{M}$ then*

$$\mathcal{M}, P \models \phi \text{ iff } CheckFin(\mathcal{M}, P : \phi) = \top$$

Model checking on arbitrary models: Using the algorithm for finite models we will develop further the general algorithm for model checking. Given a model $\mathcal{M}$, a process $P \in \mathcal{M}$ and a formula ϕ, algorithm 6 decides, in finite time, if it is the case that $\mathcal{M}, P \models \phi$, or equivalently if $Check(\mathcal{M}, P : \phi) = \top$. Indeed, by theorem 3.4 $\mathcal{M}^{act(\phi)}_{(\!|\phi|\!)}$ is a finite context, hence we can reuse the algorithm 5.

Algorithm 2 Model checking on arbitrary models

```
$Check(\mathcal{M}, P : \phi) := CheckFin(\mathcal{M}^{act(\phi)}_{(\!|\phi|\!)}, P^{act(\phi)}_{(\!|\phi|\!)} : \phi)$
Use algorithm5 to compute the value of $CheckFin(\mathcal{M}^{act(\phi)}_{(\!|\phi|\!)}, P^{act(\phi)}_{(\!|\phi|\!)} : \phi)$
```

Theorem 5.2. *If $\mathcal{M}$ is a finite context then $Check(\mathcal{M}, P : \phi) = CheckFin(\mathcal{M}, P : \phi)$.*

Theorem 5.3. *If $P \in \mathcal{M}$ then $\mathcal{M}, P \models \phi$ iff $Check(\mathcal{M}, P : \phi) = \top$.*

Satisfiability checking: Recall that the finite model property maps any satisfiability problem in a satisfiability problem over a finite domain. Indeed, given a formula ϕ, theorem 3.4 states that the set of pairs $(\mathcal{M}, P)$ with $P \in \mathcal{M} \in \mathfrak{M}^{act(\phi)+}_{(\!|\phi|\!)}$ is finite, as $act(\phi)$ is finite implying $act(\phi)_+$ finite. Further the finite model property entails that if there exists a couple $(\mathcal{N}, Q)$ such that $\mathcal{N}, Q \models \phi$ then it exists a context $\mathcal{M} \in \mathfrak{M}^{act(\phi)+}_{(\!|\phi|\!)}$ and a process $P \in \mathcal{M}$ such that

Algorithm 3 Satisfiability checking

```
$Sat(\phi) := \bigvee_{P \in \mathcal{M} \in \mathfrak{M}_{\llparenthesis\phi\rrparenthesis}^{act(\phi)+}} Check(\mathcal{M}, P : \phi)$
Use algorithm6 to compute the value of each $Check(\mathcal{M}, P : \phi)$ for $P \in \mathcal{M} \in \mathfrak{M}_{\llparenthesis\phi\rrparenthesis}^{act(\phi)+}$
```

$\mathcal{M}, P \models \phi$. Hence deciding on satisfiability can be reduced to deciding on finite number of model checking problems, hence it can be solved in finite time using the algorithm 7.

Theorem 5.4. *A formula $\phi \in \mathcal{F}_{DES}$ is satisfiable iff $Sat(\phi) = \top$*

Validity checking: In approaching the validity problem we have two equivalent alternatives: either we use the fact that validity of ϕ is equivalent with non-satisfiability of $\neg\phi$, i.e. $Vld(\phi) \overset{def}{=} \neg Sat(\neg\phi)$, or we can use the semantics of the operator $\widetilde{K}_0$ and define $Vld(\phi) \overset{def}{=} Sat(\widetilde{K}_0\phi)$.

Algorithm 4 Validity checking

```
$Vld(\phi) := Sat(\widetilde{K}_0\phi)$
Use algorithm7 to compute the value of $Sat(\widetilde{K}_0\phi)$
```

Theorem 5.5. *A formula $\phi \in \mathcal{F}_{DES}$ is valid iff $Vld(\phi) = \top$*

6 Validity, satisfiability and model-checking

In this section, underpinning on finite model property, we present some finite algorithms for validity, satisfiability and model-checking properties of concurrent distributed systems. The complexity of these algorithms remain to be studied in future works.

We begin with the case of model checking finite models as being a basic case to which all the other problems will be reduced.

Model checking on finite models: Given a finite model $\mathcal{M}$, a process $P \in \mathcal{M}$ and a formula ϕ, algorithm 5 decides, in finite time, if is the case that $\mathcal{M}, P \models \phi$, or equivalently if $CheckFin(\mathcal{M}, P : \phi) = \top$.

Algorithm 5 Model checking on finite models

```
$CheckFin(\mathcal{M}, P : \top) := \top$
$CheckFin(\mathcal{M}, P : 0) := \top$ if $P \equiv 0$
                          $:= \bot$ else
$CheckFin(\mathcal{M}, P : \neg\phi) := \neg CheckFin(\mathcal{M}, P : \phi)$
$CheckFin(\mathcal{M}, P : \phi \wedge \psi) := CheckFin(\mathcal{M}, P : \phi) \wedge CheckFin(\mathcal{M}, P : \psi)$
$CheckFin(\mathcal{M}, P : \phi|\psi) := \bigvee_{P \equiv Q|R}(\ CheckFin(\mathcal{M}, Q : \phi) \wedge CheckFin(\mathcal{M}, R : \psi)\ )$
$CheckFin(\mathcal{M}, P : \langle\alpha\rangle\phi) := \bigvee_{P \xrightarrow{\alpha} Q} CheckFin(\mathcal{M}, Q : \phi)$
$CheckFin(\mathcal{M}, P : K_Q\phi) := \bot$ if there is no $S$ such that $P \equiv Q|S$
                              $:= \bigwedge_{Q|R \in \mathcal{M}} CheckFin(\mathcal{M}, Q|R : \phi)$ else
```

Observe that the disjunctions involved in the algorithm are finitary, as the processes are considered up to structural congruence. Also the conjunction involved in the evaluation of epistemic formulas is finitary, as $\mathcal{M}$ is finite.

Theorem 6.1. *If $\mathcal{M}$ is a finite context and $P \in \mathcal{M}$ then*

$$\mathcal{M}, P \models \phi \text{ iff } CheckFin(\mathcal{M}, P : \phi) = \top$$

Model checking on arbitrary models: Using the algorithm for finite models we will develop further the general algorithm for model checking. Given a model $\mathcal{M}$, a process $P \in \mathcal{M}$ and a formula ϕ, algorithm 6 decides, in finite time, if it is the case that $\mathcal{M}, P \models \phi$, or equivalently if $Check(\mathcal{M}, P : \phi) = \top$. Indeed, by theorem 3.4 $\mathcal{M}^{act(\phi)}_{(\!|\phi|\!)}$ is a finite context, hence we can reuse the algorithm 5.

Algorithm 6 Model checking on arbitrary models

```
$Check(\mathcal{M}, P : \phi) := CheckFin(\mathcal{M}^{act(\phi)}_{(\!|\phi|\!)}, P^{act(\phi)}_{(\!|\phi|\!)} : \phi)$
Use algorithm5 to compute the value of $CheckFin(\mathcal{M}^{act(\phi)}_{(\!|\phi|\!)}, P^{act(\phi)}_{(\!|\phi|\!)} : \phi)$
```

Theorem 6.2. *If $\mathcal{M}$ is a finite context then $Check(\mathcal{M}, P : \phi) = CheckFin(\mathcal{M}, P : \phi)$.*

Theorem 6.3. *If $P \in \mathcal{M}$ then $\mathcal{M}, P \models \phi$ iff $Check(\mathcal{M}, P : \phi) = \top$.*

Satisfiability checking: Recall that the finite model property maps any satisfiability problem in a satisfiability problem over a finite domain. Indeed, given a formula ϕ, theorem 3.4 states that the set of pairs $(\mathcal{M}, P)$ with $P \in \mathcal{M} \in \mathfrak{M}^{act(\phi)_+}_{(\!|\phi|\!)}$ is finite, as $act(\phi)$ is finite implying $act(\phi)_+$ finite. Further the finite model property entails that if there exists a couple $(\mathcal{N}, Q)$ such that $\mathcal{N}, Q \models \phi$ then it exists a context $\mathcal{M} \in \mathfrak{M}^{act(\phi)_+}_{(\!|\phi|\!)}$ and a process $P \in \mathcal{M}$ such that $\mathcal{M}, P \models \phi$. Hence deciding on satisfiability can be reduced to deciding on finite number of model checking problems, hence it can be solved in finite time using the algorithm 7.

Algorithm 7 Satisfiability checking

```
$Sat(\phi) := \bigvee_{P \in \mathcal{M} \in \mathfrak{M}^{act(\phi)_+}_{(\!|\phi|\!)}} Check(\mathcal{M}, P : \phi)$
Use algorithm6 to compute the value of each $Check(\mathcal{M}, P : \phi)$ for $P \in \mathcal{M} \in \mathfrak{M}^{act(\phi)_+}_{(\!|\phi|\!)}$
```

Theorem 6.4. *A formula $\phi \in \mathcal{F}_{DES}$ is satisfiable iff $Sat(\phi) = \top$*

Validity checking: In approaching the validity problem we have two equivalent alternatives: either we use the fact that validity of ϕ is equivalent with non-satisfiability of $\neg\phi$, i.e. $Vld(\phi) \overset{def}{=} \neg Sat(\neg\phi)$, or we can use the semantics of the operator $\widetilde{K}_0$ and define $Vld(\phi) \overset{def}{=} Sat(\widetilde{K}_0\phi)$.

Algorithm 8 Validity checking

```
$Vld(\phi) := Sat(\widetilde{K}_0\phi)$
Use algorithm7 to compute the value of $Sat(\widetilde{K}_0\phi)$
```

Theorem 6.5. *A formula $\phi \in \mathcal{F}_{DES}$ is valid iff $Vld(\phi) = \top$*

7 Concluding remarks

In this paper we developed two decidable and complete axiomatized logics for specifying and model-checking concurrent distributed systems: Dynamic Spatial Logic - $\mathcal{L}_{DS}$ and Dynamic Epistemic Spatial Logic - $\mathcal{L}_{DES}$. They extend Hennessy-Milner logic with the parallel operator and respectively with epistemic operators. The lasts operators are meant to express global properties over contexts. We propose these operators as alternative to the guarantee operator of the classical spatial logics, in order to obtaining a logic adequately expressive and decidable.

$\mathcal{L}_{DES}$ is less expressive than the classic spatial logic. Using the guarantee operator and the characteristic formulas, we can express our epistemic operators in classic spatial logic, while guarantee operator cannot be expressed by using our logic: $K_Q\phi \stackrel{def}{=} c_Q|\top \wedge (\neg(c_Q|\top \rightarrow \phi) \triangleright \bot)$.

Validity and satisfiability in a model can be syntactically expressed in our logic. Combining this feature with the possibility to characterize processes and finite contexts, we may argue on utility of this logic in most of the CCS-like applications for which classic spatial logic was proposed.

In the context of decidability, our sound and complete Hilbert-style axiomatic systems provide powerful tools for making predictions on the evolution of the concurrent distributed systems. Knowing the current state or a sub-state of a system, we can characterize it syntactically. And because any other state can be characterized, we can project any prediction-like problem in syntax and verify its satisfiability. Hence if the system we considered can reach the state we check, we will obtain that the formula is satisfiable and this method will provide also a minimal model. These features allowed us to develop some finite algorithms for validity, satisfiability and model checking.

The axioms and rules considered are very similar to the classical axioms and rules in epistemic logic, and some derivable theorems state meaningful properties of epistemic agents. All these relates our logic with the classical epistemic/doxastic logics and focus the specifications on external observers as epistemic agents. This interpretation is consistent with the spirit of process algebras.

Further researches are to be considered such as optimizing the model-algorithms and adding other operators in logics to fit with more complex process calculi. Challenging will be also the perspective of considering recursion in semantics.

Acknowledgements. We thank to Alexandru Baltag for contributing with valuable comments, since the beginning, on the construction of this logic. Thanks also to Luca Cardelli for comments and related discussions. The name *structural bisimulation* was suggested to us by Gordon Plotkin.

References

[1] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese (: Special Section: Knowledge, Rationality and Action).Editors: J. Symons, J. Hintikka. Special Section Editor: W. van der Hoek. Springer Science+Business Media B.V. ISSN: 0039-7857*, 139 (2):165–224, 2004.

[2] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements. common knowledge and private suspicions. *CWI Technical Report SEN-R9922*, 1999.

[3] J. A. Bergstra. *Handbook of Process Algebra.* Elsevier Science Inc., New York, NY, USA, 2001.

[4] Luis Caires. Behavioral and spatial properties in a logic for the pi-calculus. *In Igor Walukiewicz, editor, Proc. of Foundations of Software Science and Computation Structures2004, Lecture Notes in Computer Science, Springer-Verlag*, vol:2987, 2004.

[5] Luis Caires and Luca Cardelli. A spatial logic for concurrency (part ii). *In Proceedings of CONCUR'2002, Lecture Notes in Computer Science, Springer-Verlag*, vol:2421, 2002.

[6] Luis Caires and Luca Cardelli. A spatial logic for concurrency (part i). *Information and Computation*, Vol: 186/2:194–235, November 2003.

[7] Luis Caires and Etienne Lozes. Elimination of quantifiers and decidability in spatial logics for concurrency. *In Proceedings of CONCUR'2004, Lecture Notes in Computer Science, Springer-Verlag*, vol:3170, 2004.

[8] Cristiano Calcagno, Luca Cardelli, and Andrew D. Gordon. Deciding validity in a spatial logic for trees. *In Proceedings of the ACM Workshop on Types in Language Design and Implementation*, pages 62–73, 2003.

[9] Luca Cardelli. Bioware languages. *In: Andrew Herbert, Karen Sprck Jones (Eds.): Computer Systems: Theory, Technology, and Applications - A Tribute to Roger Needham, Monographs in Computer Science. Springer*, ISBN 0-387-20170-X.:59–65., 2004.

[10] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98.* Springer-Verlag, Berlin Germany, 1998.

[11] Luca Cardelli and Andrew D. Gordon. Ambient logic. *To appear in Mathematical Structures in Computer Science*, 2003.

[12] Witold Charatonik, Andrew D. Gordon, and Jean-Marc Talbot. Finite-control mobile ambients. In *ESOP '02: Proceedings of the 11th European Symposium on Programming Languages and Systems*, pages 295–313. Springer-Verlag, 2002.

[13] Witold Charatonik and Jean-Marc Talbot. The decidability of model checking mobile ambients. *Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic, Springer-Verlag*, 2142 of Lecture Notes in Computer Science:339–354, 2001.

[14] M. Dam. Proof systems for π-calculus. *In de Queiroz, editor, Logic for Concurrency and Synchronisation, Studies in Logic and Computation. Oxford University Press. To appear.*

[15] M. Dam. Relevance logic and concurrent composition. *In Proceedings of Third Annual Symposium on Logic in Computer Science, Edinburgh, Scotland, July 1988. IEEE Computer Society.*, pages 178–185.

[16] M. Dam. Model checking mobile processes. *Information and Computation*, vol:129(1):35–51, 1996.

[17] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[18] D Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* MIT Press, 2000.

[19] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *JACM*, vol: 32(1):137–161, 1985.

[20] W. Groeneveld J. Gerbrandy. Reasoning about information change. *Journal of Logic, Language and Information*, 6:146–169, 1997.

[21] R. Mardare and C. Priami. A logical approach to security in the context of ambient calculus. *ENTCS*, vol. 99, 2004.

[22] R. Mardare, C. Priami, P. Quaglia, and A. Vagin. Model checking biological systems described using ambient calculus. *Proceedings of CMSB04, Lecture Notes in BioInformatics. Berlin: Springer-Verlag*, 3082: 3:85– 10, 2005.

[23] Radu Mardare. A decidable extension of hennessy-milner logic with spatial operators. *Technical Report DIT-06-009, Informatica e Telecomunicationi, University of Trento*, 2006.

[24] Radu Mardare. Dynamic epistemic spatial logics. *Technical Report, DIT-06-010, Department of Information and Communication Technology, University of Trento*, available at http://dit.unitn.it/research/publications/index.xml and also at http://www.dit.unitn.it/∼mardare/publications.htm, 2006.

[25] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, vol:114:149–171, 1993.

[26] Gordon D. Plotkin. A structural approach to operational semantics. *Technical Report FN-19, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark*, 43, September 1981.

[27] Colin Stirling. *Modal and temporal properties of processes*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[28] J. F. A. K. van Benthem. Games in dynamic epistemic logic. *Bulletin of Economic Research, Los Altos*, 53(4):219–248, 2001.

[29] J. F. A. K. van Benthem. Logic for information update. *In Proceedings of TARK01, Los Altos*, 2001.

[30] H. van Ditmarsch. Knowledge games. *Bulletin of Economic Research*, 53(4):249–273, 2001.

A Appendix: Some examples

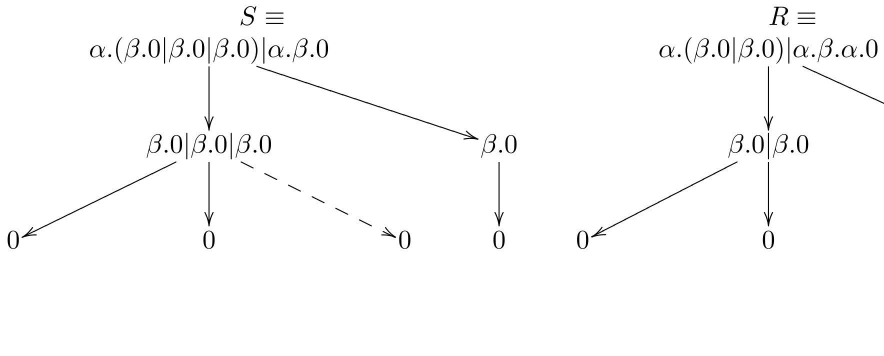
Example A.1 (Sizes of processes). *We show the size for some processes:*

1. $[\![0]\!] = (0,0)$
2. $[\![\alpha.0]\!] = (1,1)$
3. $[\![\alpha.0|\beta.0]\!] = (1,1)$
4. $[\![\alpha.0|\alpha.0]\!] = (1,2)$
5. $[\![\alpha.\alpha.0]\!] = [\![\alpha.\beta.0]\!] = (2,1)$
6. $[\![\alpha.(\beta.0|\beta.0)]\!] = (2,2)$

Example A.2 (Structural bisimulation). *Consider the processes*

$$R \equiv \alpha.(\beta.0|\beta.0|\beta.0)|\alpha.\beta.0 \textit{ and } S \equiv \alpha.(\beta.0|\beta.0)|\alpha.\beta.\alpha.0$$

We can verify the requirements of the definition 3.9 and decide that $R \approx_2^2 S$. But $R \not\approx_3^2 S$ because on the depth 2 R has an action α (in figure 1 marked with a dashed arrow) while S does not have it (because the height of S is only 2). Also $R \not\approx_2^3 S$ because R contains only 2 (bisimilar) copies of $\beta.0$ while S contains 3 (the extra one is marked with a dashed arrow). Hence, for any weight bigger than 2 this feature will show the two processes as different. But if



$$\begin{tikzcd}
 & S \equiv \alpha.(\beta.0|\beta.0|\beta.0)|\alpha.\beta.0 \arrow[d] \arrow[drr] & & \\
 & \beta.0|\beta.0|\beta.0 \arrow[dl] \arrow[d] \arrow[dr, dashed] & & \beta.0 \arrow[d] \\
0 & 0 & 0 & 0
\end{tikzcd}
\qquad
\begin{tikzcd}
 & R \equiv \alpha.(\beta.0|\beta.0)|\alpha.\beta.\alpha.0 \arrow[d] \arrow[dr] & \\
 & \beta.0|\beta.0 \arrow[dl] \arrow[d] & \beta.\alpha.0 \arrow[d] \\
0 & 0 & \alpha.0 \arrow[d, dashed] \\
 & & 0
\end{tikzcd}$$

Figure 1: Syntactic trees

we remain on depth 1 we have $R \approx_1^3 S$, as on this deep the two processes have the same number of bisimilar subprocesses, i.e. any of them can perform α in two ways giving, further, processes in the relation $\approx_0^3$. Indeed

$$R \equiv \alpha R'|\alpha R'', \textit{ where } R' \equiv \beta.0|\beta.0|\beta.0 \textit{ and } R'' \equiv \beta.0$$
$$S \equiv \alpha.S'|\alpha.S'', \textit{ where } S' \equiv \beta.0|\beta.0 \textit{ and } S'' \equiv \beta.\alpha.0$$

By definition, $R' \approx_0^3 S'$ and $R'' \approx_0^3 S''$

Example A.3 (Pruning processes). *Consider the process*

$$P \equiv \alpha.(\ \beta.(\gamma.0|\gamma.0|\gamma.0)\ |\ \beta.\gamma.0\ )\ |\ \alpha.\beta.\gamma.0$$

Observe that $[\![P]\!] = (3,3)$, hence $P_{(3,3)} \equiv P$. For constructing $P_{(3,2)}$ we have to prune the syntactic tree of P such that to not exist, in any node, more than two bisimilar branches. Hence $P_{(3,2)} = \alpha.(\ \beta.(\gamma.0|\gamma.0)\ |\ \beta.\gamma.0)\ |\ \alpha.\beta.\gamma.0$
If we want to prune P on the size $(3,1)$, we have to prune its syntactic tree such that, in any node, there are no bisimilar branches. The result is $P_{(3,1)} = \alpha.\beta.\gamma.0$.
For pruning P on the size $(2,2)$, we have to prune all the nodes on depth 2 and in the new tree we have to let, in any node, a maximum of two bisimilar branches. As a result of these modifications, we obtain $P_{(2,2)} = \alpha.(\beta.0|\beta.0)\ |\ \alpha.\beta.0$. Going further we obtain the smaller processes $P_{(0,0)} = 0$, $P_{(1,1)} = \alpha.0$, $P_{(1,2)} = \alpha.0|\alpha.0$, $P_{(2,1)} = \alpha.\beta.0$.

A small case study

Consider the scenario of the *e-mail box* receiving messages. *A message* can be a *spam* containing a *virus* that will be installed in our system if we open the attachment, or it can be a useful message that will provide information by opening its attachment. We can describe this scenario using a process calculus as follows.

The *Inbox* can be described as the agent that, being in contact with a message, can perform an *"open"* action (it opens the message), after which it can perform a *"run"* action that refers to the attachment (if any), after that stops.

$$Inb \overset{def}{=} open.run.0$$

A message containing harmless information is described as a process that can perform the action $\overline{open}$ (can be opened), then allows the Inbox to run its attachment (can perform $\overline{run}$), after which it reveals information.

$$Msg \overset{def}{=} \overline{open}.\overline{run}.Inf$$

A spam message has a structure similar to that of an ordinary message, but after consuming the $\overline{run}$ action it does not reveal information, but instals a virus.

$$Spm \overset{def}{=} \overline{open}.\overline{run}.Vrs$$

The intention in introducing the pairs of actions $(run, \overline{run})$, $(open, \overline{open})$ is to model communication-like complex actions.

Now, using the logic, we can express properties of a system that implies these processes. Suppose that we have two formulas, i, v, that describe Inf and Vrs respectively (such formulas do exist as we proved that each process P has a characteristic formula c_P and this formula uniquely describes the process up to structural congruence). As the Internet is a potentially infinite environment, for evaluating the formulas we consider as context the class of all processes $\mathfrak{P}$ (hereafter we omit to write it in the satisfiability formula).

$$Inf \models i \text{ and } Vrs \models v$$

We can describe the system in which a virus is revealed by

$$Infect \overset{def}{=} v|\top$$

while the situation in which information is revealed by

$$Update \overset{def}{=} i|\top$$

We can describe the Inbox, using composed actions, by:

$$Inb \models \langle !open\rangle\langle !run\rangle 0.$$

Because in our logic the following holds

$$\vdash \langle !\sigma\rangle\phi \to \langle\sigma\rangle\top$$

we obtain, based on the soundness results, that

$$Inb \models K_{Inb}\langle open\rangle\langle run\rangle\top.$$

This means that putting the Inbox in any environment we obtain a system that satisfies:

$$Environment|Inb \models \langle open\rangle\langle run\rangle\top.$$

The use of the epistemic operator allows us to identify the information that characterized the presence of a known subsystem. As our logic permits analysis of smaller systems inside bigger systems and so on, we can hierarchically organize the information available in given situations.

The interaction between the Inbox and a harmless message can be described by

$$Inb|Msg \models \langle open\rangle\langle\overline{open}\rangle\langle run\rangle\langle\overline{run}\rangle i$$

Further we can use the syntactic characterization of satisfiability, axiom D7 and the theorem

$$\vdash K_Q\top \rightarrow \phi \text{ implies } \vdash K_Q\top \rightarrow K_Q\phi$$

to derive the specification

$$Inb|Msg \models K_{Inb|Msg}\langle open\rangle\langle\overline{open}\rangle\langle run\rangle\langle\overline{run}\rangle Update$$

But $\vdash \top|K_Q\phi \rightarrow K_Q\phi$, and $\vdash K_Q\phi \rightarrow \phi$, hence

$$Environment|Inb|Msg \models \langle open\rangle\langle\overline{open}\rangle\langle run\rangle\langle\overline{run}\rangle Update$$

Hence we can prove that $\langle open\rangle\langle\overline{open}\rangle\langle run\rangle\langle\overline{run}\rangle Update$ is a local property of the subsystem $Inb|Msg$, and that any upper-system satisfies it.

Similarly, the interaction between the Inbox and a spam containing a virus can be described by

$$Inb|Spm \models \langle open\rangle\langle\overline{open}\rangle\langle run\rangle\langle\overline{run}\rangle v$$

wherefrom, as before, we can derive the property

$$Inb|Spm \models K_{Inb|Spm}\langle open\rangle\langle\overline{open}\rangle\langle run\rangle\langle\overline{run}\rangle Infect$$

that expresses the fact that a system containing the Inbox in parallel with a spam can be infected by a virus, i.e. that:

$$Environment|Inb|Spm \models \langle open\rangle\langle\overline{open}\rangle\langle run\rangle\langle\overline{run}\rangle Infect$$

All this provides a powerful tool to express and argue over the properties of a system such as $Environment|Msg|Spm|Inb$, even if $Environment$ is an unknown process, by analyzing its subsystems. So, this system satisfies the properties of $Environment|Inb|Spm$ but also of $Environment|Inb|Msg$, or of $Environment|Inb$, etc. All these systems satisfy

$$Environment|Msg|Inb|Spm \models [open][\overline{open}][run][\overline{run}](Update \vee Infect)$$

Spatial axioms

Axiom E 1. $\vdash \top | \bot \to \bot$

Axiom E 2. $\vdash (\phi|\psi)|\rho \to \phi|(\psi|\rho)$

Axiom E 3. $\vdash \phi|0 \leftrightarrow \phi$

Axiom E 4. $\vdash \phi|(\psi \vee \rho) \to (\phi|\psi) \vee (\phi|\rho)$

Axiom E 5. $\vdash \phi|\psi \to \psi|\phi$

Axiom E 6. $\vdash (c_P \wedge \phi|\psi) \to \bigvee_{P \equiv Q|R}(c_Q \wedge \phi)|(c_R \wedge \psi)$

Spatial rules

Rule E$_R$ 1. *If* $\vdash \phi \to \psi$ *then* $\vdash \phi|\rho \to \psi|\rho$

Dynamic axioms

Axiom E 7. $\vdash \langle\alpha\rangle\phi|\psi \to \langle\alpha\rangle(\phi|\psi)$

Axiom E 8. $\vdash [\alpha](\phi \to \psi) \to ([\alpha]\phi \to [a]\psi)$

Axiom E 9. $\vdash 0 \to [\alpha]\bot$

Axiom E 10. *If* $\beta \neq \alpha_i$ *for* $i = 1..n$ *then* $\vdash \langle !\alpha_1\rangle\top|...|\langle !\alpha_n\rangle\top \to [\beta]\bot$

Axiom E 11. $\vdash \langle !\alpha\rangle\phi \to [\alpha]\phi$

Dynamic rules

Rule E$_R$ 2. *If* $\vdash \phi$ *then* $\vdash [\alpha]\phi$

Rule E$_R$ 3. *If* $\vdash \phi \to [\alpha]\phi'$ *and* $\vdash \psi \to [\alpha]\psi'$ *then* $\vdash \phi|\psi \to [\alpha](\phi'|\psi \vee \phi|\psi')$.

Rule E$_R$ 4. *If* $\vdash \bigvee_{[\![M]\!] \leq (\!|\phi|\!)} c_{\overline{M}} \to \phi$ *then* $\vdash \phi$.

Epistemic axioms

Axiom E 12. $\vdash K_Q\phi \wedge K_Q(\phi \to \psi) \to K_Q\psi$

Axiom E 13. $\vdash K_Q\phi \to \phi$

Axiom E 14. $\vdash K_Q\phi \to K_Q K_Q\phi$.

Axiom E 15. $\vdash K_Q\top \to (\neg K_Q\phi \to K_Q\neg K_Q\phi)$

Axiom E 16. *If* $P \in \mathfrak{S}$ *then* $\vdash K_P\top \leftrightarrow c_P|\top$

Axiom E 17. $\vdash K_Q\phi \leftrightarrow (K_Q\top \wedge K_0(K_Q\top \to \phi))$

Axiom E 18. $\vdash K_0\phi \wedge \psi|\rho \to (K_0\phi \wedge \psi)|(K_0\phi \wedge \rho)$

Axiom E 19. $\vdash K_0\phi \to [\alpha]K_0\phi$

Axiom E 20. $\vdash K_0\phi \to (K_Q\top \to K_Q K_0\phi)$

Epistemic rules

Rule $\mathbf{E}_R$ 5. *If* $\vdash \phi$ *then* $\vdash K_Q\top \to K_Q\phi$.

Rule $\mathbf{E}_R$ 6. *If* $\mathcal{M} \ni P$ *is a finite context and* $\vdash c_{\mathcal{M}} \wedge c_P \to K_0\phi$ *then* $\vdash c_{\mathcal{M}} \to \phi$.

References

[1] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese (: Special Section: Knowledge, Rationality and Action).Editors: J. Symons, J. Hintikka. Special Section Editor: W. van der Hoek. Springer Science+Business Media B.V. ISSN: 0039-7857*, 139 (2):165–224, 2004.

[2] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements. common knowledge and private suspicions. *CWI Technical Report SEN-R9922*, 1999.

[3] J. A. Bergstra. *Handbook of Process Algebra.* Elsevier Science Inc., New York, NY, USA, 2001.

[4] Luis Caires. Behavioral and spatial properties in a logic for the pi-calculus. *In Igor Walukiewicz, editor, Proc. of Foundations of Software Science and Computation Structures2004, Lecture Notes in Computer Science, Springer-Verlag*, vol:2987, 2004.

[5] Luis Caires and Luca Cardelli. A spatial logic for concurrency (part ii). *In Proceedings of CONCUR'2002, Lecture Notes in Computer Science, Springer-Verlag*, vol:2421, 2002.

[6] Luis Caires and Luca Cardelli. A spatial logic for concurrency (part i). *Information and Computation*, Vol: 186/2:194–235, November 2003.

[7] Luis Caires and Etienne Lozes. Elimination of quantifiers and decidability in spatial logics for concurrency. *In Proceedings of CONCUR'2004, Lecture Notes in Computer Science, Springer-Verlag*, vol:3170, 2004.

[8] Cristiano Calcagno, Luca Cardelli, and Andrew D. Gordon. Deciding validity in a spatial logic for trees. *In Proceedings of the ACM Workshop on Types in Language Design and Implementation*, pages 62–73, 2003.

[9] Luca Cardelli. Bioware languages. *In: Andrew Herbert, Karen Sprck Jones (Eds.): Computer Systems: Theory, Technology, and Applications - A Tribute to Roger Needham, Monographs in Computer Science. Springer*, ISBN 0-387-20170-X.:59–65., 2004.

[10] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98.* Springer-Verlag, Berlin Germany, 1998.

[11] Luca Cardelli and Andrew D. Gordon. Ambient logic. *To appear in Mathematical Structures in Computer Science*, 2003.

[12] Witold Charatonik, Andrew D. Gordon, and Jean-Marc Talbot. Finite-control mobile ambients. In *ESOP '02: Proceedings of the 11th European Symposium on Programming Languages and Systems*, pages 295–313. Springer-Verlag, 2002.

[13] Witold Charatonik and Jean-Marc Talbot. The decidability of model checking mobile ambients. *Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic, Springer-Verlag*, 2142 of Lecture Notes in Computer Science:339–354, 2001.

[14] M. Dam. Proof systems for π-calculus. *In de Queiroz, editor, Logic for Concurrency and Synchronisation, Studies in Logic and Computation. Oxford University Press. To appear.*

[15] M. Dam. Relevance logic and concurrent composition. *In Proceedings of Third Annual Symposium on Logic in Computer Science, Edinburgh, Scotland, July 1988. IEEE Computer Society.*, pages 178–185.

[16] M. Dam. Model checking mobile processes. *Information and Computation*, vol:129(1):35–51, 1996.

[17] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[18] D Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* MIT Press, 2000.

[19] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *JACM*, vol: 32(1):137–161, 1985.

[20] W. Groeneveld J. Gerbrandy. Reasoning about information change. *Journal of Logic, Language and Information*, 6:146–169, 1997.

[21] R. Mardare and C. Priami. A logical approach to security in the context of ambient calculus. *ENTCS*, vol. 99, 2004.

[22] R. Mardare, C. Priami, P. Quaglia, and A. Vagin. Model checking biological systems described using ambient calculus. *Proceedings of CMSB04, Lecture Notes in BioInformatics. Berlin: Springer-Verlag*, 3082: 3:85– 10, 2005.

[23] Radu Mardare. A decidable extension of hennessy-milner logic with spatial operators. *Technical Report DIT-06-009, Informatica e Telecomunicationi, University of Trento*, 2006.

[24] Radu Mardare. Dynamic epistemic spatial logics. *Technical Report, DIT-06-010, Department of Information and Communication Technology, University of Trento*, available at http://dit.unitn.it/research/publications/index.xml and also at http://www.dit.unitn.it/∼mardare/publications.htm, 2006.

[25] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, vol:114:149–171, 1993.

[26] Gordon D. Plotkin. A structural approach to operational semantics. *Technical Report FN-19, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark*, 43, September 1981.

[27] Colin Stirling. *Modal and temporal properties of processes.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[28] J. F. A. K. van Benthem. Games in dynamic epistemic logic. *Bulletin of Economic Research, Los Altos*, 53(4):219–248, 2001.

[29] J. F. A. K. van Benthem. Logic for information update. *In Proceedings of TARK01, Los Altos*, 2001.

[30] H. van Ditmarsch. Knowledge games. *Bulletin of Economic Research*, 53(4):249–273, 2001.